GC20-1755-3
File No. S370-01

A Guide to the IBM System/370 Model 168 for System/370 Model 165 Users

Systems

IBM

GC20-1755-3
File No. S370-01

Systems

A Guide to the IBM System/370 Model 168 for System/370 Model 165 Users

This guide presents hardware, programming systems, and other pertinent information about the IBM System/370 Model 168 that describes its significant new features and advantages. Knowledge of the IBM System/370 Model 165 is assumed. Features common to Models 165 and 168 are indicated but not discussed in detail. The contents of the guide are intended to acquaint the reader with the Model 168 and to be of benefit in planning for its installation.

Associated with this guide are three optional supplements that describe operating systems for the Model 168 that support a virtual storage environment. Each supplement has its own form number and must be ordered individually, if required. Optional supplements are the following:

- *OS/Virtual Storage 1 Features Supplement* (GC20-1752)
- *OS/Virtual Storage 2 Single Virtual Storage (SVS) Features Supplement* (GC20-1753)
- *Virtual Machine Facility/370 Features Supplement* (GC20-1757)

IBM

Fourth Edition (September 1976)

This is a major revision obsoleting GC20-1755-2. Text has been added to include information about the multiprocessing feature and Attached Processor Systems. Miscellaneous other changes have been made throughout the publication as well. Changes to the text and illustrations are indicated by a vertical line in the left margin.

This guide is intended for planning purposes only. It will be updated from time to time; however, the reader should remember that the authoritative sources of system information are the system library publications for the Model 168, its associated components and its programming support. These publications will first reflect any changes.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form has been provided at the back of this publication for readers' comments. If this form has been removed, address comments to: IBM Corporation, Technical Publications/Systems, Dept. 824, 1133 Westchester Avenue, White Plains, New York 10604. Comments become the property of IBM.

PREFACE

It is assumed that the reader of this publication is familiar with System/370 Model 165 hardware features, channels, I/O devices, and programming support as described in A Guide to the IBM System/370 Model 165, GC20-1730, and/or system library publications concerning Model 165 hardware and programming systems support. This guide discusses in detail only the hardware features of the Model 168 that are different from those of the Model 165 and the programming support provided for new features of the Model 168.

Those familiar with a System/360 model only or a System/370 model other than the Model 165 should obtain A Guide to the IBM System/370 Model 168 for System/360 Users, GC20-1787, which discusses the differences between the Model 168 and the System/360 Model 65.

There are three versions of the Model 168: the Model 1, Model 3, and Model A3. The hardware differences between Model 1 of the Model 168 and the Model 165 are discussed in Sections 01 to 60. The differences between Models 3 and 1 of the Model 168 (both uniprocessor and multiprocessor systems) are discussed in Section 65. The Model A3, which together with the 3062 Attached Processing Unit forms the Model 168 Attached Processor System, is discussed in Section 67.

The Model 168 is not compared with a Model 165 II, which is a purchased Model 165 (storage model J, K, or KJ) with the optional Dynamic Address Translation Facility installed. However, functional descriptions of Model 168 features that are also part of the Dynamic Address Translation Facility of the Model 165 II apply to the Model 165 II as well, unless otherwise noted. This publication applies to systems with 60-cycle power.

The total Model 168 guide consists of this base publication (Sections 01 to 70), which covers virtual storage and virtual machine concepts and Model 168 hardware and I/O devices, and from one to three optional supplements (Sections 90 to 110). The optional supplements describe the facilities of the IBM programming systems that support a virtual storage environment using the dynamic address translation hardware of the Model 168. Each optional supplement has its own unique form number and each supplement desired must be ordered separately and inserted in this base publication, which is distributed without the automatic inclusion of any optional supplements.

The following optional supplements can be inserted in this base publication:

- OS/Virtual Storage 1 Features Supplement (GC20-1752) - assumes knowledge of OS MFT
- OS/Virtual Storage 2 Single Virtual Storage (SVS) Features Supplement (GC20-1753) - assumes knowledge of OS MVT
- Virtual Machine Facility/370 Features Supplement (GC20-1757) - does not assume knowledge of CP-67/CMS

All optional supplements also assume knowledge of virtual storage, dynamic address translation, and other new Model 168 features as described in this base publication or appropriate system library documents. However, no optional supplement requires knowledge of the contents of any other optional supplement.

This base publication, as well as each optional supplement, begins with page 1 and includes its own table of contents and index. The base publication or supplement title is printed at the bottom of each page as a means of identification.

The optional programming systems supplements contain System/370 model-independent information, unless otherwise noted, and are designed to be included in the guides for System/370 Models 135, 145, 158, and 168 as shown below.

Base Publications	Supplements			
	DOS/VS Features Supplement (GC20-1756)	OS/VS1 Features Supplement (GC20-1752)	OS/VS2 SVS Features Supplement (GC20-1753)	VM/370 Features Supplement (GC20-1757)
A Guide to the IBM System/370 Model 135 (GC20-1738-4 or later editions)	X	X		X
A Guide to the IBM System/370 Model 145 (GC20-1734-2 or later editions)	X	X	X	X
A Guide to the IBM System/370 Model 158 for System/370 Model 155 Users (GC20-1754)	X	X	X	X
A Guide to the IBM System/370 Model 158 for System/360 Users (GC20-1781)	X	X	X	X
A Guide to the IBM System/370 Model 168 for System/370 Model 165 Users (GC20-1755)		X	X	X
A Guide to the IBM System/370 Model 168 for System/360 Users (GC20-1787)		X	X	X

Additional, more detailed information regarding System/370 Model 168 hardware and programming systems support can be found in system library publications.

CONTENTS

Base Publication Sections (Sections 01 to 70)

Section 01: System Highlights of Models 1 and 3 1

Section 10: Major Components and System Technology for Models 1 and 3 6

Section 20: Architecture Design and System Components of the Model 1 9

20:05 Architecture Design 9

20:10 The Central Processing Unit 13

Extended Control Mode 13

Monitoring Feature. 19

New Instructions. 19

Clock Comparator and CPU Timer. 20

Reliability, Availability, and Serviceability Features. . 21

20:15 Storage 25

Processor (Main) Storage. 25

High-Speed Buffer Storage 28

20:20 Channels. 32

20:25 System Console. 34

20:30 Standard and Optional System Features 34

Standard Features 34

Optional Features 35

Section 30: Virtual Storage and Dynamic Address Translation 36

30:05 Virtual Storage Concepts, Advantages, and Terminology . . . 36

The Need for Larger Address Space 36

Virtual Storage and Dynamic Address Translation Concepts. 40

General Advantages Offered by IBM Operating Systems that Support a Virtual Storage Environment 46

Virtual Storage and Dynamic Address Translation Terminology 53

30:10 Dynamic Address Translation Hardware for Models 1 and 3 of the Model 168 58

Virtual Storage Organization. 58

Operation of Dynamic Address Translation Hardware 59

Features to Support Demand Paging 66

Channel Indirect Data Addressing. 69

30:15 System Performance in a Virtual Storage Environment 71

System Resources Required to Support a Virtual Storage Environment 72

New Factors that Affect System Performance. 75

Relationship Between Virtual Storage Size and System Performance 78

Increasing System Performance in a Virtual Storage Environment 82

Section 40: Virtual Machines. 86

40:05 Definition and General Operation. 86

40:10 General Advantages of a Virtual Machine Environment 93

Section 50: I/O Devices for Models 1 and 3 96

50:05 I/O Device Support. 96

50:10 3333 Disk Storage and Control Model 11 and 3330 Disk Storage Model 11. 96

Attachment via Integrated Storage Controls. 97

50:15 The 3340 Direct Access Storage Facility 101
3340 Disk Storage Drives and the 3348 Data Module 101
Attachment via 3830 Storage Control Model 2 116
Attachment via Integrated Storage Controls. 119
Intermixing 3340 and 3330-Series Strings on an Attachment. 121
Summary 122

Section 60: Multiprocessing 126
60:05 General Description 126
Introduction. 126
Definition of Multiprocessing 127
The Model 168 Multiprocessing System. 128
Advantages of Model 168 Tightly Coupled Multiprocessing Configurations. 135
Advantages of Loosely Coupled Multiprocessing Configurations. 141
60:10 Model 168 Multiprocessing Architecture. 142
Uniprocessor and Multiprocessor Hardware Differences. . . 142
3068 Multisystem Communication Unit 143
Prefixing 154
CPU Addressing. 155
Time-of-Day Clock 156
Interprocessor Programmed Communication 157
Interprocessor Hardware Communication 158
Channel Reconfiguration Hardware. 160
60:15 Recording and Diagnostic Programs 162
60:20 Planning Considerations 162
Planning for Maximum System Availability. 163

Section 65: Differences Between the Model 3 and the Model 1 165
65:05 Performance Enhancements. 165
65:10 The Service Processor 168
Functions and General Operation 168
Processor Unit. 170
Trace Unit. 171
Counters. 172
Internal Disk File. 172
CE Panel. 173
Printer Control 173
Corporate Standard Interface. 174
Modem 174
Advantages. 175

Section 67: Attached Processor System 177
67:05 System Description. 177
67:10 The 3062 Attached Processing Unit 181
67:15 Uniprocessor and Attached Processor System Hardware Differences 183
67:20 3066 Model 3 System Console 186
67:25 Recording and Diagnostic Routines 187

Section 70: Comparison Tables 189
70:05 Comparison Table of Hardware Features of the System/360 Model 65 and System/370 Models 158 (Models 1 and 3), 165, 165 II, and 168 (Models 1 and 3). 190
70:10 OS and OS/VS Support of the Model 168 (Models 1 and 3) . . 200

Index (Sections 01 to 70). 206

Optional Sections (See each supplement for detailed contents and index)

Section 90: OS/Virtual Storage 1 Features 213

Section 100: OS/Virtual Storage 2 Single Virtual Storage (SVS) Features 215

Section 110: Virtual Machine Facility/370 Features. 217

Note: This guide does not have a Section 80. DOS/Virtual Storage features are discussed in the Section 80 supplement and the Model 168 is not supported by DOS/VS.

FIGURES (Sections 01 to 70)

10.1 System/370 Model 168. 6
10.2 SLT substrate 7
20.10.1 BC mode and EC mode PSW formats 14
20.10.2 Model 168 model-independent fixed storage locations for BC and EC modes 16
20.10.3 Model 168 (Model 1) model-dependent fixed storage locations 17
20.10.4 Model 168 machine check code. 23
20.15.1 Model 168 processor storage organization and configuration panel 27
20.15.2 8K and 16K buffer organization. 30
20.15.3 Model 168 components and controls 31
30.05.1 Names and location of instructions and data in a virtual storage environment 42
30.05.2 Relationship of virtual storage, direct access storage, and real storage. 43
30.05.3 Conceptual illustration of real storage utilization in a mixed batch and online virtual storage environment. . . 52
30.05.4 Layout of virtual storage, external page storage, and real storage. 55
30.10.1 Virtual storage address fields for a 64K segment. 60
30.10.2 Segment table and page tables used for dynamic address translation 62
30.10.3 Dynamic address translation procedure 63
30.10.4 TLB purging when control register 1 is changed. 65
30.10.5 Example of IDALs required for a CCW list when page size is 2K 70
30.15.1 Possible system performance when a virtual storage operating system is used with a Model 168 with the same I/O configuration and real storage size as the replaced Model 165. 74
30.15.2 General effect on page faults of increasing the ratio of virtual storage used to real storage present in the system. 79
30.15.3 General effect on system performance of the paging factor only. 80
30.15.4 General effect of the paging factor on system performance with various active-to-passive page ratios. 81
30.15.5 General system performance curve for a virtual storage environment 82
40.05.1 Conceptual illustration of the real and virtual machine environment that is supported by VM/370 88
40.05.2 Conceptual illustration of the implementation of virtual storage in a virtual machine environment. 90
40.05.3 Segment table and page tables built when a virtual storage operating system executes in a virtual machine. . 91
50.10.1 Permissible 3330-series string configurations for the Model 168 Integrated Storage Controls feature 99

50.10.2 Sample 3330-series string configuration with string switching 100
50.15.1 A five-drive 3340 string with 3340 Model A2, B2, and B1 units. 102
50.15.2 The 3348 Data Module. 102
50.15.3 Location of physical and logical tracks and read/write heads on a data surface in a 3348 Data Module 106
50.15.4 Cylinder and read/write head layout for a 3348 Model 35 Data Module. 108
50.15.5 Cylinder and read/write head layout for a 3348 Model 70 Data Module. 109
50.15.6 Cylinder and read/write head layout for a 3348 Model 70F Data Module 111
50.15.7 A Model 168 configuration with 3340 disk storage attached via 3830 Storage Control Model 2 116
50.15.8 String switching for the 3340 facilities attached to a 3830 Model 2. 118
50.15.9 Permissible 3340 string configurations for the Model 168 Integrated Storage Controls feature 120
50.15.10 String switching for 3340 facilities attached to one ISC 121
60.05.1 Model 168 Model 1 multiprocessing system. 129
60.10.1 Configuration control panel for a Model 168 multiprocessing system. 144
60.10.2 Storage elements and floating storage addressing. 146
65.05.1 High-speed buffer and processor storage organization in the Model 3. 167
65.10.1 Components of the service processor in a Model 3. 169
67.05.1 Components of the Model 168 Attached Processor System . . 178
67.10.1 Organization of the Model A3 CPU and 3062 APU 182

<u>TABLES</u> (Sections 01 to 70)

20.15.1 Model 168 uniprocessor cycle and access times 26
20.20.1 Permissible configurations and channel priorities for highest speed System/370 I/O devices in uniprocessor systems 33
30.10.1 Number and size of segments and pages for a 16-million-byte virtual storage. 58
30.10.2 Virtual and real storage addresses used by and supplied to programs in the Model 168. 67
50.10.1 Capacity and timing characteristics for 3330-series drives. 97
50.10.2 3336 Model 1 and 11 Disk Pack characteristics 97
50.15.1 Physical and capacity characteristics of 3348 Data Modules and the 2316 Disk Pack. 113
50.15.2 Timing characteristics of the 3340 direct access storage facility and the 2314 facility. 114
50.15.3 Summary of the hardware features of 3340 and 2314 disk storage facilities 123
50.15.4 Summary of the features of 3830 Storage Control Models 1 and 2 and Integrated Storage Controls 124
60.05.1 Processor storage access and fetch times for multiprocessor models in nanoseconds. 131
60.05.2 Permissible configurations and channel priorities for highest speed System/370 I/O devices in a Model 168 tightly coupled multiprocessing configuration 134
67.15.1 Comparison of hardware features in the Model 3 CPU (uniprocessor and multiprocessor models), Model A3 CPU, and 3062 APU 185

SECTION 01: SYSTEM HIGHLIGHTS OF MODELS 1 AND 3

The System/370 Model 168 is an advanced function growth system for System/360 Models 65, 67, and 75 and System/370 Models 155, 158, and 165. The Model 168 provides major new functions that are not basic to System/360 architecture. The Model 168 has new features and new programming systems support that are designed to facilitate application development and maintenance. In addition, a Model 168 and its new programming support can ease entry into, and expansion of, online data processing operations.

Both uniprocessor and multiprocessor models of the Model 168 CPU are available. Two multiprocessor models can be connected to form a tightly coupled multiprocessing configuration. Thus, the Model 168 is also a growth system for System/360 Model 65 and System/370 Model 158 tightly coupled multiprocessing systems. Model 168 uniprocessor and multiprocessor configurations can be combined with each other and other System/370 models to form loosely coupled multiprocessing configurations.

The Model 168 makes new functions available to Model 65, 75, 155, and 165 users without requiring a major conversion effort, since the Model 168 is upward compatible with these models. Existing System/360 operating systems that support these models, namely OS MFT and MVT, support the Model 168. However, the Model 168 has standard features that are designed to support a virtual storage environment, and new versions of OS are provided that use these features.

Compatible growth from a System/360 operating system to a Model 168 virtual storage environment can be achieved using the new System/370 operating systems: OS/Virtual Storage 1 (OS/VS1) and OS/Virtual Storage 2 (OS/VS2), which are based on OS MFT and OS MVT, respectively. These operating systems will run only on System/370 models with extended System/370 functions, namely on those with extended control mode of system operation and dynamic address translation facilities. They cannot operate on System/360 models. In addition to implementing virtual storage, the System/370 operating systems offer many other new capabilities and performance-oriented enhancements that are not provided by OS MFT or MVT.

A virtual machine environment is supported by Virtual Machine Facility/370 (VM/370), the successor to CP-67/CMS for System/370. While CP-67/CMS is available only to Model 67 System/360 users, VM/370 operates on System/370 Models 135, 145, 155 II, 158, 165 II, and 168. Model 67 users who have CP-67/CMS installed can use VM/370 on a Model 168 with some conversion effort. The Virtual Machine Assist RPQ can be installed on a Model 168 (or a Model 165 II) to improve the performance of certain operating systems that execute in a virtual machine under VM/370 control.

Transition with little or no reprogramming is also provided for Model 65, 67, and 165 users who are emulating 7070-, 7080-, or 7090-series systems under OS MFT or MVT and for users with these systems installed, since the integrated emulators for 7000-series systems are also supported by OS/VS1 and OS/VS2.

Three models, 1, 3, and A3, of the Model 168 CPU are provided. The Model 3 is an advanced version of the Model 1. The Model 3 has hardware features that give it faster internal performance and higher availability than the Model 1. The new hardware features of the Model 3 consist of internal implementation differences in the Model 168 CPU and a larger high-speed buffer and do not require any programming support. Thus, programs that execute correctly on the Model 1 will execute correctly on the Model 3 without any programming changes assuming they have no timing dependencies and do not access model-dependent logout areas that differ in the two models.

The Model A3 CPU is used in Model 168 Attached Processor Systems. An Attached Processor System consists of a 3062 Attached Processing Unit (APU) connected to a Model A3 CPU. The 3062 APU is an instruction processor with instruction execution facilities equivalent to those of a Model 168 Model 3 CPU, with a few exceptions. The 3062 APU does not contain any processor storage and has no channels attached. The Model 3 CPU and 3062 APU operate together as a single tightly coupled system with shared storage under the control of a single multiprocessing operating system.

Like a tightly coupled multiprocessing configuration, the Attached Processor System provides the capability of executing two instruction streams (tasks) simultaneously. It provides internal performance improvements for uniprocessor Model 168 users. An Attached Processor System can be included in loosely coupled multiprocessing configurations. Details of this Model 168 system are covered in Section 67.

Highlights of the Model 168, when compared with a Model 165, are as follows (features apply to the Model 1 and the Model 3 unless otherwise noted).

- A basic control (BC) mode and an extended control (EC) mode of system operation are standard. Only BC mode is provided in the Model 165. EC mode of operation provides additional system control and supports new functions that are not provided in System/360 or a Model 165.

- Internal performance of a Model 168 operating in BC mode is faster than that of a Model 165. The instruction execution rate of the Model 168 Model 1 is generally in the range of 10 to 30 percent faster than that of the Model 165 when identical system configurations, identical programs, and the same operating system are used. The increased internal performance of the Model 1 results primarily from the significantly faster cycle times of processor storage in the Model 168.

  The internal performance of Model 3 of the Model 168 is generally in the range of 5 to 13 percent faster than that of a Model 1 with a 16K high-speed buffer when identical system configurations, identical programs, and the same operating system are used, and 2K pages are not used. The increase in the internal performance of a Model 3 is somewhat greater when its performance is compared with that of a Model 1 having an 8K buffer. The increase in Model 3 internal performance is the result of a standard 32K high-speed buffer and improved execution times for certain instructions and all interruptions.

- Dynamic address translation (DAT) is a standard facility that can be made operative only when the Model 168 is in EC mode. It provides hardware translation of addresses during program execution. One virtual storage of up to 16 million bytes or multiple virtual storages of up to 16 million bytes each can be supported using DAT hardware. (The amount of virtual storage that can be efficiently supported by a Model 168 depends on the hardware configuration and job stream characteristics.) The optional channel indirect data addressing feature must be installed on 2860, 2870, and 2880 channels when dynamic address translation is used. Channel indirect data addressing enables the channels to access an I/O buffer that is contained in noncontiguous processor storage areas.

- Program event recording (PER) is standard and can be made operative when the Model 168 is in EC mode. It is designed to be used as a problem determination aid. This feature includes hardware that monitors the following during program execution: successful branches, the alteration of general registers, and instruction fetches from and alterations of specified areas of processor storage.

- A monitoring feature is standard that can be used to trace user-defined program events for the purpose of debugging or statistics gathering.

- A CPU timer and clock comparator are standard. The CPU timer provides an interval timing capability similar to that of the interval timer at location 80 but it is updated every microsecond, as is the time of day clock. The clock comparator can be used to cause an interruption when the time of day clock passes a specified value. These items provide higher resolution timing facilities than the interval timer and enable more efficient timing services routines to be written.

- New instructions that support dynamic address translation, the new timing hardware, and system control facilities are added to the System/370 instructions available for the Model 165.

- Processor storage is implemented using monolithic technology instead of discrete ferrite cores, and a Model 168 can have five million more bytes than a Model 165. Processor storage sizes of 1024K, 2048K, 3072K, 4096K, 5120K, 6144K, 7168K, and 8192K are available for the Model 168. Monolithic storage for the Model 168 is faster and more compact than core storage for the Model 165. As in a Model 165, processor storage in a Model 168 is four-way doubleword interleaved.

  The physical size of a Model 168 CPU is not a function of the amount of processor storage installed. A Model 168 is smaller than a Model 165 with 512K and, therefore, is significantly smaller than Model 165 CPU's with more than 512K installed.

- The optional Power Warning feature, when installed on a Model 168 with uninterrupted power supplies, provides a warning machine check interruption when the utility-supplied power is approximately 18 percent below the rated voltage. Program support of this interruption, which is provided by OS MVT (Releases 21.6, 21.7, and 21.8), OS/VS1 (as of Release 3), and OS/VS2 (Releases 1.6 and up), is designed to permit an orderly system shutdown after a power line disturbance occurs, when necessary, so that operations can be restarted once the power supply is stabilized.

- A high-speed buffer of 32K bytes is standard in Model 3 of the Model 168. Model 1 of the Model 168, like the Model 165, has an 8K buffer as standard and optionally a 16K buffer.

- A multiprocessing feature, not available for Model 165 systems, is optional for the Model 168. When installed, this feature permits two Model 168 CPUs, any combination of Models 1 and 3, to be connected to a 3068 Multisystem Communication Unit to form a tightly coupled multiprocessing configuration that shares all available processor storage up to a maximum of 16 megabytes (8 megabytes per CPU). Model 168 tightly coupled multiprocessing configurations are supported by OS/VS2 MVS (Releases 2 and up).

- The maximum aggregate channel data rate a Model 168 can support is significantly increased over that supported by a Model 165 because of the faster cycle time of processor storage and the new channel dual I/O bus that is used to transfer data from the channels to the storage control unit. A Model 168 configuration can handle a maximum aggregate data rate of 17 megabytes per second (MB/sec). The maximum aggregate data rate possible on a Model 165 is 9.4 MB/sec.

- 3330-series disk storage (all models), 3340, 3344, and 3350 direct access storage can be attached to a 2880 channel on a Model 168 via the Integrated Storage Controls (ISC) feature as well as via 3830 Storage Control (Models 1 and 2). The optional ISC feature provides dual direct access storage control functions equivalent to two 3830 Storage Control Model 2 units, with the exception of four-channel switching. Four strings of from two to eight drives each can be attached to each of the two logical storage controls for a total of eight strings (64 drives) attached via the ISC feature. Optionally, the staging adapter feature can be installed on the ISC to permit attachment of the 3850 Mass Storage System via ISC instead of via 3830 Storage Control Model 3.

- The 3340 Direct Access Storage Facility can be attached to the Model 168 via 3830 Storage Control Model 2 and the Integrated Storage Controls feature. The 3340 facility is intermediate capacity direct access storage that, because of its unique design and advanced technology, offers advantages over 2314 disk storage in addition to those provided by 3330-series disk storage. Automatic error correction features and multiple requesting are standard on the 3340. Rotational position sensing is optional.

  The storage medium for 3340 disk storage is the removable interchangeable 3348 Data Module, a sealed cartridge that is never opened by the operator. In addition to the disks on which data is written, the 3348 Data Module contains a spindle, access arms, and read/write heads. The 3340 Disk Storage Drive contains the mechanical and electrical components required to operate the 3348 Data Module.

  The 3340 facility has an 885 KB/sec data transfer rate, average seek time of 25 ms, and full rotation time of 20.2 ms. A 3348 Data Module has a maximum capacity of approximately 35 million bytes or 70 million bytes, depending on the model. One model of the 3348 offers fixed heads for zero seek time to approximately 502,000 bytes maximum and movable heads for an average seek time of 25 ms to the remaining bytes in the data module. A string of from two to eight 3340 drives can be configured. From one to four strings can be attached to the 3830 Model 2 and to each of the logical controls in ISC. Any model of the 3348 can be mounted on a 3340 drive. Therefore, 3340 string capacity can vary from 70 million to 560 million bytes in increments of 35 and/or 70 million bytes.

  The sealed cartridge design of the 3340 facility offers the advantages of multiple capacities per 3340 drive, increased data reliability, and simplified data module loading and unloading procedures.

- 3344 Direct Access Storage can be attached to a Model 168 via 3830 Storage Control Model 2 and Integrated Storage Controls. It offers significantly increased maximum online capacity per drive for 3340 users without the necessity of program conversion. The 3344 is fixed media disk storage. Data is recorded on nonremovable disks. The 3344 is designed to eliminate operator handling, eliminate exposure to external contamination (like the 3348 Data Module), and provide high reliability.

  The 3344 has the same data transfer rate, average seek time, and full rotation time as the 3340. However, the maximum capacity of a

3344 drive is 280 megabytes, or the equivalent of four 70-million-byte 3348 Data Modules. The 3344 is a two-drive unit that attaches to the 3340 Model A2. A 3340/3344 string can contain any mixture of 3344 and 3340 units (as long as the first is a 3340 Model A2) for a maximum of eight drives with a maximum capacity of over 1.8 billion bytes.

Automatic error correction, rotational position sensing, and multiple requesting are standard in the 3344. Fixed head models are also available that contain fixed heads for zero access time to a portion of the data and movable heads for access to the balance of the data.

- 3350 Direct Access Storage can be attached to a Model 168 via 3830 Storage Control Model 2 and Integrated Storage Controls. The 3350 is very large capacity, high-speed, fixed media direct access storage. Data is stored on nonremovable disks. The 3350 is designed to eliminate operator handling, eliminate exposure to external contamination, and provide high reliability.

  The 3350 has a data transfer rate of 1198 KB/sec, average seek time of 25 ms, and full rotation time of 16.8 ms. A 3350 drive operating in native mode has a maximum capacity of 317.5 megabytes. A 3350 string can contain from two to eight drives in two-drive increments for a maximum string capacity of over 2.5 billion bytes of online disk storage.

  The Standard Selective Format feature enables the format of each 3350 to be set during volume initialization. A 3350 drive can operate in 3350 native mode, 3330 Model 1 compatibility mode, or 3330 Model 11 compatibility mode. When operating in 3330 Model 1 compatibility mode, a 3350 drive is the equivalent of two 3330 Model 1 drives in capacity. When operating in 3330 Model 11 compatibility mode, a 3350 drive is the equivalent of one 3330 Model 11 drive in capacity. This feature enables 3330-series users to obtain the price performance and functional advantages of the 3350 without program conversion.

  Automatic error correction, rotational position sensing, and multiple requesting features are standard. The 3350 is also available in fixed head models. These models provide fixed heads for zero access time to a portion of the data and movable heads for access to the balance of the data.

- A service processor unit is standard in the Model 3. This unit provides status data that is designed to improve problem analysis by the local customer engineer as well as facilities that improve the remote problem analysis capability available for a Model 3. It is also a replacement for the optional 2955 Remote Analysis Unit that is available for the Model 165 and Model 1 of the Model 168.

The Model 168 is designed primarily to support a virtual storage environment that allows programmers to write and execute programs that are larger than the processor storage available to them. When virtual storage is supported, restraints normally imposed by the amount of processor storage actually available in a system are eased. The removal of certain restraints can enable applications to be installed more easily, and can be valuable in the installation and operation of online applications. While some of the new hardware features of the Model 168 and some of the new facilities supported by System/370 operating systems are designed to improve performance, a virtual storage environment is designed primarily to help improve the productivity of data processing personnel and enhance the operational flexibility of the installation.

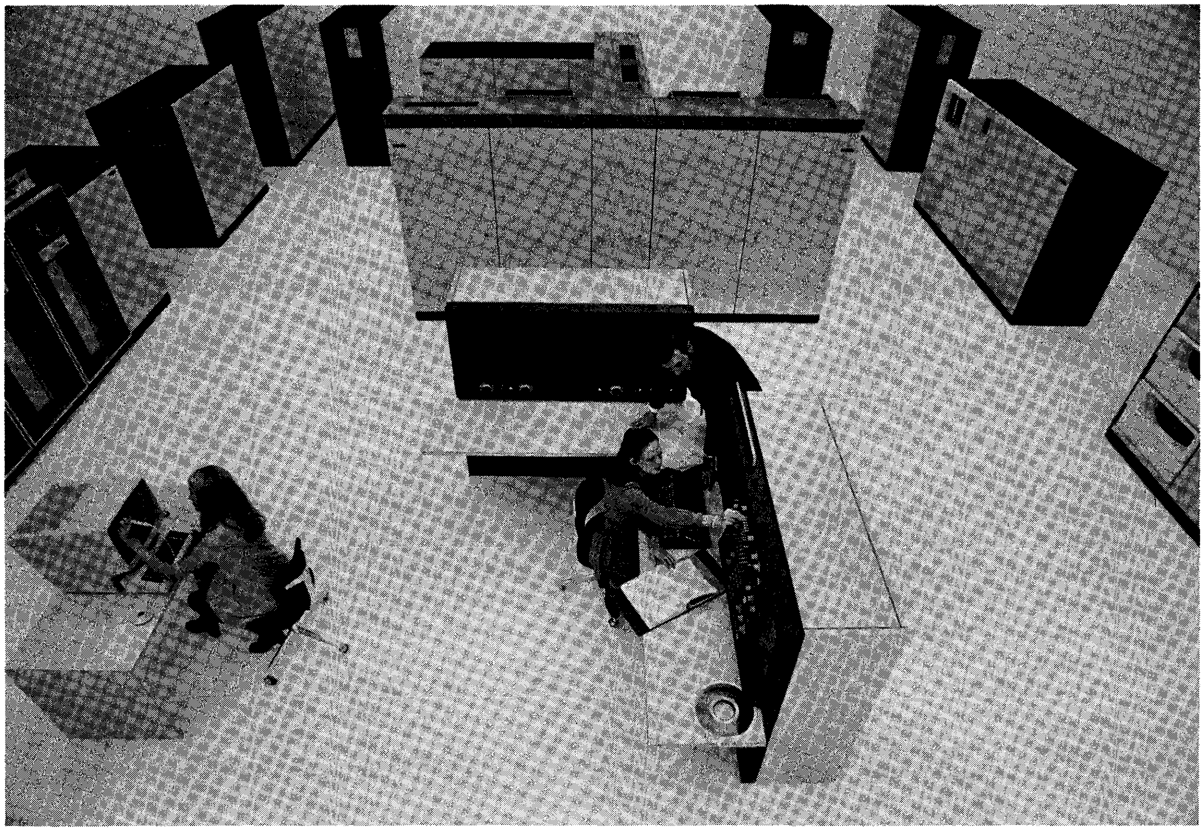


Figure 10.1. System/370 Model 168

Monolithic technology is used to implement nearly all logic and all storage (processor, local, writable control, read-only control, and buffer) in the Model 168. Use of monolithic technology for processor storage, as well as for logic, represents a significant technological advance in storage implementation. The monolithic storage implemented in the Model 168 provides several advantages over the wired, discrete ferrite core storage implemented in the Model 165.

Monolithic storage is similar in design to monolithic logic circuitry, the latter representing a technological advance over the solid logic technology (SLT) introduced with the announcement of System/360. Since the technology associated with monolithic storage is like that used to produce monolithic logic, monolithic storage can be batch-fabricated.

Solid Logic Technology (SLT)

Monolithic technology is a breakaway from the hybrid circuit design concept of SLT and can best be explained by comparison with SLT. As shown in Figure 10.2, SLT circuits were implemented on half-inch ceramic squares called substrates. Metallic lands on the substrate formed interconnections onto which the components were soldered. These components consisted of transistors and diodes, which were integrated on silicon chips about the size of a pinhead, and thin film resistors. An SLT chip usually contained one type of component, and several chips and resistors were needed to form a circuit. In general, an SLT substrate contained a single circuit.

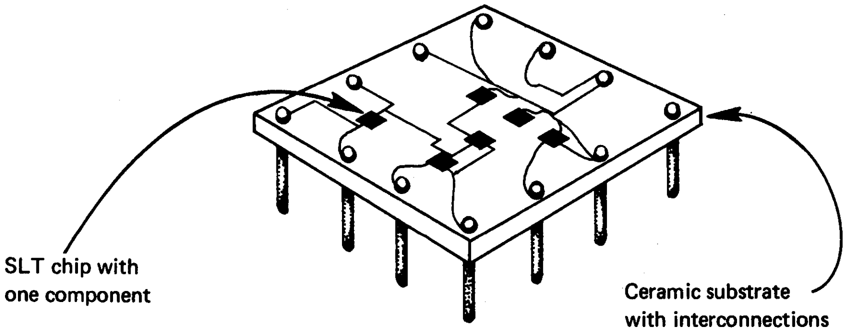

SLT chip with one component
Ceramic substrate with interconnections

Figure 10.2. SLT substrate

Monolithic System Technology (MST)

Monolithic system technology also makes use of a half-inch-square ceramic substrate with metal interconnections onto which chips are placed. However, in monolithic logic circuitry, large numbers of elementary components, such as transistors and resistors, are integrated on a single chip. Unlike an SLT chip, an MST logic chip usually contains several interconnected logic circuits instead of only one component. MST logic modules, each consisting of one substrate, are mounted on circuit cards, which are in turn mounted on circuit boards (as in SLT logic).

MST logic offers the following advantages over SLT:

- MST logic circuitry is intrinsically more reliable because many circuit connections are made on the chip, significantly reducing the number of external connections.

- Faster circuit speeds can be obtained because the path between circuits is considerably shorter.

- Space requirements for logic circuitry are reduced by the significantly higher density of components per chip.

Monolithic Storage

Monolithic storage design incorporates the same concepts described for monolithic logic. However, storage cells that are used to contain storage bits instead of logic circuits are implemented on a metal oxide semiconductor chip. In the Model 168, a monolithic storage array chip is approximately 1/8 by 3/16 of an inch in size and contains a large number of interconnected circuits. These circuits form storage bits and support circuitry (decoding, addressing, and sensing) on the chip.

Since power is required to maintain a one or zero state in a monolithic storage bit, data is lost when power is turned off, and monolithic storage is, therefore, said to be volatile. This is not true of core storage, which retains a magnetized state when power is removed.

The following are the advantages of monolithic over core storage:

- Faster storage speeds are obtained, first, because of the shorter paths between storage circuitry and, second, because of the nondestructive read-out capability of monolithic storage. Since core storage read-out is destructive, a regeneration cycle is required after a read, and a read-out cycle is required before a write. These types of regeneration cycles are not required for monolithic storage.

- Storage serviceability is enhanced because storage is implemented in accessible, easily replaceable cards. Diagnostic routines need only identify the failing storage card, which can be replaced in a matter of minutes.

- Space requirements for system storage are reduced. Dense bit packaging per chip is achieved by the use of monolithic technology and by the fact that the regularity of a storage pattern lends itself to such packaging.

SECTION 20: ARCHITECTURE DESIGN AND SYSTEM COMPONENTS OF THE MODEL 1

20:05 ARCHITECTURE DESIGN

Extended System/370 architecture embodies two different modes of system operation, basic control (BC) mode and extended control (EC) mode, as determined by bit 12 of the current PSW. When a Model 168 operates in BC mode, the contents, layout, and function of permanently assigned processor storage locations 0 to 127 are identical to these locations in System/360 Models 22 and up (except 44 and 67) with the exception of the use of PSW bit 12. BC mode essentially is the System/360-compatible mode of System/370 operation.

When EC mode is operative in the Model 168, the format of the PSW is altered and the number of permanently assigned locations extends beyond processor storage address 127. Changes to the PSW consist of the removal of certain fields to create space for additional mode and mask bits that are required for new functions, such as dynamic address translation and program event recording. The removed fields are assigned to locations above 127 and to a control register.

EC mode is effective when current PSW bit 12 is a one. BC mode is effective when this bit is a zero. BC mode is established during initial program reset. Therefore, a control program must turn on bit 12 of the PSW in order to cause EC mode to become operative. As a result, control and processing programs written for System/360 (Models 22 and up except 44 and 67) will run without modification in BC mode on a System/370 Model 168 (either a Model 1 or a Model 3) that has a comparable hardware configuration, with the following exceptions:

1. Time-dependent programs. (They may or may not execute correctly.)

2. Programs that use machine-dependent data such as that which is logged in the machine-dependent logout area. (OS SER error-logging routines for System/360 models will not execute correctly.)

3. Programs that use the ASCII mode bit in the PSW (bit 12). ASCII mode is not implemented, and this bit is used in System/370 to specify BC or EC mode of operation.

4. Programs that depend on the nonusable lower processor storage area being smaller than 1938 bytes. This area can be reduced to 512 bytes by moving the CPU extended logout area.

5. Programs deliberately written to cause certain program checks.

6. Programs that depend on devices or facilities not implemented in the Model 168.

7. Programs that use model-dependent operations of the System/370 Model 168 that are not necessarily compatible with the same operations on System/360 models.

8. Programs that depend on the validity of storage data after system power has been turned off and then on.

Only BC mode is implemented in the Model 165. Hence, control and processing programs that currently operate on a Model 165 will run without modification in BC mode on a Model 168 (either a Model 1 or a

Model 3) that has a comparable hardware configuration, with the following exceptions:

1. Time-dependent programs. (They may or may not execute correctly.)

2. Programs that depend on the nonusable lower processor storage area being smaller than 1938 bytes. (The nonusable area in the Model 165 is 1504 bytes.)

3. Programs that use machine-dependent data such as that which is logged in the machine-dependent logout area.

4. Programs deliberately written to cause certain program checks.

5. Programs that depend on the validity of storage data after system power has been turned off and then on.

OS control programs are designed to support either BC or EC mode of system operation. OS PCP, MFT, and MVT control programs generated for a Model 65, 67, or 75 support BC mode operations on a Model 168. OS control and processing programs being used on a Model 65, 67, or 75 are subject to the eight compatibility restrictions in the first list. If an OS MFT or MVT control program that was generated for a Model 65, 67, or 75 is used on a Model 168, the system should be set to check stop on machine checks. (Section 60:20 in A Guide to the IBM System/370 Model 165, GC20-1730, discusses the reason.)

OS MFT and MVT support for the Model 168 (Model 1) in BC mode is provided as of Release 21.6. OS MFT and MVT control programs generated for a Model 165 using OS Release 21.6 or later will also operate on a Model 168 to support BC mode of system operation (the Model 168 should be specified as an alternate CPU via the SECMODS macro at system generation). Processing programs that are used on the Model 165 will operate under OS MFT or MVT control on a Model 168 in BC mode subject to the five compatibility restrictions in the second list.

Support of Model 168 (Model 1) systems operating in EC mode is provided by OS/VS1, OS/VS2 Releases 1 and up, and VM/370, each of which is designated as system control programming (SCP). All of these programming systems support a virtual storage environment using dynamic address translation, which operates only when the system is in EC mode. OS/VS2 Releases 1, 1.6, and 1.7 support a single virtual storage (SVS) environment. OS/VS2 Releases 2 and up support multiple virtual storages (an MVS environment), Model 168 Attached Processor Systems, and tightly coupled and loosely coupled multiprocessing configurations. VM/370 supports a virtual machine environment.

User-written processing programs that operate on a Model 165 or Model 168 Model 1 under OS MFT or MVT control can operate under OS/VS1 or OS/VS2 SVS respectively, on a Model 168 (Model 1) with little or no modification, as discussed in the optional programming systems supplements (Sections 90 and 100). Hence, compatible growth from a System/360 or a BC mode nonvirtual storage environment to an EC mode virtual storage environment is provided.

The following are standard features of the Model 168 (Model 1) that are functionally identical to the same features of the Model 165:

- Instruction set that includes System/360 instructions and the following System/370 instructions:

  COMPARE LOGICAL CHARACTERS UNDER MASK
  COMPARE LOGICAL LONG
  INSERT CHARACTERS UNDER MASK
  LOAD CONTROL, STORE CONTROL
  MOVE LONG
  SET CLOCK, STORE CLOCK
  SHIFT AND ROUND DECIMAL
  START I/O FAST RELEASE
  STORE CHANNEL ID
  STORE CHARACTERS UNDER MASK
  STORE CPU ID

- Extended-precision floating point
- Overlap of instruction fetching and preparation with instruction execution (implementation of the instruction and execution units is enhanced in the Model 168)*
- Store and fetch protection
- Multiple control registers (more registers are implemented in the Model 168 than in the Model 165)*
- Interval timer (3.3 millisecond resolution)
- Time-of-day clock
- Byte-oriented operands
- Extended external interruption masking
- Expanded machine check interruption class (additional facilities are provided in the Model 168)*
- Extended channel logout
- Instruction retry, ECC on processor storage, and command retry
- Writable monolithic control storage
- High-speed buffer storage - 8K
- Direct control

The following are optional features of the Model 168 (Model 1) that are functionally identical to the same features on the Model 165:

- High Speed Multiply (increases speed of fixed- and floating-point multiply operations by a factor of approximately two to three)
- Buffer Expansion for the addition of 8K of buffer storage (the 16K buffer has a slightly different organization in the Model 168)*
- 7070/7074 Compatibility
- 7080 Compatibility
- 709/7090/7094II Compatibility
- 2870 Multiplexer Channels and attachment feature, 2860 Selector Channels and attachment feature, and 2880 Block Multiplexer Channels (one 2860, one 2880, or one 2870 with one selector subchannel is required)
- Extended Channels (for up to twelve channels)
- Channel-to-Channel Adapter on 2860 selector channels
- Extended Unit Control Words on 2880 Block Mutliplexer Channels
- 3066 Model 2 System Console (required) - a few new items are provided
- 2955 Remote Analysis Unit

The following are standard features of the Model 168 (Model 1) that are not available for the Model 165:

- New instructions*
  CLEAR I/O
  COMPARE AND SWAP
  COMPARE DOUBLE AND SWAP
  INSERT PSW KEY
  LOAD REAL ADDRESS
  MONITOR CALL
  PURGE TLB
  RESET REFERENCE BIT
  SET CLOCK COMPARATOR
  SET CPU TIMER
  SET PSW KEY FROM ADDRESS
  STORE CLOCK COMPARATOR
  STORE CPU TIMER
  STORE THEN AND SYSTEM MASK
  STORE THEN OR SYSTEM MASK
- EC mode of system operation*
- Dynamic address translation*
- Reference and change recording*
- CPU timer and clock comparator*
- Program event recording*
- Monitoring feature*
- Program interruption for SET SYSTEM MASK instruction*
- Store status function*
- Monolithic read-only control storage (instead of capacitor read-only)*
- Monolithic processor storage (instead of core storage)
- Channel dual I/O bus

*Part of the Dynamic Address Translation Facility of a Model 165 II. The functional descriptions of these items in this publication apply to their implementation in both the Model 168 and the Model 165 II, unless otherwise indicated.

The following are optional features of the Model 168 (Model 1) that are not available for a Model 165:

- Channel Indirect Data Addressing for 2860, 2870, and 2880 channels (required by the virtual storage operating systems and available for the Model 165 II)
- Integrated Storage Controls for attachment of 3330-series, 3340, 3344, or 3350 disk storage, or the 3850 Mass Storage System
- Two-Channel Switch for Integrated Storage Controls
- Staging adapter for Integrated Storage Controls
- Power Warning
- Multiprocessing

All the new features of the Model 168 Model 1 except Integrated Storage Controls, multiprocessing, and those related to implementing virtual storage (such as dynamic address translation and reference and change recording) are discussed in the remainder of this section.

20:10 THE CENTRAL PROCESSING UNIT

Like the Model 165, the Model 168 has a CPU cycle time of 80 nanoseconds and an internal data path that is eight bytes wide. The implementation of local storage (80 nanosecond cycle time), read-only and writable control storage (80 nanosecond cycle times), expanded external interruption masking, and parity checking is the same in the two models. Control registers in addition to the four implemented in the Model 165 are implemented in the Model 168 in order to support new EC-mode-only functions. Additional control registers are implemented in the Model 165 II as well.

Implementation of the instruction and execution units in Models 168 and 165 differs in several aspects in order to provide better overlap of instruction preparation with instruction execution and to provide functions required by new Model 168 hardware features, such as dynamic address translation. (This new implementation is also provided in a Model 165 II.) Significant differences are the following:

- In the Model 168, up to four instructions can be prepared and await execution while one instruction is being executed. The Model 165 can prepare and hold up to three instructions.

- When an incorrect estimate of the success of a conditional branch has been made, the Model 168 can decode the correct instruction one cycle sooner than can the Model 165, if the instruction is presently in an instruction buffer.

- In the Model 168, a doubleword from a given instruction stream can be placed in the instruction buffers every machine cycle. This can be done every other cycle in a Model 165.

- In the Model 168, two registers are provided to hold data that is awaiting placement in processor storage. Each can hold up to eight bytes. The Model 165 has only one such register.

- The instruction unit in the Model 168 includes an instruction pretest function (explained under "Instruction Nullification" in Section 30:10).

- Imprecise interruptions do not occur in a Model 168. In a Model 165, an imprecise interruption occurs if an attempt is made to store data at an invalid storage address or at a storage-protected location. The Model 168 implementation of pretesting (for the dynamic address translation function) also ensures that such conditions do not cause imprecise interruptions in the Model 168.

EXTENDED CONTROL MODE

Extended control mode, unlike basic control mode, is exclusively a System/370 mode and is not implemented in System/360. In a Model 168, the optional Channel Indirect Data Addressing feature must be installed on all standalone channels for the channels to operate with EC mode enabled. Note that IBM-supplied operating systems do not support System/370 models operating in EC mode without dynamic address translation operative also. Facilities that depend on which mode is in effect are discussed below. Any item not covered operates identically in BC and EC modes. (The discussion of EC/BC mode differences applies to the Model 165 II also.)

Change in PSW Format

When a System/370 operates in EC mode, the format of the PSW differs from the BC mode format. Both PSW formats are shown in Figure 20.10.1. In EC mode, the PSW does not contain individual channel mask bits, an instruction length code, or the interruption code for a supervisor call, external, or program interruption. The channel masks are contained in control register 2, and the other fields are allocated permanently assigned locations in fixed processor storage above address 127.

BC MODE PSW FORMAT

Bit	Content	
0	Channel 0 mask	System mask
1	Channel 1 mask	
2	Channel 2 mask	
3	Channel 3 mask	
4	Channel 4 mask	
5	Channel 5 mask	
6	I/O mask	
7	External mask	
8	Protect key	
9		
10		
11		
12	EC/BC mode (0 is BC)	
13	Machine check mask	
14	Wait/running state	
15	Problem/supervisor state	
16	Interruption code	
17		
18		
19		
20		
21		
22		
23		
24		
≈	≈	
30		
31		
32	Instruction length code	
33		
34	Condition code	
35		
36	Program mask	
37		
38		
39		
40	Instruction address	
41		
42		
≈	≈	
61		
62		
63		

EC MODE PSW FORMAT

Bit	Content	
0	0	System mask
1	PER mask	
2	0	
3	0	
4	0	
5	Translation mode (DAT feature mask)	
6	I/O summary mask	
7	External summary mask	
8	Protect key	
9		
10		
11		
12	EC/BC mode (1 is EC)	
13	Machine check mask	
14	Wait/running state	
15	Problem/supervisor state	
16	0	
17	0	
18	Condition code	
19		
20	Program mask	
21		
22		
23		
24	0	
≈	≈	
30		
31		
32	0	
33		
34		
35		
36		
37		
38		
39		
40	Instruction address.	
41		
42		
≈	≈	
61		
62		
63		

Figure 20.10.1. BC mode and EC mode PSW formats

Removal of the fields indicated provides room in the EC mode PSW for control of new features that are unique to EC mode (such as PER and DAT) and for the addition of summary mask bits (such as channel and I/O masks). Use of a single mask bit to control the operation of an entire facility (such as program event recording) or an entire interruption class (such as I/O and external) simplifies the coding required to enable and disable the system for these interruptions.

Change in Permanently Assigned Processor Storage Locations

When a System/370 operates in EC mode, the number of permanently assigned locations in lower processor storage is increased to include fields for storing instruction length codes, interruption codes (for supervisor call, external, and program interruptions), program event recording data, the I/O device address for an I/O interruption, and an exception address for the DAT feature. The model-independent BC mode and EC mode fixed storage areas for System/370 models are shown in Figure 20.10.2. The balance of the fixed area for the Model 168, that which has model-dependent fields, is shown in Figure 20.10.3. This model-dependent area is not affected by whether EC or BC mode is in effect except for locations 185 to 187, which contain the I/O address after an I/O interruption and an IPL only when EC mode is in effect.

The machine check interruption procedure and the format of the data logged on a machine check are the same in EC and BC modes, except for differences in the PSW format and the permanently assigned locations previously discussed.

Expansion of Storage Protect Key Size

The size of the storage protect key associated with each 2K storage block is expanded from five to seven bits in the Model 168. The two additional bits (reference and change) are included for use with dynamic address translation and are discussed in Section 30:10. The SET STORAGE KEY instruction sets a seven-bit key regardless of the mode, BC or EC, in effect. The INSERT STORAGE KEY loads a five-bit or a seven-bit key into the register indicated depending on whether BC or EC mode, respectively, is in effect.

Channel Masking Changes

When a System/370 operates in EC mode, interruptions from each channel are controlled by the summary I/O mask bit in the current PSW (bit 6) and an individual channel mask bit in control register 2. In the Model 168, bits 0 to 11 in control register 2 are assigned to control channels 0 to 11, respectively. Both the summary mask bit and the appropriate individual channel mask bit must be on in order for an interruption from a given channel to occur. In BC mode, only interruptions from channels 6 to 11 are controlled by individual channel mask bits in control register 2 and the I/O mask bit in the PSW. Interruptions from channels 0 to 5 are controlled only by channel mask bits in the current PSW (bits 0 to 5) in BC mode.

BC MODE FIXED AREA 0-159

<table>
<tr><th>Decimal locations</th><th colspan="4"></th></tr>
<tr><td>0</td><td colspan="4">IPL PSW</td></tr>
<tr><td>8</td><td colspan="4">IPL CCW 1</td></tr>
<tr><td>16</td><td colspan="4">IPL CCW 2</td></tr>
<tr><td>24</td><td colspan="4">External old PSW</td></tr>
<tr><td>32</td><td colspan="4">Supervisor call old PSW</td></tr>
<tr><td>40</td><td colspan="4">Program old PSW</td></tr>
<tr><td>48</td><td colspan="4">Machine check old PSW</td></tr>
<tr><td>56</td><td colspan="4">I/O old PSW</td></tr>
<tr><td>64</td><td colspan="4">Channel status word – CSW</td></tr>
<tr><td>72</td><td>Channel address word – CAW</td><td colspan="3">76 Unused</td></tr>
<tr><td>80</td><td>Interval timer</td><td colspan="3">84 Unused</td></tr>
<tr><td>88</td><td colspan="4">External new PSW</td></tr>
<tr><td>96</td><td colspan="4">Supervisor call new PSW</td></tr>
<tr><td>104</td><td colspan="4">Program new PSW</td></tr>
<tr><td>112</td><td colspan="4">Machine check new PSW</td></tr>
<tr><td>120</td><td colspan="4">I/O new PSW</td></tr>
<tr><td>128</td><td>0</td><td colspan="3">132 0</td></tr>
<tr><td>136</td><td>0</td><td colspan="3">140 0</td></tr>
<tr><td>144</td><td>0</td><td>148 0</td><td>Monitor class</td><td>0</td></tr>
<tr><td>152</td><td>0</td><td>156 0</td><td colspan="2">Monitor code</td></tr>
</table>

- Model independent among System/360 and System/370 models in BC mode except for PSW bit 12
- Processed by the control program

EC MODE FIXED AREA 0–159

<table>
<tr><td>0</td><td colspan="7">IPL PSW</td></tr>
<tr><td>8</td><td colspan="7">IPL CCW 1</td></tr>
<tr><td>16</td><td colspan="7">IPL CCW 2</td></tr>
<tr><td>24</td><td colspan="7">External old PSW</td></tr>
<tr><td>32</td><td colspan="7">Supervisor call old PSW</td></tr>
<tr><td>40</td><td colspan="7">Program old PSW</td></tr>
<tr><td>48</td><td colspan="7">Machine check old PSW</td></tr>
<tr><td>56</td><td colspan="7">I/O old PSW</td></tr>
<tr><td>64</td><td colspan="7">Channel status word – CSW</td></tr>
<tr><td>72</td><td colspan="3">Channel address word – CAW</td><td colspan="4">76 Unused</td></tr>
<tr><td>80</td><td colspan="3">Interval timer</td><td colspan="4">84 Unused</td></tr>
<tr><td>88</td><td colspan="7">External new PSW</td></tr>
<tr><td>96</td><td colspan="7">Supervisor call new PSW</td></tr>
<tr><td>104</td><td colspan="7">Program new PSW</td></tr>
<tr><td>112</td><td colspan="7">Machine check new PSW</td></tr>
<tr><td>120</td><td colspan="7">I/O new PSW</td></tr>
<tr><td>128</td><td colspan="3">0</td><td>CPU addr.</td><td>0</td><td colspan="2">External int. code</td></tr>
<tr><td>136</td><td>0</td><td>ILC</td><td>SVC int. code</td><td>140 0</td><td>ILC</td><td colspan="2">Program int. code</td></tr>
<tr><td>144</td><td>0</td><td colspan="2">Translation excp. addr.</td><td>148 0</td><td>Monitor class</td><td>PER code</td><td>0</td></tr>
<tr><td>152</td><td>0</td><td colspan="2">PER address</td><td>156 0</td><td colspan="3">Monitor code</td></tr>
</table>

- Model independent among System/370 models in EC mode
- PSW format is different from that of BC mode PSW
- Processed by the control program

Figure 20.10.2. Model 168 model-independent fixed storage locations for BC and EC modes

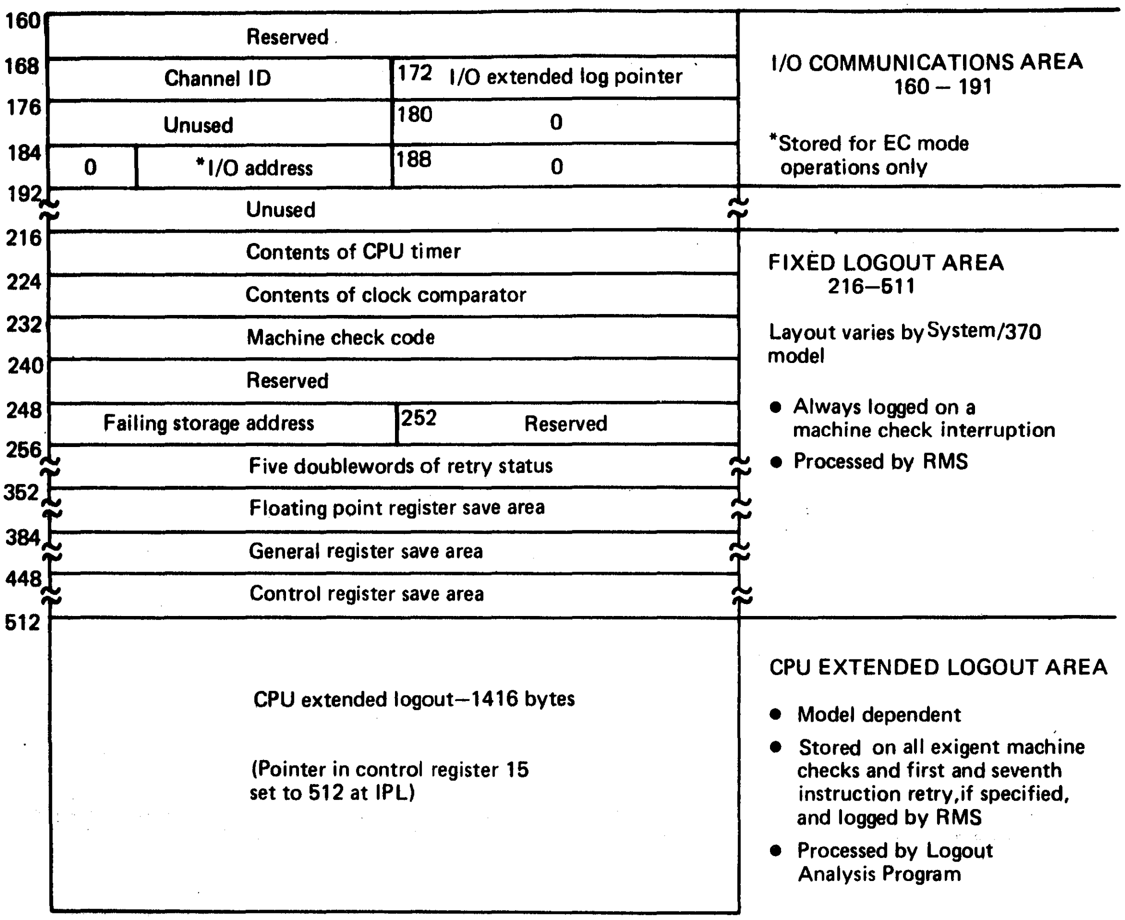

160
Reserved
168
Channel ID
172 I/O extended log pointer
176
Unused
180 0
184
0
*I/O address
188 0
192
Unused
I/O COMMUNICATIONS AREA 160 – 191
*Stored for EC mode operations only
216
Contents of CPU timer
224
Contents of clock comparator
232
Machine check code
240
Reserved
248
Failing storage address
252 Reserved
256
Five doublewords of retry status
352
Floating point register save area
384
General register save area
448
Control register save area
FIXED LOGOUT AREA 216–511
Layout varies by System/370 model
● Always logged on a machine check interruption
● Processed by RMS
512
CPU extended logout—1416 bytes
(Pointer in control register 15 set to 512 at IPL)
CPU EXTENDED LOGOUT AREA
● Model dependent
● Stored on all exigent machine checks and first and seventh instruction retry, if specified, and logged by RMS
● Processed by Logout Analysis Program

Figure 20.10.3. Model 168 (Model 1) model-dependent fixed storage locations

<u>Changes to Certain System/370 Instruction Definitions</u>

All Model 168 instructions are valid in BC and EC modes. However, because of differences between the PSW format and the permanently assigned storage locations in EC and BC modes, the definition of certain instructions is affected. Instructions provided for both System/360 and System/370 whose definition is altered for EC mode are:

BRANCH AND LINK (RR, RX)
INSERT STORAGE KEY
LOAD PSW
SET PROGRAM MASK
SET STORAGE KEY
SET SYSTEM MASK
SUPERVISOR CALL

Revised definitions of these instructions to include BC/EC mode differences are contained in <u>System/370 Principles of Operation</u> (GA22-7000-2, or later editions). Programs that operate in BC mode and that use LOAD PSW and/or SET SYSTEM MASK (SSM) instructions must be modified to operate correctly in EC mode. The eight-byte PSW to be loaded by LPSW instructions and the eight-bit system mask to be set by SSM instructions must be changed to EC mode format. (Programs that use SSM instructions and that are executed in an OS/VS1 or OS/VS2 environment need not be modified because the interruption for SSM instructions and an SSM simulation routine, described next, are supported.)

Programs that use the other instructions listed do not have to be changed to operate correctly in EC mode, unless they use other facilities that are mode dependent. Programs that operate in BC mode and that use the STORE THEN OR SYSTEM MASK and STORE THEN AND SYSTEM MASK instructions (not provided in System/360) must also be modified to operate correctly in EC mode.

Program Interruption for SET SYSTEM MASK Instruction

When a System/370 is operating in EC mode, execution of the SET SYSTEM MASK instruction is under the control of the SSM mask in control register 0. When the SSM mask bit is a one, an attempt to execute an SSM instruction causes a program interruption without execution of the SSM instruction. When the SSM mask bit is a zero, SSM instructions are executed as usual.

This interruption is implemented to enable existing programs that were written for System/360 models or for System/370 BC mode of operation to execute correctly in EC mode without modification of the system mask field addressed by existing SSM instructions. When an interruption occurs for an SSM instruction, the contents of the BC mode format system mask indicated by the SSM instruction can be inspected, and the appropriate EC mode mask bits can then be set by an SSM simulation routine.

The SSM instruction interruption is also implemented for use in Model 168 tightly coupled multiprocessing configurations and Attached Processor Systems. In a two CPU environment, both CPUs operate with the SSM control bit on. This ensures that the OS/VS2 multiprocessing control program will be notified, via an interruption, of any attempt to execute the SSM instruction. This is required for the protection of serially reusable system resources.

Program Event Recording

Program event recording (PER), a standard feature of the Model 168, is designed to assist in program debugging by enabling a program to be alerted to any combination of the following events via a program interruption:

- Successful execution of any type of branch instruction

- Alteration of the contents of the general registers designated by the user

- Fetching of an instruction from a processor storage area defined by the user

- Alteration of the contents of a processor storage area defined by the user

The PER feature can operate only when EC mode is in effect and the PER mask, bit 1 of the current PSW, is on. Control register 9 (bits 0 to 3) is used to specify which of the four PER event types are to be monitored. A PER program interruption is taken after the occurrence of an event only if both the PER mask bit and the respective event mask bit in control register 9 are on. Control register 9 (bits 16 to 31) also specifies which of the 16 general registers are to be monitored if monitoring of this event is specified. Control registers 10 and 11 indicate the beginning address and the ending address, respectively, of the contiguous processor storage area that is to be monitored for instruction fetching and/or alteration.

When an event that is being monitored is detected, PER hardware causes a program interruption, if the PER mask bit is on, and identification of the type of event is stored in the fixed processor storage area (location 150). The address of the instruction associated with the event is also stored (locations 153 to 155). Program event interruptions are lost if they occur when the PER mask bit or the particular event mask bit is off.

If dynamic address translation mode is also specified when PER is active, virtual storage addresses instead of real storage addresses (discussed in Section 30) are placed in the control registers to monitor references to a contiguous virtual storage area.

Note that when PER is enabled to monitor successful branches, general register alterations, or processor storage alterations, significant CPU performance degradation occurs.

MONITORING FEATURE

The monitoring feature is standard on the Model 168 (and on the Model 165 II). This feature provides the capability of monitoring the occurrence of programmed events. For example, monitoring can be used to perform measurement functions (how many times a routine was executed) or tracing functions for the purpose of program debugging (which routines were executed).

The MONITOR CALL instruction is provided with the monitoring feature. Execution of this instruction indicates the occurrence of one of the events being monitored. The operands of the MONITOR CALL instruction permit specification of up to 16 classes of events, each class with up to 16 million unique types of events. The 16 monitor classes are individually maskable via mask bits in control register 8. A program interruption occurs when a MONITOR CALL instruction is executed, if the monitor class indicated is specified in control register 8, and the event identification (class and type) is stored in the fixed storage area.

Both the PER facility and the monitoring feature are provided for debugging purposes. The two features differ from one another in (1) the number of events that can be defined, (2) whether events are defined by the hardware or the programmer, and (3) whether hardware or the programmer checks for the events and causes the interruptions. When PER is used, once the events to be monitored have been designated by the user, CPU hardware checks for the occurrence of the events and causes the interruption. When the monitoring feature is used, the user defines the events to be monitored (up to 16 classes with up to 16 million monitor codes each instead of only four events), determines when the events occur, and causes program interruptions by issuing MONITOR CALL instructions.

NEW INSTRUCTIONS

STORE THEN AND SYSTEM MASK and STORE THEN OR SYSTEM MASK are two new privileged instructions that affect the system mask (bits 0 to 7 in the current PSW). The STORE THEN AND SYSTEM MASK instruction provides, via a single instruction, the capability of storing the current system mask for later restoration, while selectively zeroing certain system mask bits. The STORE THEN OR SYSTEM MASK provides system mask storing and selective setting of system mask bits to ones. These two instructions simplify the coding required to alter the system mask, particularly when the existing settings must be saved.

COMPARE AND SWAP and COMPARE DOUBLE AND SWAP instructions provide the capability of controlling access to a shared real storage area in a multiprogramming or multiprocessing environment. Although the TEST AND SET instruction can also be used for this purpose, these compare instructions enable a program to leave a message when the shared area is in use. This message can be inspected, via a COMPARE AND SWAP instruction, by other programs that share the real storage area. The virtual telecommunications access method (VTAM), OS/VS2 MVS, and VSAM Release 2 use these two instructions.

The INSERT PSW KEY privileged instruction enables a program to place in general register 2 the four-bit storage protection key from the current PSW. The SET PSW KEY FROM ADDRESS privileged instruction enables a program to place a protect key contained in general register 2 or processor storage in the current PSW. When a control program is requested to access a given processor storage location by a problem program, these two instructions can be used by the control program during its processing of the request to determine whether the problem program is authorized to access the specified processor storage location.

The CLEAR I/O privileged instruction can be used together with the HALT DEVICE instruction to terminate all I/O activity on a given channel. CLEAR I/O, INSERT PSW KEY, and SET PSW KEY FROM ADDRESS are used by OS/VS2 MVS.

The new instructions discussed above are provided in the Model 165 II also. Other new instructions provided for the Model 168 are related to specific features (such as monitoring, dynamic address translation, the clock comparator, and the CPU timer) are discussed with these features.

CLOCK COMPARATOR AND CPU TIMER

These timing facilities are standard on the Model 168. (They are also provided in a Model 165 II.) The clock comparator provides a means of causing an external interruption when the time-of-day clock has passed a time specified by a program. This feature can be used to initiate an action, terminate an operation, or inspect an activity, for example, at specific clock times during system operation.

The clock comparator has the same format as the time-of-day clock and is set to zero during initial program reset. The SET CLOCK COMPARATOR privileged instruction is provided to place a value that represents a time-of-day in the clock comparator. When clock comparator interruptions are specified via the external interruption summary mask bit in the current PSW and the clock comparator subclass mask bit in control register 0, an external interruption occurs when the time-of-day clock value is greater than the clock comparator value. Bits 0 to 51 of the time-of-day clock and the clock comparator are compared. If clock comparator interruptions are masked when this condition occurs, the interruption remains pending only as long as the time-of-day clock value remains higher than the value in the clock comparator. The STORE CLOCK COMPARATOR privileged instruction can be used to obtain the current value of the clock comparator.

The use of a clock comparator, instead of the interval timer at location 80, to cause an interruption when a specified time is passed offers two advantages. First, the time-of-day clock increments when the system is in the stopped state, while the interval timer does not. Hence, if a system stop occurs during processing and the system is restarted, the clock comparator can still cause an interruption at the time requested. The interruption caused by the interval timer in such a situation is late. Second, implementing the time-of-day clock and the clock comparator in the same doubleword format eliminates having to convert doubleword time-of-day clock to single-word interval timer units.

The CPU timer provides a means of causing an external interruption when an interval of time specified by a program has elapsed. The CPU timer is implemented as a binary counter with a format identical to that of the time-of-day clock; however, bit 0 of the CPU timer is considered to be a sign. The CPU timer has a maximum time period half as large as that of the time-of-day clock and the same resolution of one microsecond. When both the CPU timer and the time-of-day clock are running, the stepping rates of the two are synchronized so that they are stepped at exactly the same rate.

The CPU timer is set to zero at initial program reset and the SET CPU TIMER privileged instruction is provided to place an interval of time in the CPU timer. The STORE CPU TIMER privileged instruction can be used to obtain the current CPU timer value. The CPU timer decrements every microsecond. If the external interruption summary mask bit in the current PSW and the CPU timer subclass mask bit in control register 0 are on, an external interruption occurs whenever the CPU timer value is negative (not just when the timer goes from positive to negative), indicating that the time interval has elapsed. The CPU timer decrements when the CPU is executing instructions (including instruction retry operations) and while the CPU is in the wait state. It is not decremented when the system is in the stopped state.

While providing essentially the same function as the interval timer at location 80, the CPU timer provides advantages over the interval timer as follows: Task processing intervals of less than 3.3 milliseconds are accurately measured because of the one microsecond resolution of the CPU timer. A pending CPU timer interruption is reset when a SET CPU TIMER instruction is issued to set a positive value in the CPU timer, eliminating the need to take an interruption in order to reset the CPU timer, as is required for the interval timer.

In addition, the amount of timing facilities processing required during a task switch can be reduced because the format of the time-of-day clock and the CPU timer are the same. Conversion of doubleword time-of-day clock values to single-word interval timer values is eliminated, and timer queues can be structured in such a way that little of the processing currently required during a task switch, when the interval timer is used, is necessary.

RELIABILITY, AVAILABILITY, AND SERVICEABILITY FEATURES

The following hardware RAS features implemented in the Model 168 are functionally identical to those provided in the Model 165:

- Automatic retry of most failing CPU operations by hardware (a few instructions are retried on a Model 165 that are not retried on a Model 168)

- ECC checking on processor storage to correct all single-bit and detect all double-bit errors. However, in a Model 168, machine check interruptions after ECC corrections are disabled during a system reset (that is, nonrecording mode is in effect). If machine check interruptions are to occur after ECC corrections, the DIAGNOSE instruction must be issued to enable full recording mode (and the recovery mask bit must be turned on). In the Model 165 (and 165 II), a system reset enables the CPU for machine check interruptions after ECC corrections.

- I/O operation retry facilities, including the storing of channel retry data during an I/O interruption that results from an error, and channel/control unit command retry procedures to correct certain failing I/O operations

Implementation of machine check interruption facilities is expanded in the Model 168 to provide more definitive logout information when a machine check interruption is taken, and new buffer row deletion and translation lookaside buffer deletion functions are implemented. Machine check interruption facilities are the same in Models 168 and 165 except for the following (which also apply to a Model 165 II except for the warning interruption):

- The instruction processing damage subclass of machine check interruption, not implemented in the Model 165, is implemented in the Model 168. Instruction processing damage is indicated in the machine check code (shown in Figure 20.10.4) when a CPU error occurs that is not retryable or that was unsuccessfully retried, unless an LPSW instruction or an interruption was in process at the time of the failure or the failure was a hang detect. In these cases, system damage is indicated. In the Model 165, system damage is indicated for all CPU and storage errors that cannot be retried or that are unsuccessfully retried.

  Implementation of the instruction processing damage subclass in the Model 168 is designed to identify errors that can be associated with a specific task so that only that task need be abnormally terminated. Code is included in the Model 165 MCH routine that attempts, when a system damage error is indicated, to distinguish system damage from damage that can be associated with a task. This code is not required for the Model 168.

- Whenever a machine check interruption is taken to record information about a correctable or an uncorrectable processor storage error, the failing processor storage address is placed in locations 248-251. The machine check code indicates the type of processor storage error and whether the stored failing storage address is valid.

- In the Model 168, each block in the high-speed buffer has a delete bit associated with it in the address array for the buffer, as in the Model 165. This bit can be turned on using a DIAGNOSE instruction. However, in the Model 168 each row within the buffer also has a row delete trigger associated with it. (There are four rows in the 8K buffer and eight rows in the 16K buffer, as shown later in Figure 20.15.2.) Whenever certain buffer errors occur and the Model 168 CPU is enabled for automatic buffer row deletions, hardware determines the buffer row in which the error occurred. The row delete trigger is turned on for that row. This indicates that the buffer row is disabled and that the CPU can no longer fetch data from or store data in the deleted buffer row. When the CPU is enabled for degradation interruptions, the machine check code stored during the interruption that occurs after a buffer row is deleted indicates a degradation condition.

  The mode bit implemented in the Model 165 that can be set by a DIAGNOSE instruction to cause the entire buffer to be disabled when a machine check occurs is not implemented in the Model 168. The buffer bypass mode bit in the Model 165 that causes the entire buffer to be bypassed when it is turned on via a DIAGNOSE instruction is implemented in the Model 168. The Model 168 also contains a mode bit that can be set using a DIAGNOSE instruction to cause the buffer row deletion mechanism to be disabled. This selective buffer deletion facility allows only one-quarter of an 8K buffer or one-eighth of a 16K buffer to be automatically disabled by hardware at the time certain buffer errors occur and avoids total buffer disabling after an error.

Fixed Logout Area Locations 232–239

Bit	0	1	2	3	4	5	6	7	8	9-14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32-45	46	47	48 – 63
	0 – 8 Machine Check Types											16 – 18 Storage Error				20 – 31, 46, 47 Validity Bits															48 – 63 CPU Extended Log Length
	SD	PD	SR	UNUSED	CD	ED	UNUSED	DG	W	UNUSED	BACKED UP	SE	SC	KE	UNUSED						UNUSED	UNUSED						UNUSED			Zero if no logout or 1416 bytes

Bit	Interrupt Type
0	SD — System Damage
1	PD — Instruction Processing Damage
2	SR — System Recovery
4	CD — Timing Facilities Damage
5	ED — External Damage
7	DG — Degradation
8	W | — Warning

Bit	Error
15	Delayed Interruption
16	Storage Error Uncorrected
17	Storage Error Corrected
18	Protection Key Error

Bit	Valid Fixed Area Data
20-23	Machine Check Old PSW (48-55) 20 AMWP 21 Masks and Protect Key 22 Program Mask and Condition Code 23 Instruction Address
24	Failing Storage Address
27	Floating Point Registers (352-383)
28	General Registers (384-447)
29	Control Registers (448-511)
30	CPU Extended Logout
31	Storage (Validity of storage being processed by instructions when interruption occurred.)
46	CPU Timer Value
47	Clock Comparator Value

Figure 20.10.4. Model 168 machine check code

- A translation lookaside buffer (TLB) deletion function is implemented and is discussed in Section 30:10.

- The time-of-day clock damage interruption, maskable by the external mask bit and PSW bit 13, is expanded to include clock comparator and CPU timer errors. Its name is changed to "Timing Facilities Damage". When a STORE CLOCK COMPARATOR or a STORE CPU TIMER instruction is issued and the addressed timing facility has an error or when the CPU timer or the clock comparator develops an error, a timing facilities damage interruption occurs if the timing facilities damage mask bit is a one.

- Whenever a machine check interruption occurs in a Model 168, the general and floating-point registers are validated, the current value of the CPU timer is stored in locations 216 to 223, and the current value of the clock comparator is stored in locations 224 to 231 during the interruption. Bits 46 and 47 of the machine check code, shown in Figure 20.10.4, are used to indicate whether the time values were stored correctly.

- The size of the CPU extended logout area in the Model 168 is 1416 bytes instead of 992 bytes as in the Model 165 in order to log additional status information when a machine check interruption occurs.

- Machine check code bits 22, 23, and 31 are set to zero only when an instruction processing damage or system damage type of machine check interruption occurs.

- A warning machine check interruption is implemented in Model 168 systems with the optional Power Warning feature installed. This field-installable feature can be used in Model 168 systems that have O.E.M. uninterruptable power supplies (UPS). A UPS is designed to protect a system from power line disturbance by providing auxiliary power for a specified interval of time during a power reduction or outage. A system can be fully or partially protected.

  Full protection involves supplying a UPS for all system components. This support provides continuous system operation for a specified interval of time during a power line disturbance. Partial protection involves supplying a UPS for a critical subset of system components, namely, the 3168 Processing Unit, 3066 Model 2 System Console, 3067 Model 2 Power and Coolant Distribution Unit, all standalone channels, and all the control units attached to one standalone channel. The Power Warning feature can be used with partially and fully protected Model 168 systems.

  A UPS for a Model 168 must generate a power warning signal when an undervoltage condition of 18% (±2%) is detected. A Model 168 CPU with the Power Warning feature recognizes this signal. If bit 13 in the current PSW and the warning submask (bit 7 in control register 14) are both on, a warning repressible machine check interruption occurs. Bit 8 in the stored machine check interruption code will be on to indicate a warning condition. The machine check handler (MCH) routine is given CPU control to process the interruption. If either mask bit is off, the warning interruption remains pending.

  The Power Warning feature is designed to enable a Model 168 system to terminate operations in an orderly manner when a power line disturbance or power shutdown occurs. When a warning interruption occurs, a determination can be made as to whether the power line disturbance is transient. Operation of a fully protected system need not be terminated for a transient disturbance of a short enough duration. If system termination is required, a complete processor storage dump can be taken first. This enables processor storage to be restored when a system restart is performed at a later time.

Model 168 recovery management routines (machine check and channel check handlers) that operate in BC mode are included in OS MFT and OS MVT as of Release 21.6. They provide recovery functions similar to those provided for the Model 165 and support of new Model 168 machine check facilities, except for MFT, which does not support the Power Warning feature.

An instruction processing damage interruption is recognized in the Model 168, and recovery management support (RMS) attempts to identify the affected task and abnormally terminate it. When a system damage error occurs, Model 168 operations are terminated without an attempt to refresh damaged control program areas. Model 168 RMS also recognizes a degradation interruption that indicates buffer row or TLB deletion by the hardware, and the operator is notified of this hardware action.

These recovery routines are also included in OS/VS1 and OS/VS2 and are modified to operate correctly when the Model 168 is operating in EC and dynamic address translation modes. A discussion of how these recovery routines differ from those provided for BC mode operations is contained in each optional programming systems supplement.

The same remote error analysis facility is provided for the Model 168 (Model 1) as for the Model 165. That is, optionally the 2955 Remote Analysis Unit can be attached to a channel in the Model 168 configuration. The 2955 can be connected to the RETAIN/370 network in Raleigh, North Carolina via a communication line. Using an OLT that runs under OLTEP and OLTSEP control, SYS1.LOGREC data can be sent via the 2955 to Raleigh for transmittal to the Large System Support Group in Poughkeepsie for problem analysis.

20:15 STORAGE

PROCESSOR (MAIN) STORAGE

Like the Model 165, the Model 168 has a two-level storage system in which large high-speed processor storage backs up small, higher-speed buffer storage. A maximum of 8192K of processor storage can be installed in a Model 168. The Model 165 can have a maximum of 3072K.
| Processor storage is available for the Model 168 Model 1 in 1024K increments as follows:

Model	Capacity (K=1024 bytes)
J	1024K
K	2048K
KJ	3072K
L	4096K
LJ	5120K
LK	6144K
LKJ	7168K
M	8192K

Processor storage in a Model 168 is four-way doubleword interleaved
| with a 2-microsecond cycle time, as it is in a Model 165. The processor storage installed in a Model 168 is divided into four logical storages, each of which can operate independently from the other three logical storages. Logical storages can be selected at 80 nanosecond intervals. The data path to and from processor storage is eight bytes wide. Consecutively addressed doublewords are spread across logical storages,

as shown in Figure 20.15.1, so that access to four doublewords can be overlapped. The processor storage control function provides the interface to the logical storages.

As in a Model 165, processor storage in a Model 168 can be accessed concurrently by any combination of one or more channels and the CPU for a total of four unique logical storage requests. When simultaneous requests for the same logical storage are received, the processor storage access priority control unit schedules the requests according to a priority scheme. This priority is the same in Models 168 and 165. That is, the channels have priority over the CPU and the priority among channels is definable at channel installation time. Table 20.15.1 summarizes Model 168 uniprocessor cycle and access times.

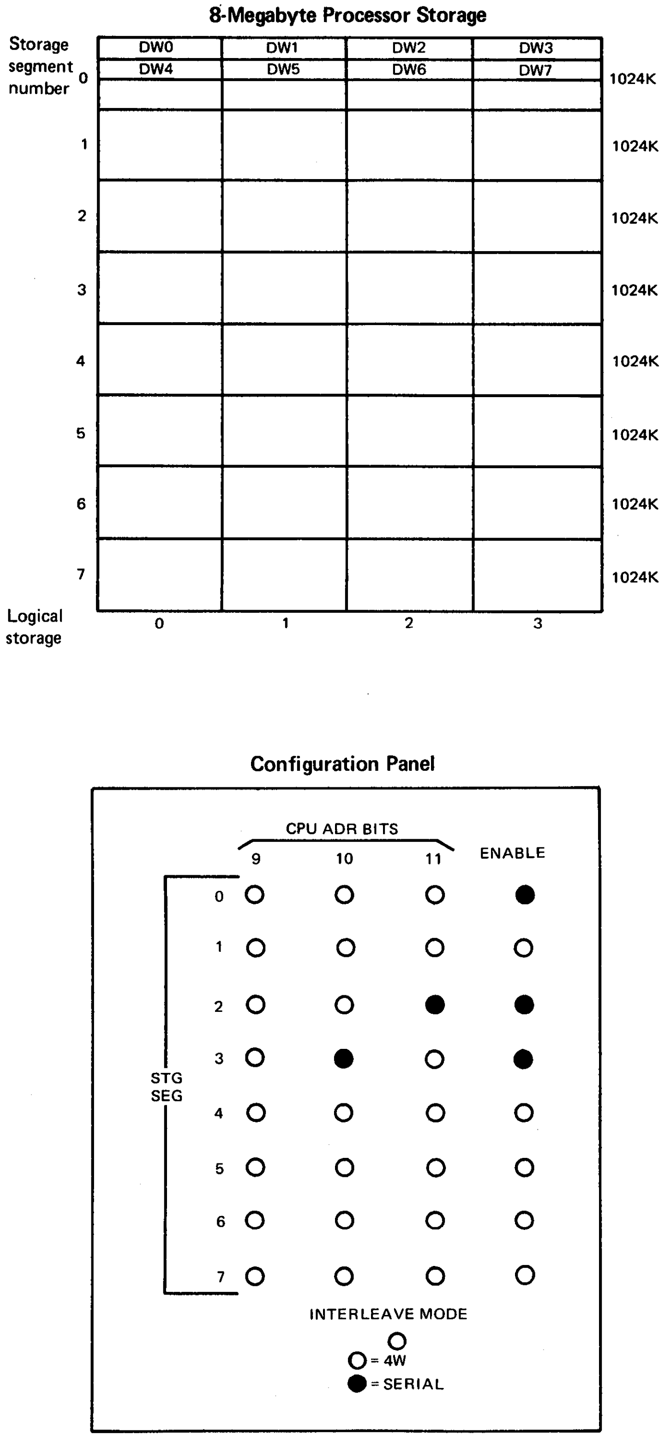
Table 20.15.1. Model 168 uniprocessor cycle and access times

Cycle or Access Time	Time in Nanoseconds
CPU cycle time	80
Local storage cycle time	80
Control storage cycle time	80
Processor (logical) storage read/write cycle time (for eight bytes on a doubleword boundary)	320
Processor (logical) storage cycle time for a partial write (fewer than 8 bytes)	640
Minimum time between successive selects to processor storage	80
Processor storage access time (from time of PSCF select to availability of data in the PSCF)	400
CPU fetch of 8 bytes from processor storage (from time of request acceptance to availability of data in a CPU register)	640
CPU fetch of eight bytes from buffer (from time of request acceptance to availability of data in a CPU register)	
Minimum time between successive buffer requests	80

Processor Storage Reconfiguration

As shown in Figure 20.15.1, the processor storage present in the Model 168 is divided into from one to eight elements of 1024K bytes each. Element numbers 0 to 7 are used. If an uncorrectable processor storage error occurs, the element containing the malfunctioning location(s) can be manually configured out of the system by the operator.

The configuration panel on the 3066 Model 2 System Console is used to enable storage elements, assign a one-megabyte range of addresses to each enabled element, and establish four-way interleaving or serial operations. The configuration panel is also shown in Figure 20.15.1. The operator selects a configuration by inserting pins in the appropriate hubs. The storage configuration indicated by the panel is made effective during a system reset. If necessary, the storage configuration that is actually enabled can be displayed on the indicator viewer (configuration registers in Image C0).



STG SEG	9	10	11	ENABLE
0	○	○	○	●
1	○	○	○	○
2	○	○	●	●
3	○	●	○	●
4	○	○	○	○
5	○	○	○	○
6	○	○	○	○
7	○	○	○	○

Figure 20.15.1. Model 168 processor storage organization and configuration panel

The absence of a pin in the interleave mode hub selects four-way interleaving. When a pin is inserted in this hub, serial (noninterleaved) operations are selected. The presence of a pin in an enable hub indicates the associated storage element is to be included in the active storage configuration. The three CPU address bit hubs for an element are used to indicate the range of processor storage addresses that are to be assigned to the element. Shown below are the pin combinations that are required to select the various ranges of addresses. A zero in an address bit hub column indicates the absence of a pin. A one indicates the presence of a pin.

Address Range	Address Bit Hub		
	9	10	11
0-1024K	0	0	0
1024K-2048K	0	0	1
2048K-3072K	0	1	0
3072K-4096K	0	1	1
4096K-5120K	1	0	0
5120K-6144K	1	0	1
6144K-7168K	1	1	0
7168K-8192K	1	1	1

The storage configuration selected by the control panel shown in Figure 20.15.1 is the following:

- Elements 0, 2, and 3 are enabled and all other elements are disabled.

- Element 0 is assigned addresses 0 to 1024K, element 2 is assigned addresses 1024K to 2048K, and element 3 is assigned addresses 2048K to 3072K.

- Four-way interleaving is enabled.

Storage ripple functions are provided in the Model 168 for read-only control storage, writable control storage, local storage, and processor storage, as for the Model 165. The inline ripple facility of the Model 165 is not implemented in the Model 168.

HIGH-SPEED BUFFER STORAGE

As in the Model 165, an 8K buffer is standard in the Model 168 (Model 1) and installation of the optional Buffer Expansion feature permits inclusion of an additional 8K of buffer storage. Buffer storage provides high-speed data access for CPU fetches. In a Model 168, as in a Model 165, the CPU can obtain eight bytes from the buffer in 160 nanoseconds (two CPU cycles) and a request can be initiated every cycle. This is the time between request acceptance and availability of the data in a CPU register. If the buffer does not contain the data required, the data must be obtained from processor storage.

Use of the high-speed buffer in Models 168 and 165 is almost identical. (This description of the buffer in Model 1 of the Model 168 also applies to the Model 165 II.) When a data fetch request is made by the CPU, a determination is made of whether the requested data is in the high-speed buffer by the interrogation of the address array of the buffer's contents. If the data requested is present in the buffer, it is sent directly to the CPU without a processor storage reference. If the requested data is not currently in the buffer, a processor storage fetch is made and the data obtained is sent to the CPU. The data is also assigned a buffer location and stored in the buffer. When data is stored by the CPU, both the buffer and processor storage are updated if the contents of the processor storage location being altered are currently being maintained in the buffer.

The channels never access the buffer directly. They read into and write from processor storage using a eight-byte-wide path between the CPU and processor storage that bypasses the buffer. When a channel stores data in processor storage, the address array is inspected. If the data from the affected processor storage address is being maintained in the buffer, appropriate bits are set in the address array to indicate that this buffer data is no longer valid. In a Model 165, the buffer is updated instead of invalidated when a channel stores data in a processor storage location whose contents are currently in the buffer.

As in a Model 165, the entire buffer in a Model 168 can be disabled manually by a system console switch. When the buffer is disabled, all CPU fetches are made directly to processor storage and effective system execution speed is reduced. Selective buffer disabling by row performed by hardware, as described previously, is also provided for the buffer in the Model 168.

The 8K and 16K buffers are shown in Figure 20.15.2 together with their address arrays. The 8K buffer is organized in the same way in Models 168 and 165. The 8K buffer contains 64 columns of 128 bytes each. Every buffer column is subdivided into four blocks. A block is 32 bytes and can contain 32 consecutive bytes from processor storage that are on a 32-byte boundary. The 8K buffer can contain a maximum of 256 different blocks of processor storage data (four blocks per column times 64 columns). A valid trigger is associated with each buffer block and is set to indicate whether the block contains valid data. All valid triggers are set off during an initial program reset. There are four rows in the 8K buffer. The first row consists of block 0 of each column (64 blocks). The last row consists of block 3 of each column.

The organization of the 16K buffer in Models 168 and 165 is slightly different. In the Model 168, the 16K buffer still contains 64 columns but each column has eight blocks instead of four. In a Model 165, the 16K buffer has 128 columns of four blocks each. The approach taken in the Model 168 enables bits 21 to 31 of the storage address in an instruction to be used to address the index array for the buffer whether the storage address is virtual or real. This enables interrogation of the index array to be performed simultaneously with interrogation of the translation lookaside buffer, which is part of the Dynamic Address Translation Facility. (See Section 30:10 for more details.) There are eight rows in the 16K buffer. The first row consists of block 0 of each column (64 blocks). The last row consists of block 7 of each column.

Processor storage is logically divided into the same number of columns as buffer storage, which is always 64 in the Model 168. While there are four or eight blocks in a buffer column, depending on buffer size, the number of blocks in a processor storage column varies with the size of processor storage. When buffer storage is assigned, bits 21-26 of the processor storage address determine which one of the 64 columns in buffer storage is to be used. The organization of 2048K bytes of processor storage is shown in Figure 20.15.2. Any of the 1024 blocks in a given processor storage column can be placed in any one of the four (8K buffer) or eight (16K buffer) blocks in a corresponding buffer column.

Figure 20.15.2 also shows the organization of the address array for the 8K and the 16K buffer. The address array contains the processor storage addresses of the data that is currently in the buffer. A least-recently-used algorithm, similar to that used in the Model 165, is implemented in the Model 168 to determine which block within a buffer column is to be assigned when data is placed in the buffer.

Buffer and processor storage components and controls in the Model 168 are shown in Figure 20.15.3.

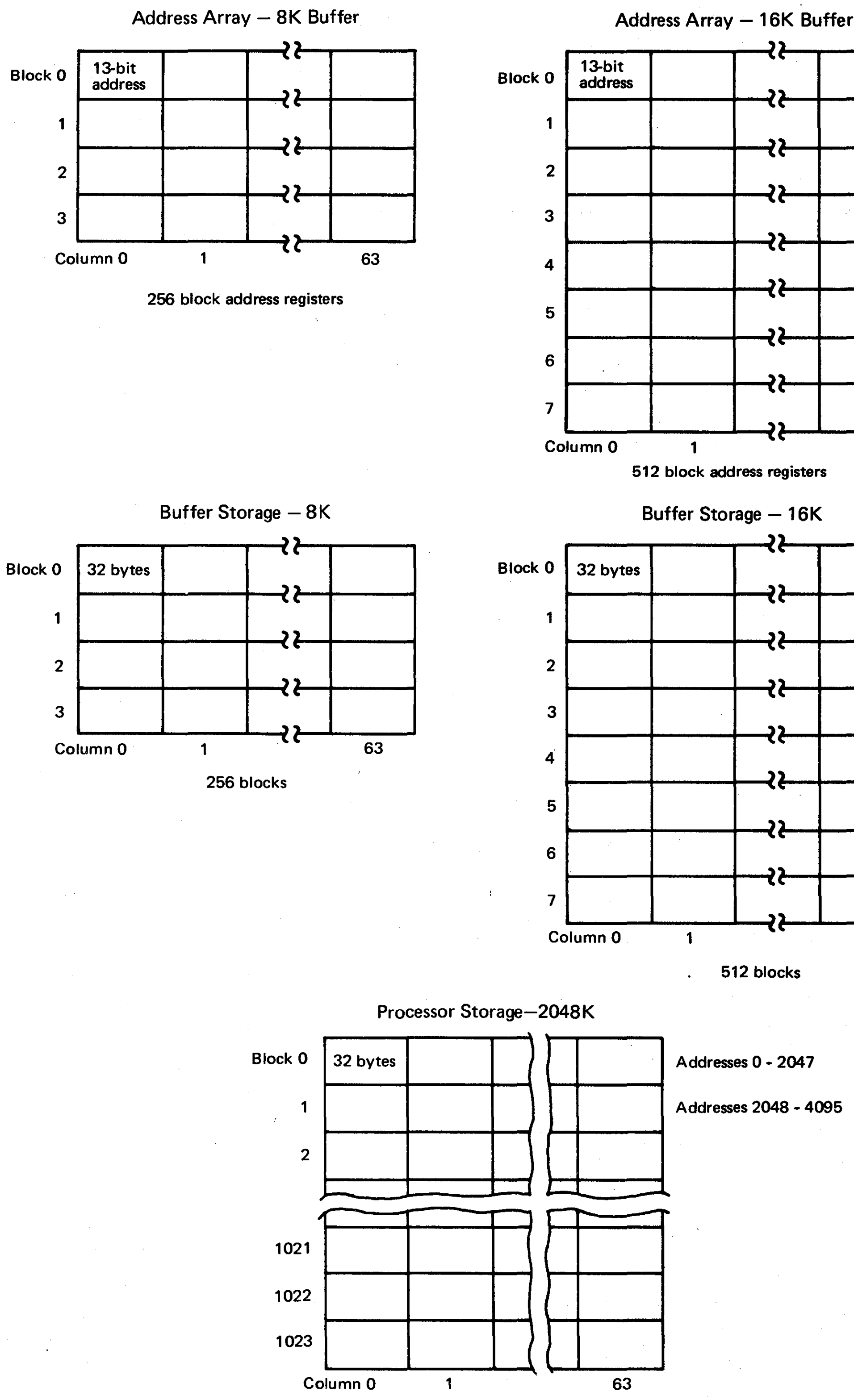


Figure 20.15.2. 8K and 16K buffer organization

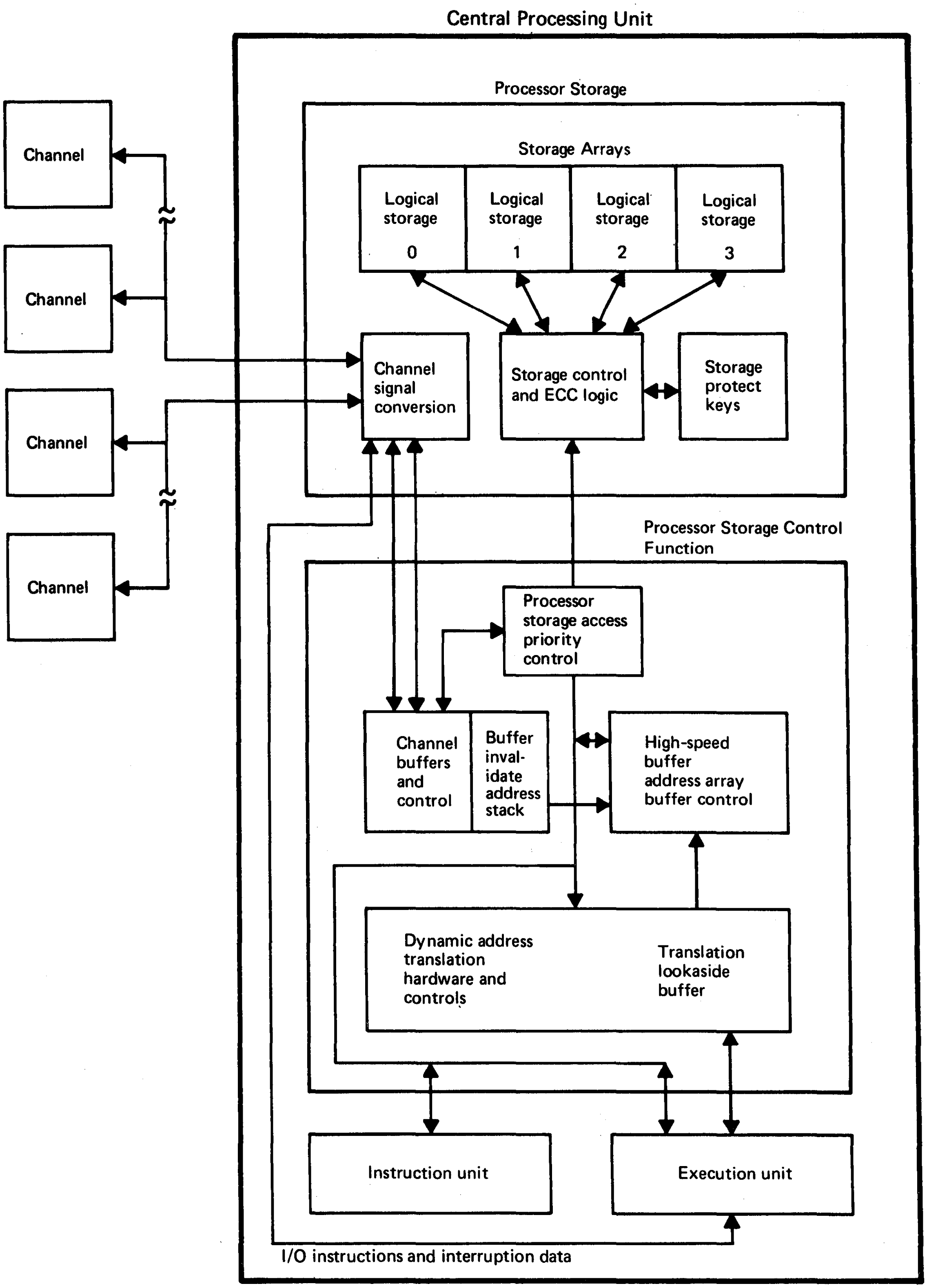

Central Processing Unit
Processor Storage
Storage Arrays
Logical storage 0
Logical storage 1
Logical storage 2
Logical storage 3
Channel
Channel
Channel
Channel
Channel signal conversion
Storage control and ECC logic
Storage protect keys
Processor Storage Control Function
Processor storage access priority control
Channel buffers and control
Buffer invalidate address stack
High-speed buffer address array buffer control
Dynamic address translation hardware and controls
Translation lookaside buffer
Instruction unit
Execution unit
I/O instructions and interruption data

Figure 20.15.3. Model 168 components and controls

20:20 CHANNELS

The number and types of channels that can be attached to Models 165 and 168 are the same. The capability of attaching up to seven standalone channels to the Model 168 is standard. Any combination of one or two 2870 Multiplexer, up to six 2860 Selector, and up to six 2880 Block Multiplexer Channels can be attached to a Model 168, up to the limit of seven channels. Installation of the optional Extended Channels feature permits attachment of a maximum of twelve channels. Any combination of one (with address 0) or two (with an address from 1 to 6) 2870s, six 2860s (with addresses 1 through 6), and eleven 2880s (with addresses 1 through 11), up to the limit of twelve, can be installed. A maximum of seven channel frames (for a maximum of twelve channels) can be attached to the Model 168.

As for a Model 165 channel configuration, the addresses and priorities of the channels present in a Model 168 configuration are established at channel installation time as indicated by the user, within the restraints specified for the Model 168. The channel buffering scheme implemented in the storage control unit is the same for Models 168 and 165.

The 2870, 2860, and 2880 channels that attach to the Model 168 are functionally and physically identical to those that attach to a Model 165. The same attachment feature that must be installed on a 2870 or a 2860 channel in order to attach the channel to a Model 165 must be installed on 2870 and 2860 channels that are to be attached to a Model 168.

The 2880 has one shared subchannel and 56 nonshared subchannels. The shared subchannel always has 200 device addresses associated with it plus one additional address for each nonshared subchannel not plugged during installation. The Extended Unit Control Words feature can be installed on a 2880 attached to a Model 168 to increase the number of nonshared subchannels in the 2880 from 56 to 256. This feature is mutually exclusive with the Two-Byte Interface feature for 2880 channels.

While the data rates of channels that attach to the Model 168 are the same as for the Model 165, the maximum aggregate data rate that a Model 168 can sustain with minimal overrun exposure is significantly higher than that of the Model 165. The Model 168 can also have more high-speed I/O devices, such as the 2305, operating concurrently. The increased data rate is made possible by the use of a channel dual I/O bus to transfer data between the channels and the storage control unit so that the faster cycle time of Model 168 processor storage can be utilized to advantage.

The channel dual I/O bus in the Model 168 consists of bus A and bus B. Each bus provides a path between from one to six channels and a register in the storage control unit. A channel is connected to one bus or the other (not to both). Data can be transferred simultaneously on the two buses. This facility is used for input operations to transfer simultaneously data from two different channels to registers in the storage control unit.

A Model 168 without the Extended Channels feature can have three channels attached to bus A and four channels attached to bus B. When the Extended Channels feature is installed, a maximum of six channels can be attached to each bus. The channel priority assigned to a channel determines the bus to which it must be attached. A channel assigned priority 1, 2, 3, 9, A(10), or B(11) must be attached to the A bus. A channel assigned priority 4, 5, 6, 7, C(12), or D(13) must be attached to the B bus. Channels within the same channel frame must be attached to the same bus. Channels with the highest speed devices attached should be positioned closest to the Model 168 processor on the bus to which they are attached. Channel priority is established by plugging jumpers on matrix cards in the processor storage control function.

Channel priorities 1 to D refer to the high to low priorities the buffers for each channel have for accessing processor storage. Channel priorities also determine the relative priority channels attached to the same I/O bus have for using the bus to access their associated buffers.

Thus, for the A bus high to low channel priorities for using the bus to access channel buffers are 1, 2, 3, 9, A, B. For the B bus, high to low channel priorities are 4, 5, 6, 7, C, D. In effect, the two channels with priority 1 and 4 have the same priority for accessing their respective channel buffers but the channel with priority 1 has the highest priority for accessing processor storage. Similarly, the channels with priorities 2 and 5 have the same relative buffer access priority, etc.

A 2780 channel without a selector subchannel or with one or two selector subchannels should be given as high a channel priority as possible. A 2780 channel with more than two selector subchannels should be assigned priority position 1, 2, 3, or 4.

An aggregate data rate of 8.5 MB/sec can be sustained on each bus, which provides a total maximum aggregate data rate of approximately 17 MB/sec for the system. As a general rule, devices with the highest data rates should be attached to the highest priority channels. Table 20.20.1 indicates the channel priorities that the highest speed System/370 I/O devices require. That is, each I/O device in the table can be assigned only those priorities indicated in its column. Each column also indicates the maximum number of channels to which the device can be attached (four for the 2305 Model 1, six for the 2305 Model 2, etc.).

Table 20.20.1. Permissible configurations and channel priorities for highest speed System/370 I/O devices in uniprocessor systems. An asterisk indicates the device attaches via the 2880 only.

	Device Type				
Channel Priority	2305 Model 1* 3 MB/sec	2305 Model 2* 1.5 MB/sec	3420 Model 8 1.25 MB/sec	3330-series* 3340* .8 MB/sec	3420 Model 6 .8 MB/sec
1	X	X	X	X	X
2	X	X	X	X	X
3		X	X	X	X
4	X	X	X	X	X
5	X	X	X	X	X
6		X	X	X	X
7			X	X	X
9			X	X	X
A			X*	X	X
B				X	X
C					X
D					X

Permissible I/O device configurations are also shown by table 20.20.1, which in turn indicates the I/O device configurations that can operate concurrently. In general, any other device type with similar characteristics and the same or a slower data rate than the listed device can also be assigned the indicated channel priority. Negligible or no overrun exposure exists in a Model 168 system when the guidelines indicated in Table 20.20.1 are followed.

The presence of the channel dual I/O bus in the Model 168 permits greater flexibility in the physical layout of Model 168 components since the channel frames are attached to two separate cable sets instead of only one, as for a Model 165. Greater flexibility in the cable lengths between channel frames attached to the same I/O bus is also provided by the Model 168.

20:25 SYSTEM CONSOLE

The 3066 Model 2 System Console for the Model 168 has the same features as the 3066 Model 1 System Console for the Model 165: a cathode ray tube and keyboard, a microfiche indicator viewer, a microfiche document viewer, a processor storage configuration panel, a system activity monitor, and a device for loading microcode and diagnostics. In addition, the store status function is implemented. (The store status function is implemented in a Model 165 II as well.)

The operator can cause the contents of the following to be placed in processor storage by pressing the store status button on the control panel:

CPU timer - locations 216-223
Clock comparator - locations 224-231
Current PSW - locations 256-263
Floating-point registers - locations 352-383
General registers - locations 384-447
Control registers - locations 448-511

In addition to the store status button, the control panel on the 3066 Model 2 has system clear and cooling reset alarm pushbuttons, and a switch associated with the dynamic address translation feature.

20:30 STANDARD AND OPTIONAL SYSTEM FEATURES

STANDARD FEATURES

Standard features for the System/370 Model 168 (Models 1 and 3) are:

- BC and EC mode of operation
- Instruction set that includes binary, decimal, floating-point, and extended precision floating-point arithmetic, and System/370 instructions. Standard System/370 instructions for the Model 168 are:

*CLEAR I/O
COMPARE AND SWAP
COMPARE DOUBLE AND SWAP
COMPARE LOGICAL CHARACTERS UNDER MASK
COMPARE LOGICAL LONG
INSERT CHARACTERS UNDER MASK
*INSERT PSW KEY
*LOAD CONTROL
*LOAD REAL ADDRESS
MONITOR CALL
MOVE LONG
*PURGE TLB
*RESET REFERENCE BIT
*SET CLOCK
*SET CLOCK COMPARATOR

*Privileged instruction

*SET CPU TIMER
*SET PSW KEY FROM ADDRESS
SHIFT AND ROUND DECIMAL
*START I/O FAST RELEASE
*STORE CHANNEL ID
STORE CHARACTERS UNDER MASK
STORE CLOCK
*STORE CLOCK COMPARATOR
*STORE CONTROL
*STORE CPU ID
*STORE CPU TIMER
*STORE THEN AND SYSTEM MASK
*STORE THEN OR SYSTEM MASK

*Privileged instruction

- Dynamic Address Translation
- Reference and Change Recording
- Instruction retry
- Interval timer (3.3 ms resolution)
- Time-of-day clock, clock comparator, and CPU timer
- Monitoring feature
- Program Event Recording
- Program interruption for SSM instruction
- Expanded machine check interruption class
- ECC on processor storage
- Byte-oriented operands
- Store and fetch protection
- High-speed buffer storage - 8K bytes (Model 1), 32K bytes (Model 3)
- Attachment for up to seven channels
- Channel dual I/O bus
- Channel retry data in extended channel logout area
- Writable and read-only control storage
- Store status function
- Direct Control
- Service processor (Model 3 only)

OPTIONAL FEATURES

Optional features for the System/370 Model 168 (Models 1 and 3), which can be field installed unless otherwise indicated, are:

- 3066 Model 2 System Console (required in all configurations)
- High-Speed Multiply**
- Buffer Expansion for inclusion of a 16K buffer (Model 1 only)
- 7070/7074 Compatibility**
- 7080 Compatibility**
- 709/7090/7094/7094II Compatibility**
- 2870 Byte Multiplexer Channels, 2860 Selector Channels, and 2880 Block Multiplexer Channels
- Channel Indirect Data Addressing for 2870, 2860, and 2880 channels (required when OS/VS1, OS/VS2, or VM/370 is used)
- Extended Channels (for up to twelve channels)
- Channel-to-Channel Adapter on 2860 channels
- Extended Unit Control Words on 2880 channels (mutually exclusive with the Two-Byte Interface feature)
- Integrated Storage Controls
- Two-Channel Switch and/or Staging Adapter for Integrated Storage Controls
- Power Warning
- 2955 Remote Analysis Unit (Model 1 only)
- Multiprocessing

**Not recommended for field installation

Note: Compatibility features are mutually exclusive

SECTION 30: VIRTUAL STORAGE AND DYNAMIC ADDRESS TRANSLATION

The first subsection, 30:05, discusses the needs that virtual storage and dynamic address translation in System/370 are designed to address. No previous understanding of these facilities is assumed. In this discussion, an address space is defined as a consecutive set of addresses that can be used in programs to reference data and instructions. System operation in IBM-supplied virtual storage environments is explained conceptually, without use of all the terminology new to such an environment.

The general advantages of IBM-supplied virtual storage operating systems are presented also. Included in this subsection are those that apply to OS/VS1 and OS/VS2. Additional advantages of virtual storage that are specific to a particular IBM-supplied operating system are discussed in the optional supplement for that operating system.

The last portion of subsection 30:05 defines the terminology associated with virtual storage and dynamic address translation hardware. The terminology included is that common to the four IBM-supplied programming systems that support a virtual storage environment for System/370. However, specific references to DOS/VS are not made where a difference between DOS and OS exists, since DOS/VS does not support the Model 168. Terms unique to a particular programming system are defined in the optional supplement that describes that programming system.

Subsection 30:10 describes in detail the implementation and operation of dynamic address translation and channel indirect data addressing hardware in the Model 168 (Models 1 and 3). Other hardware items associated with dynamic address translation, such as reference and change recording, are discussed as well.

The last subsection, 30:15, discusses the new factors that affect system performance in a virtual storage environment. The information presented is related to efficient installation and utilization of an IBM-supplied virtual storage operating system.

The two optional programming systems supplements (Sections 90 and 100) for the virtual storage operating systems for the Model 168 (OS/VS1 and OS/VS2 SVS assume knowledge of the entire contents of Section 30. The optional supplement for VM/370 (Section 110) assumes knowledge of subsections 30:05 and 30:10 only, since performance in a virtual machine environment is discussed in the VM/370 supplement. This entire section applies to the Model 165 II as well as to the Model 168, except where differences are noted.

30:05 VIRTUAL STORAGE CONCEPTS, ADVANTAGES, AND TERMINOLOGY

THE NEED FOR LARGER ADDRESS SPACE

The past and present rapid growth in the number and types of data processing applications being installed has led to an increasing demand for more freedom to design applications without being concerned about, or functionally constrained by, the physical characteristics of a particular computer system--system architecture, I/O device types, and processor storage size. As program design and implementation become easier, they can enable more rapid installation of applications, so that the benefits of data processing can be achieved sooner.

The design of System/360 and OS MFT and MVT allowed programmers to be less concerned than before about specific CPU architecture and I/O device types when designing and implementing applications by (1) providing a compatible set of CPU models ranging in size from small to large scale, (2) providing a variety of high-level languages with greatly expanded capabilities, including a new language (PL/I), (3) providing comprehensive data management functions, including support of I/O device independence where data organization and the physical characteristics of devices permitted, and (4) supporting dynamic allocation of system resources (channels, I/O devices, direct access space, and processor storage).

While System/360 and OS represented major steps toward giving programmers a larger measure of system configuration independence, constraints that resulted from the necessity to design applications to fit within the amount of processor storage available still existed. In addition, although System/360 models provided more and less costly processor storage than was previously available, increasingly larger amounts of processor storage began to be required as the use of high-level languages increased, the usage and level of multiprogramming increased, the functions supported by operating system control programs expanded, and applications that require relatively larger amounts of processor storage (such as teleprocessing and data base) were designed and installed more frequently.

The requirement for more processor storage is still growing. The new applications being developed and installed tend to have larger and larger storage design points in order to provide the functions desired. More processor storage is also required for I/O buffer areas to achieve maximum capacity and performance for sequential operations using new System/370 direct access devices with significantly larger track capacities. Larger blocking of tape records, which requires larger I/O buffers, also results in increased tape reel capacity and decreased tape processing time. As a result, System/370 models provide significantly more processor storage than their predecessor System/360 models and offer it for a lower cost.

The availability of more processor storage, however, has not relieved all the constraints associated with processor storage. Applications still must be tailored to the amount of processor storage actually available in a given system even though storage design points (partition and region sizes) can be larger than they were previously.

Consider the following situations that can occur in installations:

1. An application is designed to operate in a 50K processor storage area that is adequate to handle current processing needs and that provides room for some expansion. Some time after the application is installed, however, maintenance changes and the addition of new functions cause one of the programs in the application to require 51K and another to require 52K. Installation of the next processor storage increment cannot be justified on the basis of these two programs, so time must be spent restructuring and retesting the programs to fit within 50K.

2. An existing application has programs with a planned overlay structure. The volume of transactions processed by these programs has doubled and better performance is now required. Additional processor storage is installed. However, the overlay programs cannot automatically use the additional storage. Therefore, reworking of the overlay programs is required to take them out of planned overlay structure and, thereby, achieve the better performance desired.

3. A low-volume, terminal-oriented, simple inquiry program that will operate for three hours a day is to be installed. If the program is written without any type of overlay structure, it will require 60K of processor storage to handle all the various types of inquiries. However, because of a low inquiry rate, only 8K to 12K of the total program will be active at any given time. In order to justify its operational cost, considerable additional program development time is spent designing the inquiry program to operate with a dynamic overlay structure so that only 12K of processor storage is required for its execution.

4. A multiprogramming installation has a daily workload consisting primarily of long-running jobs. There are also certain jobs that require a relatively small amount of time to execute. The times at which these jobs must be executed are unpredictable; however, when they are to be run, they have a high completion priority. While it is desirable to be able to initiate these high-priority jobs as soon as the request to execute them is received, this cannot be done because long-running jobs are usually in operation. Hence, a certain time of day is established for initiating high-priority jobs and the turnaround time for these jobs is considerably longer than is desired.

5. A series of new applications are to be installed that require additional computing speed and twice the amount of processor storage available in the existing system. The new application programs have been designed and are being tested on the currently installed system until the new one is delivered. However, because many of the new application programs have storage design points that are much larger than those of existing applications, testing has to be limited to those times when the required amount of processor storage can be made available. Although another smaller-scale model is also installed that has time available for program testing, it cannot be used because it does not have the amount of processor storage required by the new application programs. In addition, although the smaller-scale model now provides backup for the currently installed larger-scale model, the smaller-scale model cannot be used to back up the new system because of processor storage size limitations.

6. A large terminal-oriented application is to be operative during one entire shift. During times of peak activity, four times more processor storage is required than during low-activity periods. Peak activity is experienced about 20 percent of the time and low activity about 40 percent. The rest of the time, activity ranges from low to peak. Allocation of the peak activity processor storage requirement for the entire shift cannot be justified and a significantly smaller storage design point is chosen. As a result, a dynamic program structure must be used, certain desired functions are not included in the program, and response times during peak and near-peak activity periods are increased above that originally planned.

   In this installation, most of the batched jobs are processed during the second shift. However, there is also a need to operate the large terminal-oriented application for a few hours during second shift. This cannot be done because the system does not have the amount of processor storage required for concurrent operation of the batched jobs and the terminal program (which must have its storage design point amount allocated even though that amount of processor storage would not be required during second shift operations). The large amount of additional processor storage required to operate the terminal program for only a portion of the second shift cannot be justified.

7. An application program with a very large storage design point is executed only once a day as a batched job. A significant benefit would result from putting the program online to a few terminals during the morning hours. However, the program continues to be run as a batched job because it is very large and would be made larger by putting it online. The large amount of additional processor storage required to operate the program concurrently with the existing morning workload cannot be justified.

8. A terminal-based application has been installed on a full production basis for several months. During this period, the benefits accrued from the online application have encouraged the gradual addition of several more terminals, and peak activity is considerably higher than it was initially. Because growth has been gradual, much additional programming time (significantly more than is required to maintain batch-oriented applications) has to be spent periodically restructuring the terminal-based application program to handle the increasing volume of activity.

9. An online application is currently active during an entire shift and operates concurrently with batched jobs. It would be advantageous to install a second terminal-oriented application that would operate concurrently with the existing workload during the entire shift. However, the amount of processor storage that would have to be dedicated to each online application for the entire shift in order to handle its peak activity is very large, and times of peak activity for the two applications do not completely overlap. Because so much processor storage would be unused during a large portion of the shift if both online applications were always active, installation of the second online application is difficult to justify.

In the situations described, processor storage is a constraining factor in one way or another and the constraints highlighted can apply in some degree to all systems regardless of their scale (small, intermediate, large) or processor storage size. The fact that larger, less expensive processor storage is now available on System/370 models does not remove these constraints for two major reasons.

First, once a storage design point has been chosen for an application, whether the design point is relatively large or small, the application is dependent on that processor storage size for its operation. The application cannot execute in less than its design point storage amount, nor can it take advantage of additional available processor storage without being modified (unless it has been specifically structured to use additional storage as, for example, are most IBM-supplied language translators).

Second, although processor storage has become less costly, it still is a resource that should be used efficiently because of its importance in the total system operation. Thus, when storage design points are chosen, tradeoffs among processor storage cost, application function, and system performance are often made. Making applications fit within the storage design points selected becomes the responsibility of application designers and programmers. This situation is made more difficult by the fact that for many applications an optimum storage design point cannot be determined until the application is written and tested using expected transaction volumes.

The significance of processor storage restraints should be evaluated in light of the following trends evidenced by new types of applications: (1) the total amount of storage required to support their new facilities continues to grow larger, (2) the storage they actually require for operation during their execution is tending to become more variable, and (3) it is becoming as desirable to install many of these new applications on smaller-scale systems with relatively small maximum processor storage sizes and low volume requirements as it is to install them on larger-scale systems. Reduction of the constraining factors currently imposed by processor storage is, therefore, a necessary step in making new applications easier and less costly to install and available to a wider range of data processing installations.

Given the existing processor storage restraints on application design and development and the storage requirements that are becoming increasingly more characteristic of many of the new types of applications, it becomes advantageous to allow programmers to design and code applications for a larger address space than they currently have. That is, programmers should be able to use as much address space as an application requires so that special program structures and techniques are not required to fit the application into a given storage size. Programmers can then concentrate more on the application and less on the techniques of programming. In addition, the size of the address space provided should not be determined by processor storage size, as it is in OS MFT and MVT, so that the address space can be larger than the processor storage available.

A larger address space should be provided, therefore, by a means other than making processor storage as large as the address space desired. This requirement can be satisfied by providing programmers with an address space (called virtual storage) that is supported using online direct access storage and dynamic address translation hardware. This approach also offers the advantage of supporting a larger address space for a lower cost than if larger processor storage is used, since direct access storage continues to be significantly less expensive per bit than processor storage. In addition, dynamic address translation hardware offers functional capabilities that large processor storage alone cannot provide.

VIRTUAL STORAGE AND DYNAMIC ADDRESS TRANSLATION CONCEPTS

Virtual storage is an address space the maximum size of which is determined by the addressing scheme of the computing system that supports it rather than by the actual number of physical processor storage locations present in the computing system. In System/370, for example, which uses a 24-bit binary address, a virtual storage as large as 16,777,216 bytes can be supported. When virtual storage is implemented, the storage that can be directly accessed by the CPU, normally called processor or main storage, is referred to as real storage.

The concept of virtual storage is made possible by distinguishing between the names of data and instructions and their physical location. In a virtual storage environment, there is a distinction between address space and real storage space. Address space (virtual storage) is a set of identifiers or names (virtual storage addresses) that can be used in a program to refer to data and instructions. Real storage space is a set of physical storage locations in the computer system in which instructions and data can be placed for processing by the CPU. The number of addresses in the two spaces need not be the same, although both spaces begin with address zero and have consecutive addresses. The programmer refers to data and instructions by name (virtual storage address) without knowing their physical (real storage) location.

When virtual storage is not implemented, there is, in effect, no differentiation between address space and real storage space. The address space that can be used in programs is identical in size to the real storage space available and the address in an instruction represents both the name and the location of the information it references.

In a virtual storage environment, therefore, the address space available to programmers is that provided by the virtual storage size implemented by a given system--not the address space provided by the real storage available in the given system configuration. In OS/VS1 and OS/VS2, virtual storage rather than real storage is divided into consecutively addressed partitions or dynamically allocated regions for allocation to problem programs. The fact that storage addresses in executable programs are virtual rather than real does not affect the way in which the programmer handles addressing. In System/370, for example, an Assembler Language programmer assigns and loads base registers and manipulates virtual storage addresses in a program just as if they were real storage addresses.

Virtual storage is so named because it represents an "image of storage" rather than physical processor storage. Since virtual storage does not actually exist as a physical entity, the instructions and data to which its virtual storage addresses refer, which are the contents of virtual storage, must be contained in some physical location.

In OS/VS1 and OS/VS2 environments, the contents of virtual storage are divided into a portion that is always present in real storage, namely, part of the control program, and another portion that is not always present in real storage. The instructions and data that are not always present in real storage must be placed in locations from which they can be brought into real storage for processing by the CPU during system operation. This requirement is met by using direct access storage to contain this portion of the contents of virtual storage (see Figure 30.05.1). The amount of direct access storage required to support a given amount of virtual storage varies by operating system, depending on how direct access storage is organized and allocated.

In addition, a mechanism is required for associating the virtual storage addresses of instructions and data contained in direct access storage with their actual locations in real storage when the instructions and data are being processed by the CPU. This requirement is met by using dynamic address translation (DAT) hardware in the CPU to associate virtual storage addresses with appropriate real storage addresses.

With this design, a system can support an address space that is larger than the actual size of the real storage present in the system. This is accomplished by bringing instructions and data from direct access storage into real storage only when they are actually required by an executing program, and by returning altered instructions and data to direct access storage when the real storage they occupy is needed and they are no longer being used. At any given time, real storage contains only a portion of the total contents of virtual storage.

Such a design is made practical by the fact that the logical flow of processing within the majority of programs is such that the entire program need not be resident in real storage at all times during execution of the program. For example, initialization and termination routines are executed only once during the operation of a program. Any exception-handling procedure, such as an error routine, is required only if the exception condition occurs. A program that handles a variety of transaction types (whether batch or online oriented) need have resident at any given time only the transaction routine required to process the current transaction type. It is this property of programs that has enabled planned overlay and other dynamic program structures to be used successfully in nonvirtual storage environments when the amount of processor storage available was not large enough. As indicated previously, this variable storage requirement characteristic of programs tends to be even more pronounced in new types of applications and in online environments in which processing is event-driven.

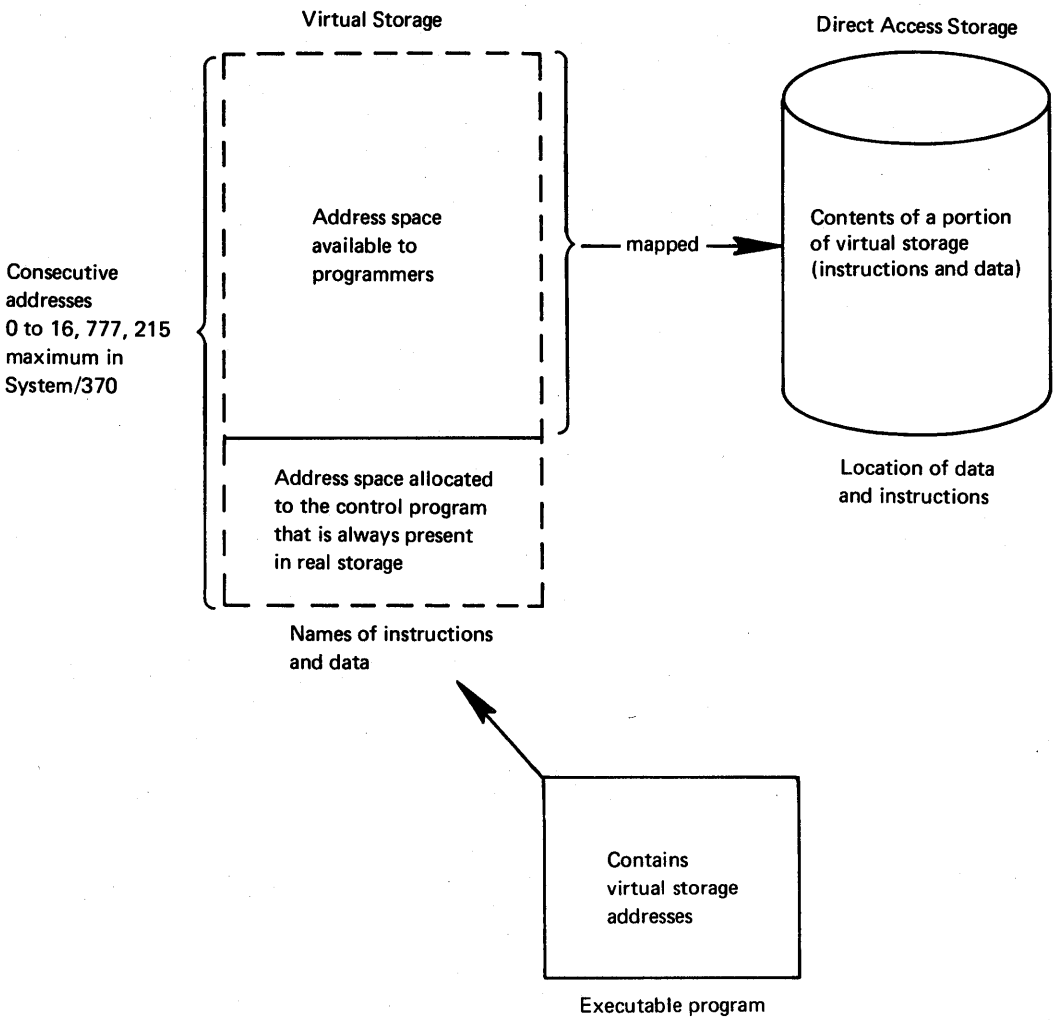

Virtual Storage
Direct Access Storage
Address space available to programmers
mapped
Contents of a portion of virtual storage (instructions and data)
Consecutive addresses 0 to 16, 777, 215 maximum in System/370
Address space allocated to the control program that is always present in real storage
Location of data and instructions
Names of instructions and data
Contains virtual storage addresses
Executable program

Figure 30.05.1. Names and location of instructions and data in a virtual storage environment

For the purpose of resource management in a virtual storage environment, virtual storage and its contents, direct access storage used to contain a portion of the contents of virtual storage, and real storage are divided into contiguous fixed-length sections of equal size. Once a program has been fetched from a program library and initiated, instructions and data within a program are transferred between real storage and direct access storage a section at a time, during program execution. A section of an executing program is brought into a real storage section only when it is required, that is, only when a virtual storage address in the section is referenced by the executing program. A program section that is present in real storage is written in a direct access storage section only when the real storage assigned to it is required by another program section and only if the section has been changed.

A virtual storage operating system control program monitors the activity of the sections of all executing programs and attempts to keep the most active sections in real storage, leaving the least active sections in direct access storage. Figure 30.05.2 illustrates the relationship of virtual storage, direct access storage, and real storage without regard to a specific virtual storage operating system implementation.

The division of a program and its data into sections and the transfer of these sections between direct access storage and real storage during program execution is handled entirely by the virtual storage operating system without any effort by the programmer. When a planned overlay or dynamic overlay program structure is used, the programmer is responsible for dividing the program and its data into phases, determining which phases can be present at the same time in the amount of real storage available (partition or region), and indicating when phases are to be loaded into real storage during processing.

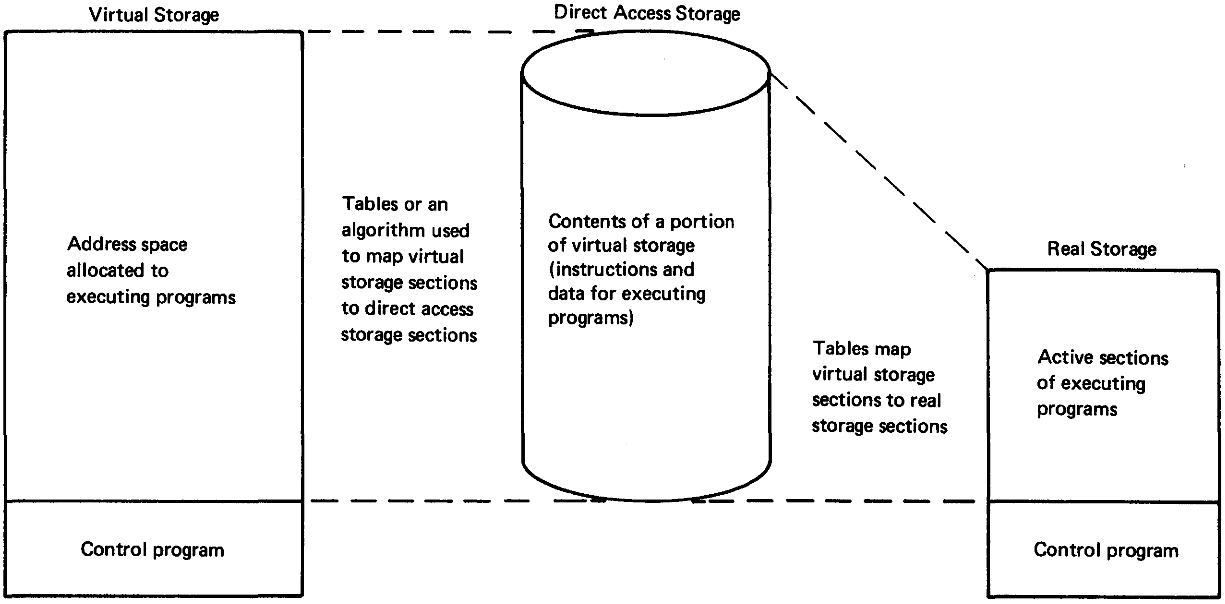

Virtual Storage
Direct Access Storage
Address space allocated to executing programs
Tables or an algorithm used to map virtual storage sections to direct access storage sections
Contents of a portion of virtual storage (instructions and data for executing programs)
Tables map virtual storage sections to real storage sections
Real Storage
Active sections of executing programs
Control program
Control program

Figure 30.05.2. Relationship of virtual storage, direct access storage, and real storage

While a virtual storage up to 16 million bytes in size can be addressed by any System/370 model with DAT hardware, the virtual storage size that can be effectively implemented by a given system is affected by (1) the amount of real storage present, (2) the amount of direct access storage space made available to contain the contents of virtual storage, (3) the speed of the direct access storage devices containing virtual storage contents and contention for these devices or the channels to which they are attached, (4) the speed of the CPU, and (5) the characteristics of the programs operating concurrently. Hence, the amount of real storage required to effectively implement a specific amount of virtual storage can vary by system, depending on the characteristics of the applications in the workload and the performance desired, as is discussed in Section 30:15.

Once a program section has been loaded into real storage, its virtual storage addresses can be translated when they are referenced. Dynamic address translation hardware is the mechanism that translates the virtual storage addresses contained in instructions into real storage addresses during instruction execution. Address translation is accomplished in System/370 using a hardware-implemented table lookup procedure that accesses tables contained in real storage. These tables, which are maintained by control program routines, (1) define the amount of virtual storage supported and allocated, (2) indicate whether any given program section is currently present in real storage, and (3) contain the addresses of real storage sections allocated to the program sections that are currently present in real storage.

During the execution of each instruction, address translation is performed on any virtual storage address in the instruction that refers to data or to an instruction. Translation occurs after the 24-bit effective virtual storage address has been computed by adding together base, displacement, and, if any, index values as usual. The result of the address translation is a 24-bit real storage address designating the location containing the data or instruction referenced by the virtual storage address in the instruction. The virtual storage addresses in channel programs (CCW lists) are not translated by channel hardware during channel program execution; therefore, programmed translation is required prior to initiation of a channel operation.

In reality, DAT hardware provides dynamic relocation of the sections of a program during its execution. This capability is not provided by OS MFT and OS MVT, which support program relocation at link-edit and program load time only. Once a program has been loaded into an area of real storage by the program fetch routine, these operating systems cannot relocate the program to another area of real storage during its execution. Thus, an entire program or a portion of a program cannot be written on direct access storage during execution and later reloaded into different real storage locations to continue execution. Once loaded, therefore, a program is bound during its execution to its initially allocated real storage addresses. In a virtual storage environment, a program is bound only to the virtual storage addresses it was assigned during loading.

The dynamic relocation provided by DAT hardware eliminates, for most programs, the need for allocating and dedicating a contiguous area of real storage to an entire program for the duration of its execution, a requirement for all programs in MFT and MVT. (As discussed later in this subsection, some programs cannot operate in the manner being described, that is, with sections transferred only as required between direct access storage and real storage.) In a virtual storage environment, real storage is no longer divided into contiguously addressed partitions or dynamically allocated regions that can contain one executing job step (program) at a time.

Further, when real storage is allocated to a section of an executing program, the real storage is not dedicated to that program section for the duration of program execution. Concurrently executing programs can dynamically share the same real storage sections. That is, in general, the real storage available for allocation to executing programs can be allocated to any program section as needed. When a section of an executing program must be loaded, any available section of real storage can be assigned (subject to certain restrictions imposed by operating-system-dependent real storage organizations). When the program section is no longer required, it can be written in direct access storage, if it has been altered, and the real storage assigned to it can be made available for allocation to another section of the same program or to a section of another program.

The assignment of real storage sections is handled entirely by the operating system, which also keeps account of which sections of concurrently operating programs are the most active. The operating system does not attempt to allocate a given amount of real storage to each executing program. It merely allocates real storage to those sections it determines are the most active, without taking into account the particular program to which the active section belongs.

DAT hardware, therefore, provides more than translation from address space (virtual storage) to real storage space. It provides the capability of implementing dynamic real storage management that requires no effort on the part of the programmer and significantly less CPU time than programmed address translation during program execution. (The large amount of CPU time required to translate addresses during program execution using programmed means has precluded implementation by IBM of an operating system that supports such programmed dynamic address translation.) Much of the real storage utilization preplanning required for OS MFT and MVT environments in order to use real storage effectively can be eliminated in a virtual storage environment. Dynamic real storage management capability is another advantage the technique of using direct access storage and DAT hardware to support a larger address space has over using larger real storage to provide a larger address space.

Another capability made available by the implementation of large address space using direct access storage and dynamic address translation is that of supporting more than one virtual storage with only one system. Multiple virtual storages are supported by OS/VS2 MVS (Releases 2 and up) and also can be used to support multiple virtual machines. A discussion of the concepts and general advantages of virtual machines is contained in Section 40. The features and operation of VM/370 are presented in Virtual Machine Facility/370 Features Supplement.

The use of virtual storage and DAT hardware to enable programs to operate in less real storage than the total storage requirement of the programs can also offer better performance potential than the technique of using a planned overlay program structure. When a planned overlay program executes in MFT or MVT, considerable time can be spent executing the overlay supervisor in order to perform programmed address translation (relocation) when a program phase is loaded. In addition, more efficient real storage utilization may be achieved in a virtual storage environment, since the control program reacts to changing processing needs and only portions of the program that are actually required are loaded (all phases of an overlay program may not be the same size and all code within a phase may not be used when the phase is loaded). Once a planned overlay program has been structured to handle the currently required set of program phases efficiently, it cannot automatically adapt to a change in the set of program phases required or to a change in the activity of the required set of phases.

In a virtual storage environment, the performance of the system can be directly affected by the amount of time spent transferring program sections between direct access storage and real storage. Satisfactory system performance is achieved when each of the concurrently executing programs has enough real storage dynamically allocated to it to keep the need for transferring program sections into and out of real storage at an acceptable level.

As previously mentioned, most programs can be structured so that processing activity is localized in one area of the program or another during time intervals rather than equally spread over the entire program. In other words, at any given time period during execution of the program, only a subset of the entire program need be referenced. This is sometimes called the "locality of reference" characteristic of programs. Therefore, a program achieves satisfactory performance when its most active sections in any given time interval remain in real storage and there is a limited amount of program section transfer activity.

Most programs require a certain minimum amount of real storage in which to execute in order to achieve satisfactory performance. If such programs operate with less than their minimum requirement dynamically allocated, program section transfer activity increases and performance degradation can occur. The minimum real storage requirement of a program is related to the amount of real storage required by the most active sections of the program. Because of the locality of reference characteristic of most programs, the minimum real storage requirement of a program for satisfactory operation frequently can be less than its total storage requirement. This fact enables an operating system to efficiently support a virtual storage that is larger than the real storage actually present in the computing system.

A virtual storage environment, therefore, enables most programs to be independent of real storage size to a large degree. A program can execute using varying amounts of dynamically available real storage without being modified. The amount of real storage dynamically available to a program during its execution primarily affects its performance, to the extent that program section transfer activity is affected, rather than its capability to be executed. For example, while a given 200K language translator might be able to operate with an average of 100K of real storage dynamically available to it during its operation, the time required to compile a program under these conditions might be unacceptable. Alternatively, the performance desired might be achieved if an average of 130K is dynamically available to the language translator while it operates. Without a virtual storage operating system, the 200K language translator might not be used at all because of its design point size.

In addition to the requirement for larger address space, there is still a requirement for larger real storage sizes in order to meet the functional and performance needs of the larger, more complex, multiprogramming environments. The availability of large lower-cost real storage for the Model 168 and the real storage independence that a virtual storage environment offers provide new flexibility in tradeoffs among real storage cost, function, and individual program or total system performance.

GENERAL ADVANTAGES OFFERED BY IBM OPERATING SYSTEMS THAT SUPPORT A VIRTUAL STORAGE ENVIRONMENT

Each of the IBM operating systems that supports a virtual storage environment for System/370 models using dynamic address translation offers the capability of using address space that is larger than that provided by available real storage, and each supports dynamic real storage management that is transparent to the user. As a result, these operating systems offer certain general potential advantages that do not depend on their unique features. The implementation of virtual storage also provides benefits that are specific to each of these operating systems because of their design and the particular functions they support. The following discusses the potential advantages of virtual storage and dynamic address translation that are common to OS/VS1 and OS/VS2 environments.

The general advantages of virtual storage operating systems are the potential they offer for:

- Increased application development

- Expanded operational flexibility

- System performance improvement

A virtual storage operating system can facilitate more rapid development of new applications because, by removing most existing real storage restraints on application design, it can help improve the productivity of programmers. Specifically, a virtual storage operating system has characteristics that can be used to reduce the effort, time, and cost associated with application design, coding, testing, and maintenance. This makes the installation of new applications more readily justifiable and encourages the addition of new functions to existing applications. The potential advantage of improved operational flexibility is made possible by the greater independence of applications from real storage size. Enhanced system performance can result from improved real storage utilization. While these latter two benefits have their own individual value, they also, either indirectly or directly, ease the installation of new applications.

Potential for Increased New Application Development

The following capabilities are characteristic of a virtual storage operating system environment:

- Greater flexibility in the design of applications is possible.

  Larger programs can be written without the necessity of using planned overlay techniques or other dynamic program structures designed to fit programs into the amount of real storage available. The need for intermediate (or working) data sets is reduced or eliminated because tables, relatively small data groups, etc., that are placed on direct access storage because of real storage limitations can become part of the program and will be brought into real storage automatically as required. Program planning, coding, and testing time can be reduced by elimination of the use of these programming techniques and other real storage management facilities, which also require additional programming knowledge and skill. Also avoided is the restructuring of application programs after they have been written because they are larger than the real storage available for their execution. Hence, applications can become operational more quickly.

  Open-ended, straightforward application design is possible, and more comprehensive programs can be written. An application can be segmented into a series of programs according to its logical flow instead of according to the functions that can be performed in the specific amount of real storage available to each step in the application. Programming and processing duplication inherent in the approach of using two or more job steps to perform one logical process is thereby avoided.

  Additional programming facilities can become available that otherwise could not be used because of real storage limitations. Specifically, full-function high-level language translators, which offer more capabilities than their subset versions (such as additional debugging facilities and performance options) but which also have larger storage design points, can be used because they can operate in a virtual storage environment using less real storage than their design point requirement.

- Preproduction testing of larger-than-average application programs can be increased if enough virtual storage can be made available to enable them to run during normal testing periods. Turnaround time during testing can be reduced.

  In a nonvirtual storage environment such programs are usually grouped together and executed only at certain times when their larger design point storage requirements can be made available.

- Fine tuning of application programs to achieve performance improvements, when necessary, can be delayed until after the application is in production. This capability enables an application to become operative sooner.

- Startup costs for new applications may be reduced.

  A new application can be developed and tested on the existing system, assuming the required I/O devices are present in the configuration, before the additional real storage the application requires for performance on a production basis is actually installed. When the application is ready for production, the additional real storage required can be added to the system. In some cases it may be possible to operate the application on a production basis on the existing system without adding real storage initially, because during the startup period, transaction volume is very low. As the volume grows, real storage can be added to achieve better performance.

- Growth of existing applications and the maintenance of operational programs is simplified.

  Because of the removal of most real storage restraints, new functions can be more easily and more rapidly added to most existing applications. Program expansion because of added functions or maintenance changes does not require the use of overlay techniques, multiple job steps, etc., when the size of the extended program exceeds the original storage design point size.

  In general, alteration and debugging of nonoverlay programs are also easier than alteration and debugging of programs with planned overlay or dynamic structures.

- Application programs whose real storage requirements, based on transaction volume and complexity, vary widely during their execution may be justified, designed, and installed more easily.

  Design, coding, and testing time can be reduced because dynamic storage management is automatically provided by the operating system. Time and effort need not be spent structuring such programs to use available real storage dynamically to support the functions and/or response times required.

- Design and installation of one-time, low-usage, or low-volume programs of very large storage size are more easily justified. Existing applications in these categories that currently operate in a batch environment can also more easily be altered to operate online, a growth step that might not be justifiable in a nonvirtual storage environment.

- Applications can be installed on a trial basis for the purpose of observing and evaluating their functions and their operation.

  Most IBM-supplied application program products can be temporarily installed on an existing system, assuming the required I/O devices are present. The additional hardware resources that may be required to operate the application on a production basis can be added later, when the application is permanently installed.

- The benefits of the functions provided by many IBM-supplied application program products with larger storage design points can be realized using smaller amounts of available real storage.

  It may be difficult to justify the real storage required to install a relatively large storage design point application on a system to handle a low volume of transactions, even though the functions provided by the application are very desirable. In a virtual storage environment, such an application can execute using that amount of dynamically available real storage required to satisfy the desired performance requirements for the low volume of activity.

Potential for Additional Operational Flexibility

The reduction of real storage restraints makes most applications more independent of the real storage size of a system configuration and permits most applications to be processed on systems with varying amounts of available real storage without program modification. Dynamic real storage management reduces the amount of job stream and operations preplanning that is normally done to use real storage as efficiently as possible in a multiprogramming environment. The following benefits can be the result:

- A system can back up another system even though it has less real storage than the system it backs up.

  A smaller-scale system with the appropriate I/O configuration can provide backup for a larger-scale system if necessary. (Performance experienced on the backup system may vary from that normally achieved depending on the two system configurations involved.)

- A single design and one operating procedure can be used for an application that is to operate on multiple systems with varying amounts of real storage, as long as the virtual storage required is supported by all the systems.

  When data processing is decentralized among multiple installations with systems that have different amounts of real storage, one location can design, implement, and maintain an application that can be used by other installations. Duplication of this type of effort can be minimized or eliminated.

- Most applications can be tested on systems with less real storage than the one on which they will run in a production environment, as long as the required amount of virtual storage is supported.

- Growth to a larger real storage configuration can be easier.

  Real storage can be added to an existing system to improve system performance (by the reduction of program section transfer activity) without the necessity of modifying existing application programs so that they take advantage of additional real storage. Additional real storage (up to a maximum of their design point size) is automatically used by programs that operate in a virtual storage environment.

- Operators need not perform certain procedures that are solely related to efficiently managing real storage.

  The operator is concerned with the division of virtual storage and therefore need not change partition sizes at various times (in OS/VS1, for example) for the purpose of making storage available for larger than average jobs. (An installation can define virtual storage partitions that are larger than those currently defined in the OS MFT environment, and the partitions can be made big enough to contain the largest existing or currently planned storage design point programs.) Similarly, in an OS/VS2 environment, the operator no longer need start long-running jobs at certain points in time to ensure that available real storage is fragmented as little as possible.

- Priority jobs whose need to be processed cannot be predicted can be scheduled when required.

  A nonvirtual storage environment does not provide the capability of effectively handling the scheduling of high-priority jobs on a random basis. Hence, this type of job is not permitted to exist in an installation, or such jobs must be scheduled to operate only at certain times. In a virtual storage environment, a high-priority virtual partition can be defined in an OS/VS1 environment and reserved for the purpose of processing only high-priority jobs. Except for that required for certain tables, real storage is not required for this partition until a job is actually scheduled. In an OS/VS2 environment, an initiator with a special class can be started that will handle only high-priority jobs. This can be done in MVT as well but because of the possibility of real storage fragmentation, there is no assurance that the high-priority job can be started.

Potential for Performance Improvement

The improved real storage utilization made possible by the use of dynamic address translation hardware can have a positive effect on the performance of a system that handles a job mix whose use of real storage varies considerably while it is being processed. The extent of the performance improvement depends on the types of applications involved and the current utilization of system resources. Therefore, the amount of performance gain, if any, that may be achieved is highly variable by installation. Environments with the greatest potential for improved performance are as follows:

- Batch-oriented multiprogramming environments with application programs of widely varying real storage requirements.

  Real storage may not be most efficiently used in such an environment because (1) real storage can become fragmented when regions are dynamically allocated and freed or (2) it is difficult to divide real storage into a set of areas that is optimum for all programs when real storage is partitioned. (Consider the inefficient use of real storage in an 80K partition allocated for assemble, link-edit, and test jobs in which an 80K language translator, a 44K linkage editor, and problem programs no larger than 60K execute.) In addition, real storage is not efficiently used when the real storage requirement of a given program, based on transaction mix or volume, varies widely, and the amount of real storage that is allocated is designed to handle the peak requirement. (This is typically true of graphics applications, for example.) Further, real storage assigned to a program is not productively used during the time the program is waiting for a human response, such as for the operator to locate and/or mount a volume or to make a decision and enter a message on the console, or during the time required to quiesce the system in order to change partition definitions, start high-priority jobs, or start a teleprocessing program in high real storage.

  In a virtual storage environment, in which all concurrently executing job steps share real storage dynamically and use real storage only when it is actually required for program execution, real storage is more efficiently used. Hence, if real storage currently is the restraint, a given real storage size might be capable of supporting a higher level of multiprogramming than can be achieved without the use of dynamic storage management (assuming other required resources, such as CPU time, I/O devices, and channels, are available). For example, installation of a large storage design point, terminal-based application to handle only a few terminals might be possible. Alternatively, a higher level of multiprogramming might be supported by the addition of a smaller real storage increment than would otherwise be required.

  System performance may also be improved if more efficient use of available real storage enables additional heavily used functions to be made resident instead of transient or allows the incorporation of performance-oriented options in the control program. This improvement can apply to environments with batch and online operations, as well as to batch-only multiprogramming environments.

- Multiprogramming environments with a mixture of batch-oriented and terminal-based applications.

  While the real storage required for the communication control portion of a teleprocessing application remains constant, terminal-based processing programs are typically subject to wide variations in the amount of real storage they require during their execution because the transaction mix being handled concurrently varies, the activity of each terminal online varies, or the number of terminals operating concurrently changes. In order to provide the functions desired, ensure the capability of handling peak activity periods and maximum transaction type mixes, and guarantee a given response during times of peak activity, a certain amount of real storage is required. This peak requirement is generally significantly more than is needed during periods of medium and low activity. Allocation of the maximum storage requirement results in inefficient use of real storage, since unused real storage dedicated to any terminal program cannot be used by other concurrently operating batched or terminal-oriented jobs in a nonvirtual storage environment. In addition, it is usually difficult, and sometimes impossible, to effectively preplan real storage usage for an online application.

  Dynamic real storage management in a virtual storage environment automatically provides a much more efficient method of allocating real storage in such an environment. Real storage is not divided into that which can be used only by the terminal-based program(s) and that which can be used only by batched jobs. During times of peak terminal activity, the active sections of terminal-oriented processing programs with a higher priority are automatically allocated real storage, making less real storage available to the lower priority batched jobs in execution at that time. During periods when terminal activity is relatively low, real storage not used by any terminal program is available for assignment to the active sections of executing batched jobs. Such an environment is represented conceptually in Figure 30.05.3.

  In existing mixed batch- and online-oriented installations, dynamic real storage management allows programming techniques that can improve the performance of the online application. This improvement can be in the form of better response for existing terminals or the ability to support more terminals. A given online application may also be able to support a higher level of multiprogramming, as a result of better real storage utilization, without any additional programming effort (more TSO regions, for example). A virtual storage environment also makes the concurrent operation of multiple terminal-based applications more practical.

Figure 30.05.3 shows sample allocations of real storage to two batched jobs and two terminal-oriented jobs in a multiprogramming environment during low, medium, and peak activity points in time. Job priority from high to low is TP2, TP1, BJ2, BJ1. For simplicity, virtual and real storage are shown to be totally allocated at all times and no particular virtual storage operating system (OS/VS1 or OS/VS2) is assumed, since the concepts illustrated apply to both, regardless of differences in the way virtual storage is allocated by these operating systems. Real storage is shown to be contiguously allocated to each job in high-to-low priority sequence. This is done only to illustrate the relative amount of real storage the control program has dynamically allocated to each program during the instant shown. In reality, the total amount of real storage allocated to an executing program at any given time is usually not contiguous in a virtual storage environment. In addition, during times of low terminal program activity, it may be possible to support a higher level of batched job multiprogramming, which is not shown in the figure.

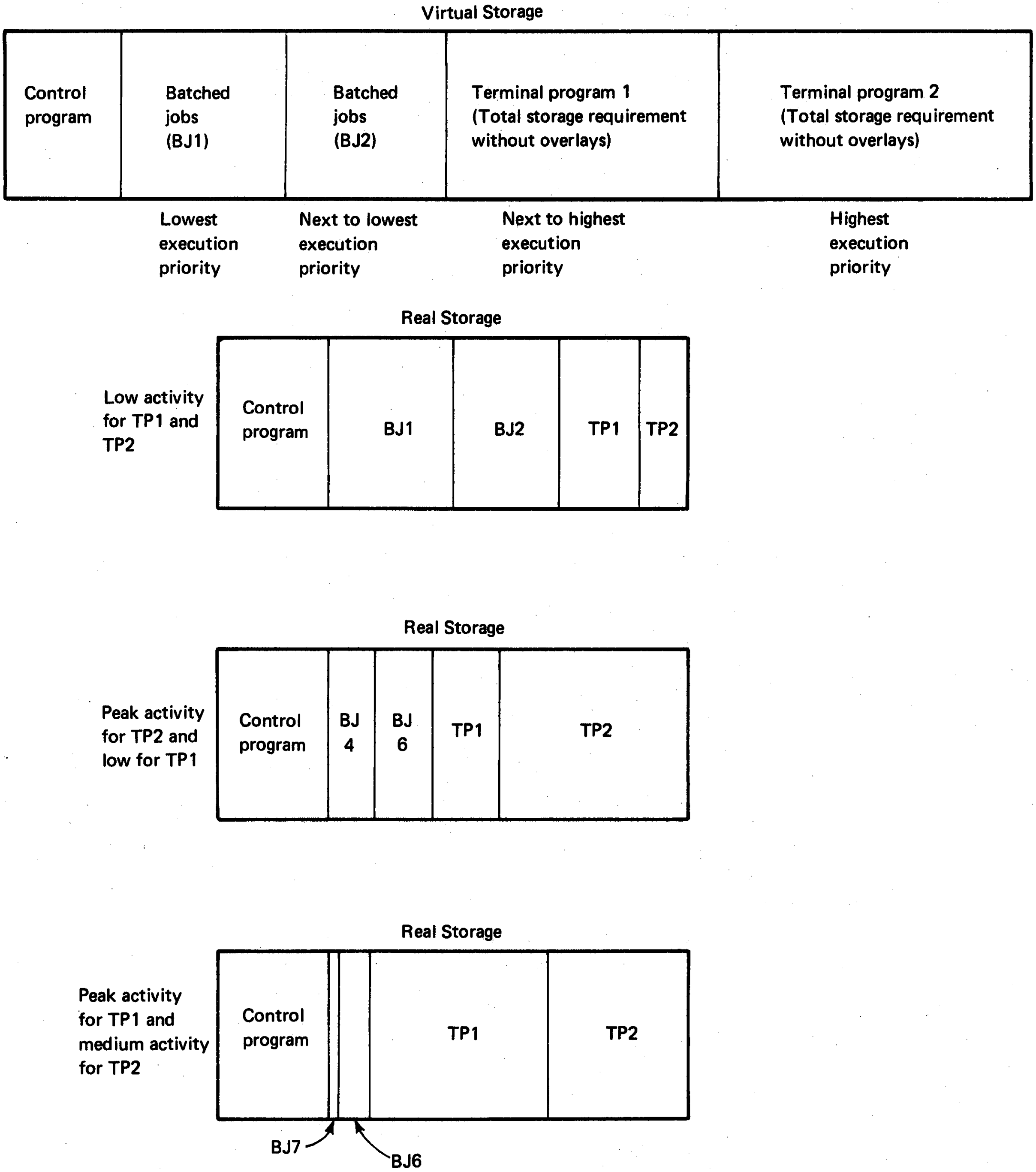

Virtual Storage
Control program
Batched jobs (BJ1)
Batched jobs (BJ2)
Terminal program 1 (Total storage requirement without overlays)
Terminal program 2 (Total storage requirement without overlays)
Lowest execution priority
Next to lowest execution priority
Next to highest execution priority
Highest execution priority
Real Storage
Low activity for TP1 and TP2
Control program
BJ1
BJ2
TP1
TP2
Real Storage
Peak activity for TP2 and low for TP1
Control program
BJ 4
BJ 6
TP1
TP2
Real Storage
Peak activity for TP1 and medium activity for TP2
Control program
TP1
TP2
BJ7
BJ6

Figure 30.05.3. Conceptual illustration of real storage utilization in a mixed batch and online virtual storage environment

<u>Summary</u>

As the preceding discussion indicates, a virtual storage environment is designed primarily to provide new functional capabilities for the installation as a whole, although performance gains are possible for installations with particular environmental characteristics. The general functional aims of IBM-supplied virtual storage operating systems are (1) to use new hardware features and additional control program processing to support certain facilities that are not possible in a nonvirtual storage environment because of real storage restraints, and (2) to handle other functions that must be performed by installation personnel (programmers, operators, and system designers) when virtual storage and dynamic address translation are not used.

It is also important to note that while a virtual storage operating system permits an installation to be independent of real storage restraints to a large degree and enables real storage to be utilized more efficiently, the performance of the system and the specific advantages that can be achieved are still largely dependent on the amount of real storage present in the system and on the computing speed of the CPU, among other things. Hence, virtual storage and dynamic address translation are not a substitute for real storage. Rather, they provide an installation with greater flexibility in the tradeoff between real storage size and function or performance.

The degree to which a particular installation experiences the potential benefits of a virtual storage/dynamic address translation environment is system-configuration dependent and highly application dependent (number, type, complexity of applications installed). In addition, consideration must be given to the system resources that are specifically required to support a virtual storage environment (discussed in Section 30:15). Some of the potential advantages, such as those associated with application maintenance and operational flexibility and those that result from better management of real storage, can be experienced as soon as a virtual storage operating system is installed. Others may be achieved in the future when new applications are installed, and the less restrictive program design techniques available in a virtual storage environment are more fully utilized. In any case, installation of a virtual storage operating system can make System/370 easier to use and can be a major step toward more rapid installation of applications. Such an operating system can be of greatest benefit to installations desiring to move to or to extend online operations and thereby attain the advantages such an environment offers.

VIRTUAL STORAGE AND DYNAMIC ADDRESS TRANSLATION TERMINOLOGY

For the purpose of presenting the concepts of virtual storage and dynamic address translation in the previous discussion, virtual storage, programs and data, direct access storage, and real storage were described as being divided into areas called sections. In reality, a unique term is used to describe each one of the various sections, namely, virtual storage page, page, slot, and page frame. In addition, virtual storage has two levels of subdivision in System/370. The following defines the new terminology actually used by the System/370 virtual storage operating systems.

Virtual storage in System/370 is divided into contiguous segments, which contain virtual storage pages. A virtual storage segment, as implemented in System/370, is a fixed-length, consecutive set of addresses for either 64K or 1024K bytes that begins on a 64K or 1024K boundary, respectively, in virtual storage. A virtual storage is divided into segments all of one size or the other. In general, in OS/VS1 and OS/VS2 environments, a segment is the unit of virtual storage allocation. Each segment of virtual storage is divided into contiguous, fixed-length, consecutive sets of addresses called virtual storage pages. Each segment in the virtual storage contains the same number of virtual storage pages, each of which is the same size. A virtual storage page, as implemented in System/370, can be either 2K or 4K bytes and is located on a 2K or 4K virtual storage boundary, respectively, within a segment.

The contents of virtual storage--instructions and data--are divided (by the operating system) into fixed-length contiguous areas called *pages*, corresponding in size to the virtual storage page size chosen, either 2K or 4K bytes. The addresses associated with a virtual storage page refer to the contents of a page.

The direct access storage used to contain the portion of the total contents of virtual storage that is not always present in real storage is called *external page storage*. Direct access space within external page storage is divided into physical records called *slots*, which are of page size, either 2K or 4K bytes. A slot, therefore, can contain one page at a time. A virtual storage page that is allocated and that actually has contents usually has a slot in external page storage associated with it to contain these contents (depending on the nature of the contents and how external page storage is managed by the operating system).

Instructions and data are transferred between external page storage and real storage as needed on a page basis. This transfer process is called *paging*, and a direct access device that contains external page storage is called a *paging device*. A slot in external page storage is associated with a particular virtual storage page by means of an algorithm or via tables that are maintained by the control program.

Real storage also is divided into fixed-length, consecutively addressed areas called *page frames*, which are always the same size as the virtual storage page being used, either 2K or 4K bytes. Page frames are located on 2K or 4K real storage boundaries. A page frame is a block of real storage that can contain one page. Hence, a page of data and/or instructions occupies a slot when it is in external page storage and a page frame when it is in real storage. Whether or not a page is present in real storage, a program addresses the contents of the page using virtual storage addresses.

The act of transferring a page from external page storage into real storage is called a *page-in*. This action may also be described as the *loading of a page*. The reverse act, transferral of a page contained in real storage to a slot in external page storage, is called a *page-out*. Figure 30.05.4 illustrates the relationship of virtual storage, external page storage, and real storage that was conceptually shown in Figure 30.05.2. (Note that the terms *swap-in*, *swap-out*, and *working set* have a specific meaning in an OS/VS2 TSO environment and are defined in *OS/Virtual Storage 2 Single Virtual Storage (SVS) Features Supplement*. The definition of a *working set* in a virtual machine environment is given in *Virtual Machine Facility/370 Features Supplement*.)

As previously indicated, DAT hardware uses tables to perform address translation. These tables are the *segment table* and *page tables*. One segment table and a set of page tables are required to perform address translation for one virtual storage. The segment table defines the virtual storage size, indicates allocated virtual storage, and points to the real storage location of the page tables. The page tables indicate which pages are currently in real storage and contain the real storage addresses of these pages. As pages are paged in and out, the control program makes changes to the page tables as required.

Basic to the implementation of virtual storage using direct access storage and DAT hardware is the method of determining when pages are to be brought into real storage and, therefore, when real storage is allocated to pages. The method supported by IBM-supplied virtual storage operating systems, that of bringing a page into real storage only when it is needed by an executing program, is called a *demand* paging technique. Since programs execute on a priority basis in OS/VS1 and OS/VS2 environments, as they do in OS (MFT and MVT) environments, real storage is, in effect, still allocated on a priority basis.

A request for a page-in is generated by the occurrence of a page exception or a page translation exception, a condition that is also called a page fault. An interruption occurs during the execution of an instruction when DAT hardware attempts to translate a virtual storage address into a real storage address and the appropriate page table indicates that the page is not currently present in real storage. A page fault condition causes an interruption in order to alert the control program to the fact that a page frame must be allocated. Usually, a page-in is required also to bring in the referenced instruction or data.

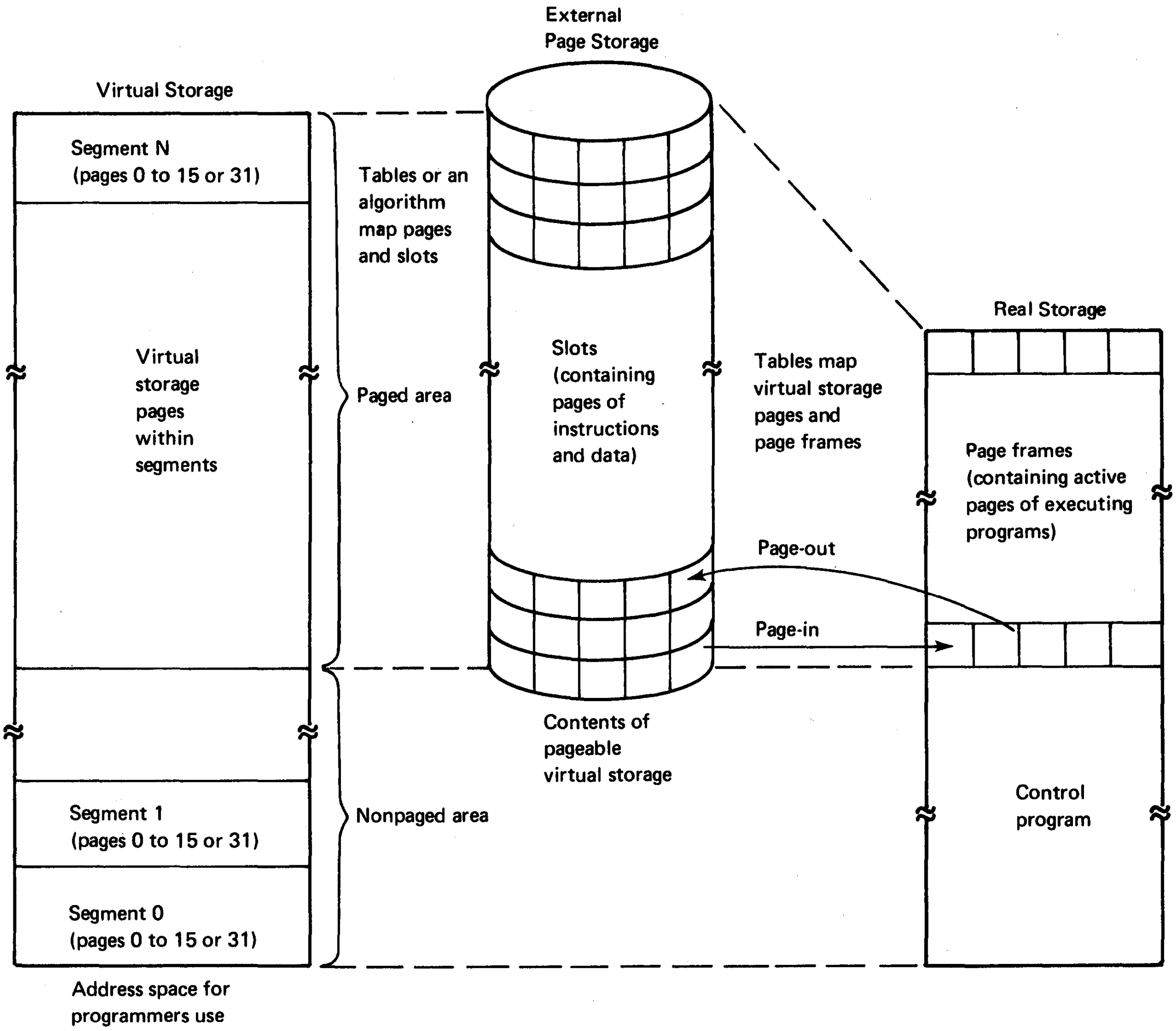


Figure 30.05.4. Layout of virtual storage, external page storage, and real storage

While page-ins are usually initiated as a result of a page fault, OS/VS1 and OS/VS2 provide an Assembler Language macro that can be used to cause one or more pages to be brought into real storage before they are referenced. Such requests are sometimes referred to as page-ahead requests. A page-ahead is required if, for reasons of proper system operation, a routine must operate without incurring any page faults. Use of this macro is restricted because unlimited use of this facility can defeat the objective of demand paging.

When a page fault occurs and the control program determines that a page frame is not currently available for allocation, a choice must be made as to which allocated page frame will be taken away from the page to which it is currently assigned. The rule governing this choice is called the page replacement algorithm. If the page replacement algorithm is designed to choose from among only those page frames currently allocated to the program that caused the page fault, it is said to operate locally. If a page frame can be chosen from among all those available for allocation to all executing programs, the algorithm is said to operate globally. OS/VS1 and OS/VS2 implement a global page replacement algorithm. VM/370 supports a global page replacement algorithm and supports a local page replacement algorithm as an option. The algorithms used attempt to keep the most active pages of executing programs present in real storage. Hardware is included in System/370 models with dynamic address translation that indicates whether a page has been referenced or changed. Hence, when a page frame is required, a page determined by the algorithm to be relatively inactive is chosen for replacement.

Before a new page is loaded into the page frame chosen, the existing contents of the page frame must be saved if they were modified during processing. If modification occurred, a page-out operation is required; otherwise, an exact copy of the page already exists in external page storage. Code that is not modified during its execution, therefore, has an additional advantage in a virtual storage environment in that it need never be paged out once it has been written in external page storage. A program requiring a page-in is placed in the wait state until the page it requires has been loaded, during which time CPU control is given to another ready task, if one is available.

For various reasons, it is necessary to prevent a page-out of certain pages that are in real storage. One reason is for better operation of the system. This reason applies to certain frequently used control program routines, some routines that operate with the CPU in a disabled state (masked for I/O and external interruptions), most system tables, and most system control blocks. Integrity of system operation is another reason. Pages associated with certain types of operations must not be paged out while the operation is in progress, so that the operation can proceed correctly. For example, pages that contain I/O buffer areas must remain in real storage while the buffers are being referenced during an I/O operation, after which a page-out can take place, if necessary. Another reason is the existence of time dependency. A page should not be written out if the program to which the page belongs must complete a logical operation that requires the page in less time than it takes to perform a page-in. Programs that handle I/O device testing operations, such as online tests (OLT's), can have such a time dependency.

A page that is identified as one that cannot be paged out (or that is nonpageable) is called a fixed page in OS/VS1 and OS/VS2 and a locked page in VM/370. OS/VS1 and OS/VS2 support both long-term fixing and short-term fixing. Pages that should never be paged out when they are present in real storage are marked long-term fixed. The resident portion of an operating system control program is never paged and, therefore, its pages are marked long-term fixed. Pages that must be fixed for only a portion of the time they are present in real storage are marked short-term fixed. For example, a page containing an I/O buffer is marked short-term fixed before the initiation of the I/O operation that references the buffer. After the I/O operation completes, the page is unfixed and it becomes eligible for a page-out. Pages should be marked fixed only when necessary, since page fixing reduces the amount of real storage that can be shared by concurrently executing paged programs (that which is available to be allocated to the nonfixed pages) and can, therefore, affect system performance.

As indicated previously, in OS/VS1 and OS/VS2 environments, a portion of the control program is resident in real storage. Its pages are marked fixed. This portion of the control program is not placed in external page storage (because it is not paged) even though it is allocated space in virtual storage. Certain other portions of an OS/VS1 and an OS/VS2 control program are pageable and are made resident in virtual storage, which means they are contained in external page storage during system operation. During system initialization, these pageable control program routines are allocated virtual storage and loaded into real storage from system libraries by the program fetch routine. These routines will be written in external page storage as a result of normal paging activity in OS/VS1 and as a result of specific page-out requests in OS/VS2. Control program routines that are resident in virtual storage are brought into real storage from external page storage, instead of from a system library, when they are required during system operation.

Just as control program routines can be fixed or pageable, problem programs operate in one of two modes in OS/VS1 and OS/VS2 environments: paged mode or nonpaged mode. The latter is also sometimes called virtual equals real (V=R) mode. When a problem program operates in paged mode, it is resident in virtual storage and pageable. A pageable program operates in a contiguous area of virtual storage (partition or region) and is assigned available real storage on a demand paged basis. Hence, virtual storage addresses must be translated into real storage addresses. The real storage dynamically allocated to programs operating in paged mode need not be contiguous and such programs normally can operate with less real storage than their design point (virtual storage) amount dynamically allocated to them. This is the mode of operation described in Section 30:05.

Paged mode is the normal mode of operation of programs in a virtual storage environment. However, certain programs cannot operate correctly in this mode, and must run in nonpaged (V=R) mode. In general, a program must operate in nonpaged mode if it:

- Contains a channel program that is modified while the channel program is active (Section 30:10 discusses the reason)

- Is highly time dependent (involves certain testing operations on I/O devices, for example)

- Must have all of its pages in real storage when it is executing (for performance reasons, for example)

Other characteristics that require a program to be executed in nonpaged mode and that are operating system dependent are listed in the programming systems supplements, which also discuss steps that can be taken to avoid running a program in nonpaged mode.

In OS/VS1 and OS/VS2 environments, a program that operates in nonpaged mode is dynamically allocated a contiguous virtual storage area and a contiguous real storage area of the same size with addresses identical to those of the allocated virtual storage area. (That is, virtual and real storage addresses of the allocated area are equal.) Since programs operating in V=R mode are not paged, they do not occupy external page storage. The entire program (except for dynamically requested modules) is loaded into real storage when it is initiated, and all its pages are fixed. The amount of real storage allocated to a program that runs in nonpaged mode must be a multiple of the page size used.

30:10 DYNAMIC ADDRESS TRANSLATION HARDWARE FOR MODELS 1 AND 3 OF THE MODEL 168

Dynamic address translation is a standard facility of the Model 168. It is made operative by turning on the translation mode bit in the current PSW. The system must also be operating in EC mode. When DAT is operative, storage addresses in programs referring to instructions and data are translated into real storage addresses after instructions are fetched during program execution. The address in the instruction counter is translated also. When DAT is not in operation, storage addresses in programs are used as real storage addresses. The storage addresses in CCW lists are not translated by channel hardware during channel program operation. The channel indirect data addressing feature, required on all installed channels for a Model 168 when a virtual storage operating system is used, and programmed channel program translation are discussed later in this subsection under "Channel Indirect Data Addressing".

The following instructions are associated with dynamic address translation: LOAD REAL ADDRESS, RESET REFERENCE BIT, and PURGE TLB. These instructions are valid in BC mode as well as in EC mode. They operate identically regardless of which mode is in effect. All are privileged instructions.

VIRTUAL STORAGE ORGANIZATION

The Model 168 (as well as other System/370 models with DAT hardware) supports a virtual storage segment size of either 64K or 1024K bytes, as determined by bits 11 and 12 of control register 0. With either segment size, the page size can be 2K or 4K, as determined by bits 8 and 9 of control register 0. A segment size of 1024K bytes is not supported by DOS/VS, OS/VS1, OS/VS2, or VM/370. Table 30.10.1 summarizes the virtual storage organization provided in System/370.

Table 30.10.1. Number and size of segments and pages for a 16-million-byte virtual storage

CR 0 Bits 11,12	 8,9	Segment Size (bytes	Number of Segments in the Virtual Storage	Page Size (bytes)	Number of Pages in a Segment
10	01	1,048,576	16	2048	512
10	10	1,048,576	16	4096	256
00	01	65,536	256	2048	32
00	10	65,536	256	4096	16

As already described, the addresses supplied in programs directly address a location in the virtual storage that is supported by the virtual storage operating system. In this sense, program-supplied addresses can be viewed as virtual storage addresses that specify a byte within a particular virtual storage page and segment. The logic of the translation process is described in this subsection in these terms. The architectural definition of dynamic address translation found in System/370 Principles of Operation (GA22-7000-2 and later editions) assumes that the addresses in programs consist of three fields, two of which are used to index tables during the translation process. Under these conditions, the addresses supplied by a program are considered to be logical addresses instead of virtual storage addresses.

For the purpose of translation, a virtual storage address is divided into three fields: (1) a segment field, which identifies a segment within the virtual storage, (2) a page field, which identifies a page within the segment addressed, and (3) a byte displacement field, which identifies a byte within the page addressed. The number of bits in each field varies depending on the segment and page sizes used. Virtual storage address fields for a segment size of 64K and a specific example of how the fields are used to address a location in virtual storage are shown in Figure 30.10.1.

OPERATION OF DYNAMIC ADDRESS TRANSLATION HARDWARE

Address Translation Tables

One segment table is required to describe one virtual storage. If more than one virtual storage is supported by a single computing system, there is a segment table for each virtual storage implemented. A segment table contains one four-byte entry for each segment in the virtual storage the table describes, up to a maximum of 256 entries for the maximum size virtual storage of 16 million bytes (using 64K segments). The real storage address of the segment table (or of the currently active segment table if multiple virtual storages are implemented) is contained in control register 1. The current length of the segment table is also indicated in control register 1. The length value is used by the hardware during translation to ensure that the segment entry being referenced falls within the segment table.

The segment table entries point to the real storage locations of the page tables. There is one page table for each segment in the virtual storage defined (or, in OS/VS2, currently allocated), up to a maximum of 256 page tables for a 16-million-byte virtual storage with 64K segments. A segment table entry contains an indication of the length of the page table, the high-order 21 bits of the real storage address of the page table, and an indication of whether or not the entry itself is valid and can be used for translation purposes (invalid bit). If the invalid bit is on in a segment table entry, a segment translation exception occurs during the translation process.

A page table has one entry for each page in the particular segment the page table describes. For a 64K segment, there are 32 or 16 entries in a page table depending on whether a 2K or a 4K page is used, respectively. A page table entry is two bytes in size. It contains the 12 (for a 4K page) or 13 (for a 2K page) high-order bits of the real storage address of the page frame that is currently allocated to the virtual storage page that the page table entry describes. Each page table entry also contains an invalid bit to indicate whether the entry can be used for translation. The invalid bit is on when a virtual storage page does not have real storage currently allocated to it. A page translation exception occurs during the translation procedure if this invalid bit is on.

Segment and page table formats and entries used for address translation are shown in Figure 30.10.2. In effect, the segment and page tables define the relationship between virtual and real storage at any given time. The segment table reflects the current size of virtual storage and the location of required page tables. The segment table also indicates, by means of its invalid bits, which segments of virtual storage are currently allocated and have a page table available. The page tables indicate, via their invalid bits, which virtual storage pages currently have a page frame allocated and the location (real storage address) of these page frames.

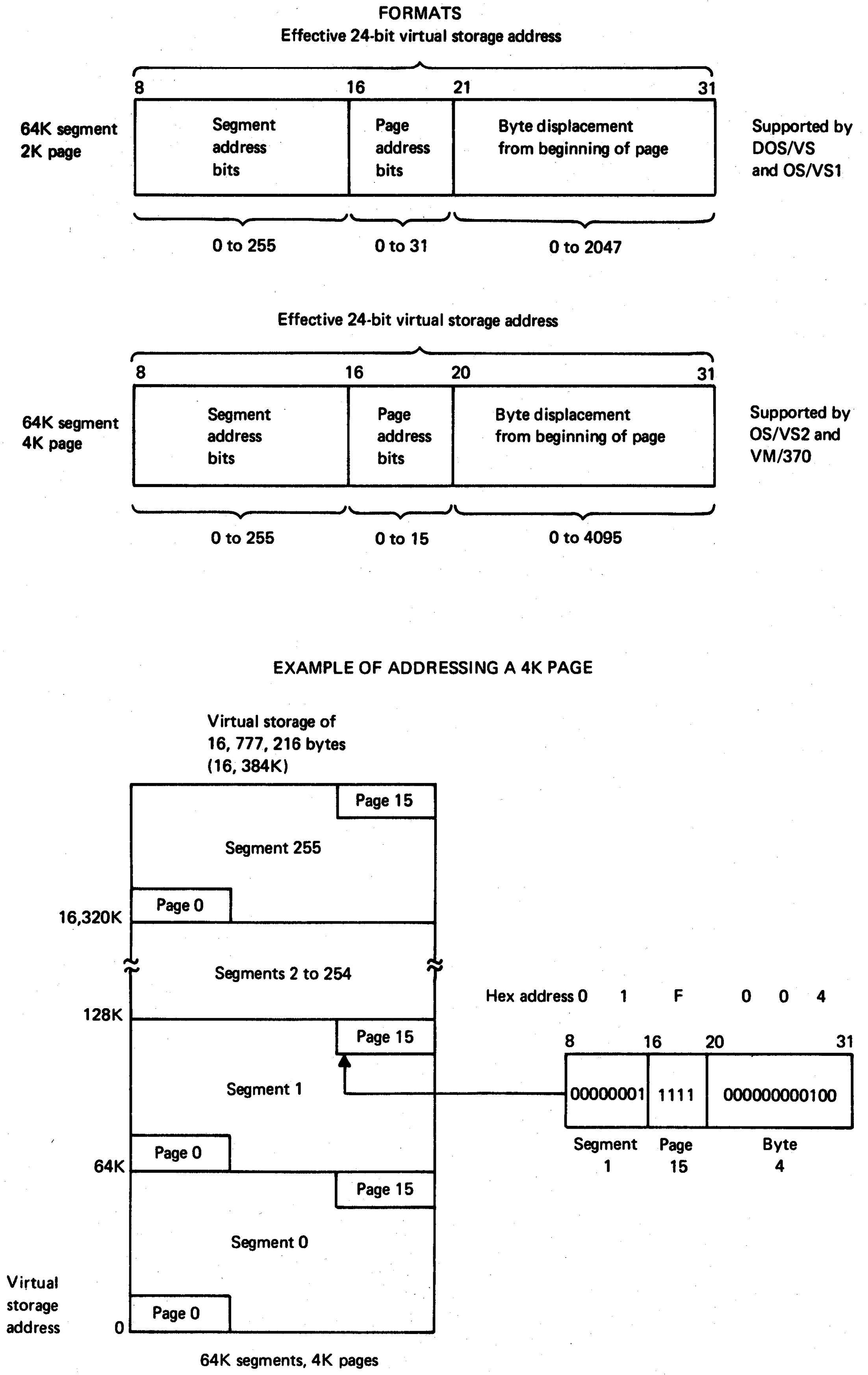


Figure 30.10.1. Virtual storage address fields for a 64K segment

In an OS/VS1 environment, segment and page tables are established at system initialization. Page tables are modified during system operation by control program routines to reflect the current allocation of real storage to virtual storage so that address translation can take place. In an OS/VS2 environment, in which virtual storage as well as real storage is dynamically allocated and deallocated, the segment table constructed during IPL is modified as required during system operation to reflect the allocation of virtual storage, and page tables are created and destroyed as necessary.

Address Translation Process

A translation request is either explicit or implicit. Explicit translation is invoked via execution of the LOAD REAL ADDRESS instruction. Implicit translation is invoked to translate all instruction addresses and data addresses contained in other instructions. Implicit address translation takes place during instruction execution.

The logical flow and the details of the translation process are given in Figure 30.10.3. The procedure consists of a two-level, direct address table lookup operation. Any type of translation exception causes a program interruption and termination of the hardware translation process. The CPU cannot be disabled for translation exception interruptions. Segment and page translation exceptions that occur during an explicit translation request (LOAD REAL ADDRESS instruction) are indicated via the condition code setting instead of via an interruption.

Translation Lookaside Buffer

In the Model 168, a translation lookaside buffer (TLB) is implemented to reduce the amount of time required to perform address translation. The translation lookaside buffer is used to retain up to 128 previously translated addresses. Addresses associated with up to six different virtual storages can be contained in the TLB at any time. Every time a virtual storage address is translated during instruction execution, the virtual storage address, the resulting real storage address and its associated storage protect key, and identification of the virtual storage to which the virtual storage address belongs are placed in one of the 128 TLB locations. A hashing algorithm is applied to the virtual storage address in order to determine which of the 128 TLB locations is to be used.

After the effective virtual storage address has been computed and before performing the translation using segment and page tables, the TLB is interrogated to determine whether it contains the required translated address. Interrogation of the TLB is done in parallel with reference to the index array for the buffer. Therefore, no translation cycles are required when the translated address is obtained from the TLB. If the TLB does not contain the required translation or if the entry is invalid, as indicated by a zero identification code, the complete table-lookup translation procedure, as previously described, is performed. In the Model 168, the number of CPU (80-nanosecond) cycles required for address translation when the translation is not obtained from the TLB varies from a minimum of 8 to a maximum of 26, assuming no I/O interference, depending on the locations of the segment table and the page table entries required for the translation. In the Model 165 II, from 8 to 46 CPU cycles are required for the translation process when the required translation is not contained in the TLB.

If an error occurs in the TLB, half of the TLB (64 locations) is disabled and a machine check interruption occurs if the CPU is enabled for degradation interruptions. The degradation bit will be on in the stored machine check code. The disabled half of the TLB is reenabled during an IPL, system reset, or when a bit that controls whether or not the disabling function is active is set to disable the function. The control bit is set via a DIAGNOSE instruction and is the same bit that controls high-speed buffer deletions. On a system reset, the control bit is set to enable the TLB delete function.

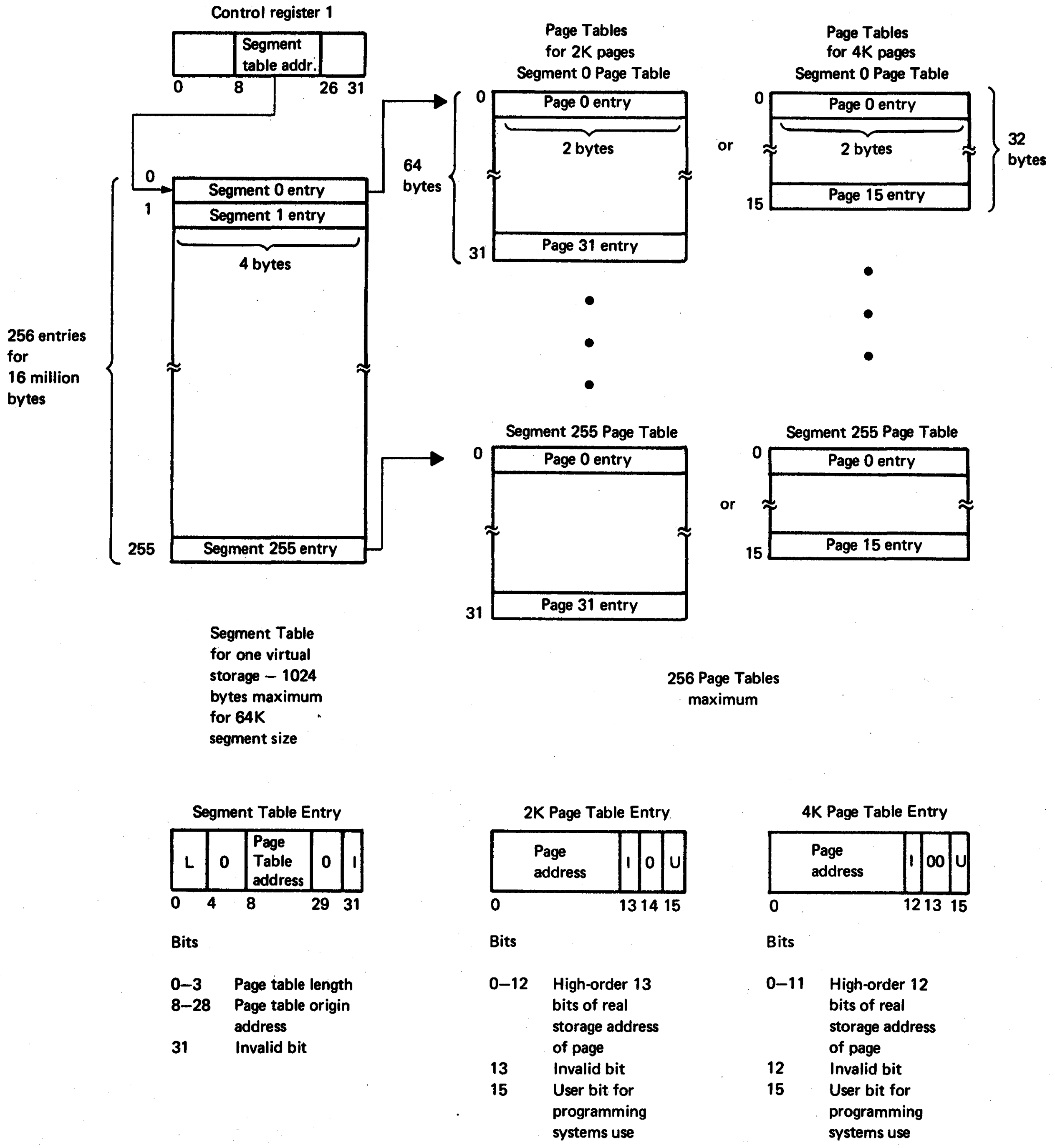
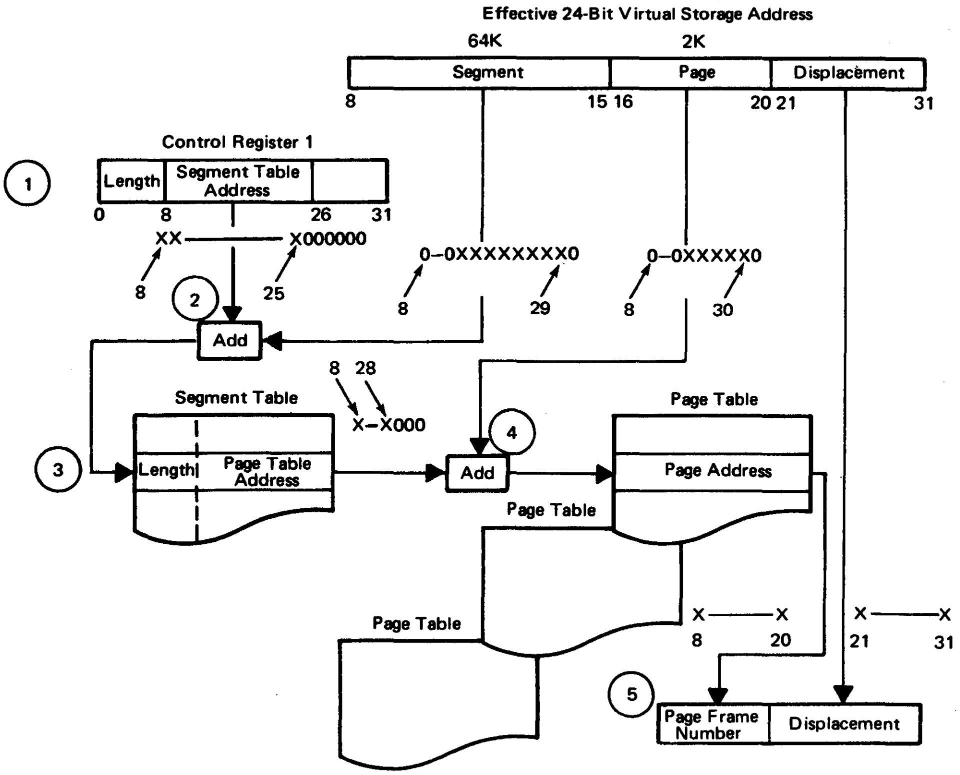


Figure 30.10.2. Segment table and page tables used for dynamic address translation



1. Bits 8, 9, 11, and 12 in control register 0 are checked for validity. A translation specification interruption occurs if an invalid setting is present. Segment address bits from the virtual storage address are checked using length bits in control register 1. If the segment entry address is outside the segment table, a segment translation exception is indicated.

2. Six low-order zeros are appended to the segment table address in control register 1. Two low-order zeros are appended to the segment bits from the virtual storage address. The two values are added to obtain a segment table entry. If the invalid bit is on in this entry, a segment translation exception is indicated.

3. Page address bits from the virtual storage address are checked using page table length bits contained in the segment table entry. A page translation exception is indicated if the entry addressed is outside the page table.

4. Three low-order zeros are appended to the page table address contained in the segment entry. One low-order zero is appended to the page address from the virtual storage address. The two values are added to obtain a page table entry. If the invalid bit is on in this entry, a page translation exception is indicated.

5. The 24-bit real storage address is formed using the 12 or 13 high-order bits from the page table entry and the 12 or 11 low-order bits from the virtual storage address.

Figure 30.10.3. Dynamic address translation procedure

All the entries in the TLB are invalidated (identification codes set to zero) when a reset occurs, the operator enters a storage configuration via the configuration panel, or retry recovery is attempted after a machine check occurs. When a SET STORAGE KEY is issued and valid translated addresses are in the TLB, the TLB is searched and each entry is invalidated that has the same real address as the one for which the key is being set. The PURGE TLB instruction is provided to enable a program to invalidate all 128 TLB entries. In general, this instruction must be issued when an entry in a page table is invalidated, since the real storage address being invalidated could be contained in the TLB. The TLB will be purged by the virtual storage operating systems as required.

A change in segment table origin address, segment size, or page size can also affect the validity of current TLB entries. In order to reduce the number of full TLB purges required by such changes, a segment table origin address register stack (STO-stack) is implemented. The STO-stack can contain the address of six different segment tables at a time. Each segment table could define a different virtual storage. An STO-stack entry also indicates the segment and page size in effect for the virtual storage associated with the segment table address.

The six entries in the STO-stack have a unique identification number associated with them. One of these numbers is denoted to be the currently active identification number. Whenever a segment table address is placed in control register 1, the segment table address is also placed in the STO-stack, if it is not already there, and the identification number the segment table address is assigned becomes the new active identification number.

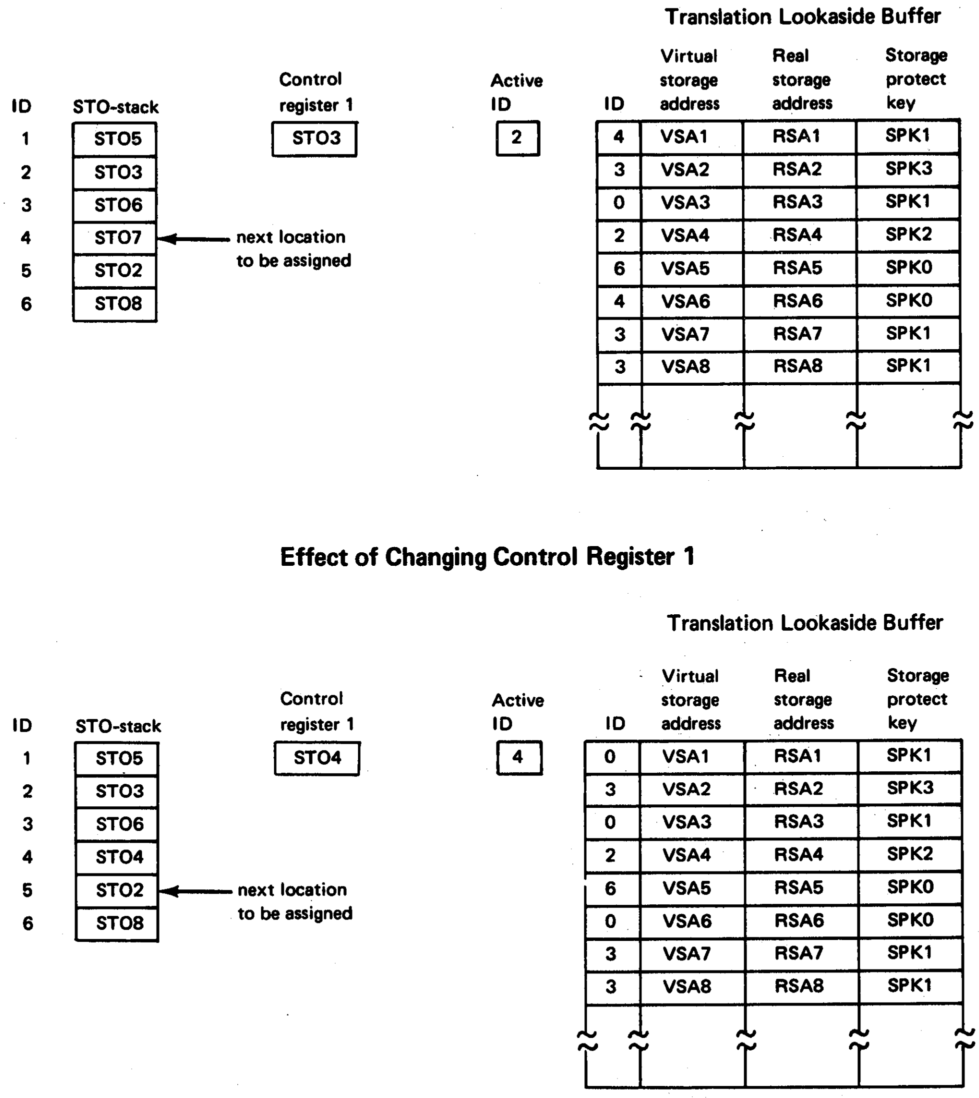
An STO-stack identification number is stored with each TLB entry to identify the segment table, and thereby the virtual storage, with which the TLB entry is associated. When the TLB is interrogated to see whether it contains the required translation, the STO-stack identification number of the TLB entry is compared with the active identification number. If the identifications are equal, the TLB location contains a translation from the virtual storage associated with the active identification number. If the identifications are not equal, the TLB location contains a translation for a different virtual storage and, therefore, the TLB entry does not contain the required translation even though it may contain a virtual storage address equal to the one that is to be translated.

When DAT mode is entered or a LOAD CONTROL instruction is issued when DAT mode is operative, the segment table address in control register 1 and page and segment size specifications from control register 0 are compared with each of the STO-stack locations to determine whether a change in these specifications is being made. If a change is indicated, some TLB purging may be required.

An equal comparison between an STO-stack entry and the segment table address, segment size, and page size in control registers 0 and 1 indicates that the virtual storage associated with the segment table address now in control register 1 is currently one of the six virtual storages whose translations are being maintained in the TLB and that segment and page size have not been changed. The STO-stack identification number of the segment table address now in control register 1 is designated to be the active identification. No TLB purging is required.

No equal comparison between an STO-stack entry and the segment table address, segment size, and page size in control registers 0 and 1 indicates that translations for the segment table now indicated by control register 1 are not currently being maintained in the TLB or that segment or page size is being changed. The new segment table address is placed in the STO-stack, and the STO-stack identification number assigned becomes the active identification.

A first-in, first-out algorithm is used to determine which STO-stack location to assign. If the new address displaces another segment table address, the TLB entries associated with the displaced segment table (and virtual storage) must be purged. This is done by setting the identification number to zero for each entry in the TLB that has the same STO-stack identification number as the segment table address that was displaced. This identification number is now assigned to the newly stored segment table address. The other TLB entries need not be invalidated. See Figure 30.10.4 for an example of TLB purging when control register 1 is changed.



ID	STO-stack
1	STO5
2	STO3
3	STO6
4	STO7
5	STO2
6	STO8

ID	Virtual storage address	Real storage address	Storage protect key
4	VSA1	RSA1	SPK1
3	VSA2	RSA2	SPK3
0	VSA3	RSA3	SPK1
2	VSA4	RSA4	SPK2
6	VSA5	RSA5	SPK0
4	VSA6	RSA6	SPK0
3	VSA7	RSA7	SPK1
3	VSA8	RSA8	SPK1

Effect of Changing Control Register 1

ID	STO-stack
1	STO5
2	STO3
3	STO6
4	STO4
5	STO2
6	STO8

ID	Virtual storage address	Real storage address	Storage protect key
0	VSA1	RSA1	SPK1
3	VSA2	RSA2	SPK3
0	VSA3	RSA3	SPK1
2	VSA4	RSA4	SPK2
6	VSA5	RSA5	SPK0
0	VSA6	RSA6	SPK0
3	VSA7	RSA7	SPK1
3	VSA8	RSA8	SPK1

Figure 30.10.4. TLB purging when control register 1 is changed

Implementation of the STO-stack in the Model 168 enables a control program that supports multiple virtual storages (such as VM/370) to alter control registers 0 and 1 in order to change the virtual storage for which address translation is effective, without automatically causing purging of the entire TLB. The STO-stack facility will also be of benefit in an OS/VS2 environment, since OS/VS2 SVS supports two

segment tables to provide fetch protection for all regions (see
| OS/Virtual Storage 2 Single Virtual Storage (SVS) Features Supplement).

Addresses Translated

All storage addresses that are explicitly designated by a program and that are used by the CPU to refer to instructions or data in processor storage are virtual storage addresses and are subject to address translation. Thus, when DAT is operative, the starting and ending storage addresses used with the program event recording feature are virtual, as are the storage addresses stored in PSW's during interruptions. Address translation is not applied to addresses that explicitly designate protect key storage locations or to quantities that are formed as storage addresses from the values designated in the base and displacement fields of an instruction but that are not used to address processor storage (shift instructions, for example). In addition, address translation is not applied to the storage addresses in CCW lists used for I/O operations.

Some of the storage addresses supplied to a program by the CPU are virtual and some are real. Table 30.10.2 lists, for the Model 168, those storage addresses designated by a program, either explicitly or implicitly, that are virtual (and, therefore, are subject to translation) and those addresses that are real or not used to reference processor storage and, thus, are not translated. The table also indicates which storage addresses supplied to a program are virtual and which are real.

FEATURES TO SUPPORT DEMAND PAGING

Reference and Change Recording Facility for Real Storage Blocks

A hardware recording facility is standard in the Model 168. This facility provides continuous recording of the activity of all 2K real storage blocks via reference and change bits. The settings of these recording bits can be used by control program routines to support a demand paging environment. This hardware facility is always active; it does not depend on EC or translation mode being operative.

The seven-bit key associated with each 2K real storage block in the Model 168 has four storage-protect bits, one fetch-protect bit, one reference bit, and one change bit. During system operation, the activity of each 2K real storage block is monitored by hardware. Whenever a fetch is made either by a CPU or a channel to a real storage address, the reference bit in the key associated with the 2K storage block that contains that real storage address is turned on by the hardware. A store into any real storage address causes the hardware to turn on both the change bit and the reference bit for the affected 2K block.

Store/display operations initiated from the 3066 console also cause appropriate changing of the reference and change bits. The RESET REFERENCE BIT instruction is provided to allow the reference bit of any 2K real storage block to be reset by programming without altering the contents of the other six bits in the protect key. A CPU fetch that is satisfied with data contained in the buffer does not cause reference recording in the Model 168. There are situations, however, in which instruction or operand prefetching may cause the reference bit for a page frame to be turned on even though the contents of that page are never used.

Table 30.10.2. Virtual and real storage addresses used by and supplied to programs in the Model 168

<u>Virtual Storage Addresses Explicitly Designated by the Program</u> (translated)

- Instruction address in the PSW
- Branch addresses in instructions
- Addresses of operands in instructions
- Operand address in the LOAD REAL ADDRESS instruction
- PER starting address in control register 10 and PER ending address in control register 11

<u>Real Storage Addresses Explicitly Designated by the Program</u> (not translated)

- Operand addresses in SET STORAGE KEY, INSERT STORAGE KEY, and RESET REFERENCE BIT instructions
- Machine check extended log pointer in control register 15
- I/O extended log pointer in location 172
- Segment-table-origin address in control register 1
- Page-table-origin address in a segment table entry
- Page frame address in a page table entry
- CCW address in the channel address word (CAW)
- Address in a CCW specifying a data area or the location of another CCW
- Data address in channel indirect data address lists

<u>Addresses Not Used to Address Storage</u> (not translated)

- Operand addresses specifying the amount of shift in fixed-point, logical, or decimal shift instructions
- Operand address in LOAD ADDRESS and MONITOR CALL instructions
- I/O addresses in I/O instructions and in the Input/Output Communication Area (IOCA)

<u>Real Storage Addresses Used Implicitly</u> (not translated)

- Addresses of PSW's used during an interruption and in executing the programmed or manually initiated restart function
- Address used by the CPU to update the timer at location 80
- Address of the CAW, the CSW, and the I/O address within the IOCA used during an I/O interruption or during execution of an I/O instruction, including execution of STORE CHANNEL ID
- Addresses used for the store status function

<u>Virtual Storage Addresses Provided to the Program</u>

- Address stored in the instruction address field of the old PSW during an interruption
- Address stored by a BRANCH AND LINK instruction
- Address stored in register 1 by TRANSLATE AND TEST and EDIT AND MARK instructions
- Address stored in location 144 on a program interruption for a page translation or segment translation exception
- Address stored in location 152 on a PER interruption

<u>Real Storage Addresses Provided to the Program</u>

- The translated address generated by the LOAD REAL ADDRESS instruction
- Address of the segment table entry or page table entry provided by the LOAD REAL ADDRESS instruction
- Failing storage address in location 248
- CCW address in the CSW

The hardware reference and change recording facility is used by the page replacement algorithm of a virtual storage operating system. When a page is loaded into a page frame, the reference and change bits for that page frame are set to zero. (When a 4K page size is used, the reference and change bits for both of the 2K storage blocks involved are reset.) Thereafter, the reference bit is used to determine the activity of a page. The change bit is inspected to determine whether a page must be paged out when its page frame is reassigned. The SET STORAGE KEY instruction must be used to reset the change bit.

Instruction Nullification

When a page fault occurs in a demand paging environment, execution of the instruction that caused the page fault stops and the control program gains control to initiate a page-in operation. When the contents of the missing page have been loaded (and the appropriate page table entry has been updated), the instruction that caused the page fault is reissued. For the instruction to operate correctly the second time, execution of the instruction must have been stopped so that reexecution gives the same results as would have occurred if the instruction had been executed only once. Therefore, the contents of real storage, the general and floating-point registers, and the PSW must not be altered.

The execution of an instruction is said to be nullified when it is stopped in such a way that no operation was performed, no fields were changed, and the PSW indicates the address of the instruction that was stopped. Interruptible instructions, such as MOVE LONG, are divided into execution units. One or more execution units may have completed before a page fault is detected. In this case, only the current execution unit is nullified.

Various methods are used, depending on the type of instruction, to determine the need for nullification. In some cases, execution is attempted where hardware detection of page faults permits nullification. In other cases, pretesting is required to determine whether the virtual storage pages to be referenced have page frames allocated. Nullification testing is required only for instructions whose translated addresses reference storage. In the Model 168, testing is performed by instruction unit hardware and/or additional microcode routines that are executed before normal instruction execution. However, for some instructions, prefetching of the data accomplishes pretesting, so that no additional pretesting cycles are required. A LOAD instruction that addresses a word on a fullword boundary is an example of such an instruction.

Similarly, if a store fullword instruction addresses a four-byte field that is not on a fullword boundary, a pretest is required to determine whether all four bytes are contained in real storage. The pretest microcode for this instruction issues a fetch to the highest addressed byte in the four-byte data field (virtual storage address in the instruction plus 3). The absence of a page translation exception during translation of the virtual storage address indicates that (1) if the data field spans two pages, at least the second of the two pages is present in real storage or (2) the data field is totally contained in one page, which is present in real storage. Hence the instruction is allowed to proceed without nullification. If the data field actually does span two pages and the first page is not present in real storage, this fact will be indicated by a page fault during translation of the address of the high-order byte of the field. Instruction nullification will occur and the page fault will cause a page-in of the first page to be initiated by the control program as usual.

If the pretest fetch operation does cause a translation exception, the store fullword instruction is nullified and the control program gains CPU control to load the missing page. Once again, the page-in caused by the pretest may have brought in the second of two pages spanned by the data field or the only page containing the data field. After the page-in, the instruction is reexecuted.

CHANNEL INDIRECT DATA ADDRESSING

Since address translation is not performed by the channels for programs that operate in paged mode, address translation must be performed on CCW lists by programming before the initiation of START I/O instructions. Such address translation need not be performed on the CCW lists in programs that operate in nonpaged mode.

In addition, a contiguously addressed I/O area in virtual storage can span a set of noncontiguous page frames. Hence, a method of handling a noncontiguously addressed I/O area in real storage during the operation of a CCW list is required. The channel indirect data addressing feature is used to provide this capability. As is shown in Figure 30.10.5, the use of channel indirect data addressing allows the channel program logic used in the CCW list with virtual storage addresses to be maintained in the new CCW list that contains real storage addresses.

When channel indirect data addressing is present, bit 37 of a CCW is designated as the indirect data address (IDA) flag. The IDA flag applies to read, read backward, write, control, and sense commands and is valid in both BC and EC modes. When the IDA flag in a CCW is zero, bits 8 to 31 of the CCW specify the real storage address of the beginning of the I/O area as usual. When the I/O area referenced by a CCW is completely contained in one page, an indirect data address list (IDAL) is not required and the IDA flag is set to zero. When the IDA flag is one, CCW bits 8 to 31 specify the real storage address of an IDAL instead of an I/O area. When the I/O area referenced by a CCW spans two or more pages, an IDAL is required and the IDA flag is set to one.

An IDAL consists of two or more contiguous indirect data address words (IDAWs) of four bytes each. There is one IDAW in an IDAL for each 2K storage block spanned by the I/O area. An IDAW, which must be aligned on a fullword boundary, contains a real storage I/O area address in bits 8 to 31. Bits 0 to 7 must be zero. The first IDAW in the list points to the beginning of the I/O area to be used by the CCW and is obtained by translating the virtual storage address contained in the original CCW. Any valid real storage address can be specified in the first IDAW of a list. All IDAWs after the first must address the beginning (or end for a read backward operation) of a 2048-byte block located on a 2048-byte boundary, or a program check occurs. That is, bits 21-31 of the address in the IDAW must be zeros (or ones for a read backward).

Figure 30.10.5 shows an example of the IDALs required for a command-chained CCW list when 2K pages are used. The IBM-supplied virtual storage operating systems construct a new CCW list with translated addresses that is used to control the I/O operation. The new CCW list points to any required IDALs.

When a START I/O instruction is executed, the channel fetches the first CCW in the list, pointed to by the channel address word (CAW), and inspects bit 37. If it is zero, the operation is started in the I/O area specified by the real storage address in the CCW. If bit 37 is a one, the first IDAW is fetched from the real storage address in the CCW. The I/O operation is begun using the real storage address in the first IDAW and, assuming that the I/O operation is not a read backward, ascending real storage addresses in the I/O area are used by the channel until a 2048-byte boundary is reached.

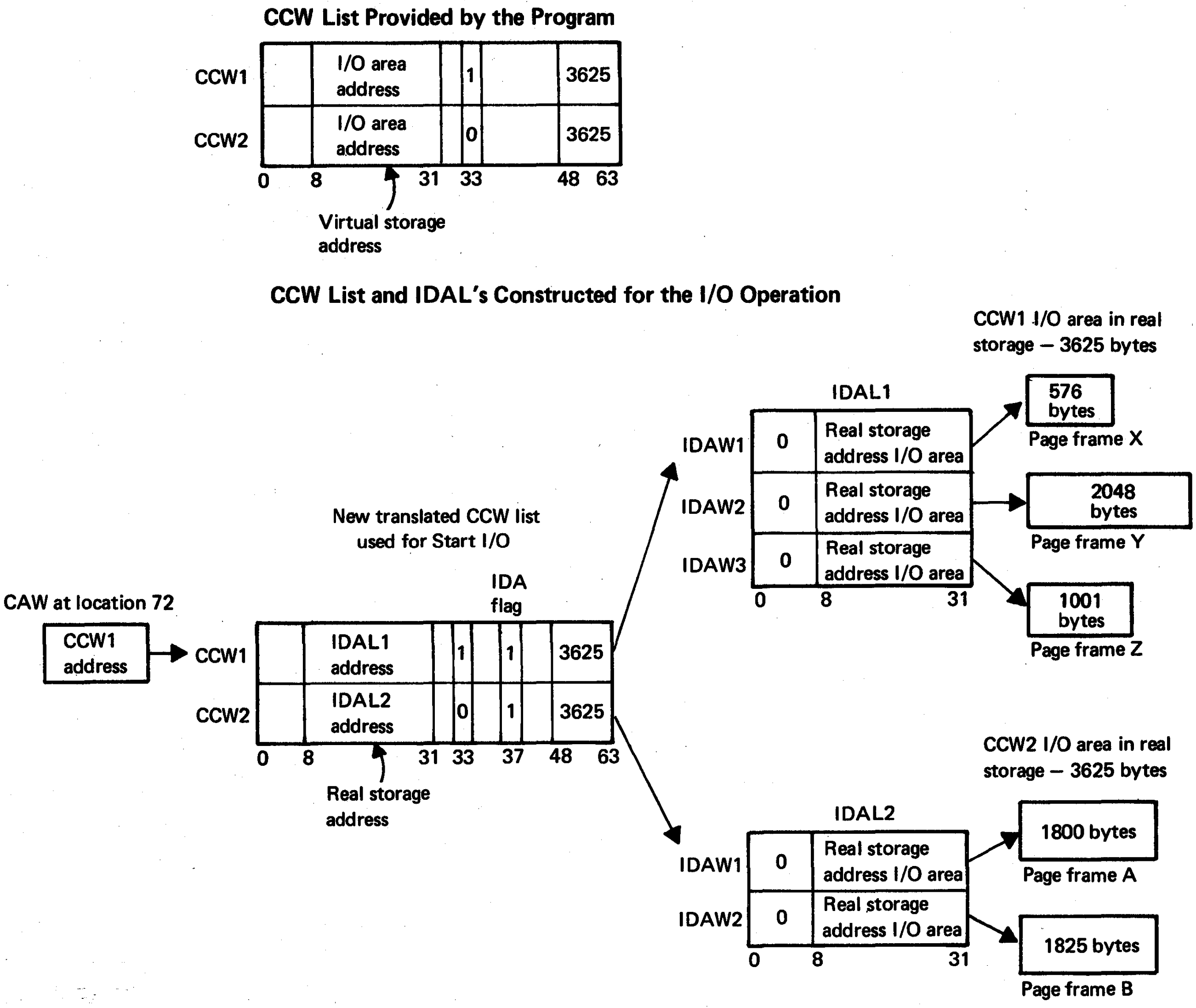

CCW List Provided by the Program
CCW1
I/O area address
1
3625
CCW2
I/O area address
0
3625
0
8
31
33
48
63
Virtual storage address
CCW List and IDAL's Constructed for the I/O Operation
CCW1 I/O area in real storage — 3625 bytes
IDAL1
IDAW1
0
Real storage address I/O area
IDAW2
0
Real storage address I/O area
IDAW3
0
Real storage address I/O area
0
8
31
576 bytes
Page frame X
2048 bytes
Page frame Y
1001 bytes
Page frame Z
New translated CCW list used for Start I/O
IDA flag
CAW at location 72
CCW1 address
CCW1
IDAL1 address
1
1
3625
CCW2
IDAL2 address
0
1
3625
0
8
31
33
37
48
63
Real storage address
CCW2 I/O area in real storage — 3625 bytes
IDAL2
IDAW1
0
Real storage address I/O area
IDAW2
0
Real storage address I/O area
0
8
31
1800 bytes
Page frame A
1825 bytes
Page frame B

Figure 30.10.5. Example of IDALs required for a CCW list when page size is 2K

The channel detects a 2K boundary by monitoring I/O area address bits 21-31. When these bits change from all ones to all zeros, the first byte of the next 2K real storage block is indicated. At this point, the channel accesses the second IDAW in the list to obtain the next real storage I/O area address to be used, and the data transfer operation continues. The channel continues using the IDAL until the operation indicated by the CCW completes (CCW count reaches zero, interrecord gap on tape reached, etc.). The next CCW is accessed if command or data chaining is indicated. Bit 37 is inspected and the I/O operation continues as described until the CCW list is exhausted.

When a program operates in paged mode, the CCW list for an I/O operation must be inspected and the appropriate IDALs must be constructed prior to issuing a START I/O instruction. At the completion of the I/O operation, some retranslation is also required. In general, the following steps must be taken for each CCW in a given list:

1. Determine whether the I/O area referred to in the CCW spans pages or is contained in only one. If a single page is involved,

translate the virtual storage address to real and store it in the CCW. Ensure that a page frame is allocated to the page containing the buffer and that the page frame is marked fixed.

2. If two or more pages are involved, set up the required number of IDAWs, place a pointer to the IDAL in the CCW, and turn on CCW bit 37.

3. While setting up IDAWs, determine whether all pages in the I/O area have real storage allocated. If not, ensure that page frames are allocated and fixed.

At the completion of the I/O operation, the real storage address in the channel status word must be translated to a virtual storage address, and the pages that were short-term fixed prior to initiation of the I/O operation must be unfixed. Channel program translation and page fixing are performed by the I/O control portion of the control program in IBM-supplied virtual storage operating system support. A program that contains a CCW list that is dynamically modified during its execution cannot operate correctly in paged mode, since the modification is made to the CCW list with virtual storage addresses rather than to the translated CCW list that is actually controlling the I/O operation on the channel.

30:15 SYSTEM PERFORMANCE IN A VIRTUAL STORAGE ENVIRONMENT

A virtual storage environment is designed to provide new data processing capabilities. As is true for any other capability offered by an operating system, support of a new function requires control program use of a certain amount of the hardware resources of the system. In this respect, virtual storage is no different from multiprogramming and the many other new capabilities that have continuously been added to OS since its initial release.

The characteristic that makes virtual storage different from most other features is that virtual storage is not primarily designed to improve system performance, as are many other control program facilities. Virtual storage is first a functional tool and, in certain cases, can also be a performance tool. The objectives of OS virtual storage operating systems are to (1) provide new functions, (2) maintain upward compatibility with OS nonvirtual storage environments, and (3) provide performance equal to or better than that achieved with a nonvirtual storage operating system using the same system configuration. Attainment of the last objective will not be possible for all existing System/370 configurations.

In addition, some of the new functions a virtual storage environment provides cannot be achieved in a nonvirtual storage environment or are not practical, and in these cases, performance is not the primary consideration when using the facility virtual storage offers. As the cost of hardware resources continues to decline on a unit cost basis (cost per processor storage bit, cost per direct access bit, etc.), it becomes increasingly more economical to use system resources to perform functions that otherwise are handled by installation personnel.

The other new characteristic of virtual storage is that it enables a given system configuration to provide a wider range of performance, as well as function, as a result of the new factors that affect operation of a system with virtual storage support. Thus, a slightly different approach must be taken in planning for and in evaluating system performance in a virtual storage environment.

Many of the same factors that affect system performance in an OS/VS1 or OS/VS2 environment are the same as those that apply to OS MFT or OS

MVT, respectively. First, the system configuration must include the hardware resources (CPU speed, channels, I/O devices, real storage) required for the control program and job mix. This subsection identifies the system resources specifically required to support a virtual storage environment. Second, the system should be designed to balance resource usage to achieve optimum throughput, and to use applicable performance and control program design options the particular operating system offers, taking into account the characteristics of the installation job stream.

The performance of a system in a virtual storage environment is also affected by certain new factors that do not apply to systems without virtual storage support. This subsection identifies these new factors, explains how they generally affect system performance, and indicates steps that can be taken to increase and maximize system performance when a virtual storage operating system is used.

This discussion applies to OS/VS1 and OS/VS2, and is restricted to performance factors that are common to the virtual storage environments they support. The virtual storage operating systems also offer new performance-oriented enhancements that are not related to the implementation of virtual storage. These unique performance features are discussed in the optional programming systems supplements.

The performance information in this subsection is designed to present concepts and considerations for a virtual storage environment. Figures and graphs are used for illustrative purposes. They do not represent any particular installation or measured results. Their purpose is to illustrate the interrelated factors of multiprogramming performance in a virtual storage environment. The performance information presented is conceptual. It is based on the experience and judgment of IBM individuals with performance knowledge and on performance measurements made during development of OS/VS1 and OS/VS2. Therefore, it may not apply to all installations.

SYSTEM RESOURCES REQUIRED TO SUPPORT A VIRTUAL STORAGE ENVIRONMENT

In order to support a demand-paged virtual storage environment using System/370, in which programs are operating in paged mode, additional system resources are used by the IBM-supplied virtual storage operating systems, as follows:

- Dynamic address translation hardware requires CPU time to perform virtual-storage-to-real-storage address translation. The amount of time required is determined by the System/370 model and the number of times the full table-lookup translation procedure must be performed. The Model 168, for example, has a translation lookaside buffer that is designed to reduce use of the full table-lookup translation procedure. The CPU time required is also affected by program structure (which is discussed later). A small amount of additional CPU time is also required to pretest certain instructions that reference storage, as discussed under "Instruction Nullification" in Section 30:10. Studies have shown that a relatively small percentage of the total CPU time specifically required to support a virtual storage environment is devoted to address translation by DAT hardware.

- CPU time is required to translate the virtual storage addresses in channel programs (CCW lists) into real storage addresses, build indirect data address lists (where necessary), and short-term fix pages that will be referenced during I/O initiation, execution, and interruption handling. Channel program translation and page fixing are performed prior to the initiation of each I/O operation with a channel program that contains virtual storage addresses. Channel status word retranslation and page unfixing is performed at the completion of these I/O operations. The amount of CPU time this function requires per data set is affected by the number of I/O requests (EXCP macros) issued, the number of CCW's in the channel programs started, the number of pages that must be fixed, and whether or not indirect data address lists have to be constructed. Studies have shown that a large portion of the total CPU time specifically required to support a virtual storage environment is used to perform channel program translation and page fixing.

- CPU time is required to process page faults and for the execution of other control program code that is specifically required to support a virtual storage environment. CPU time is required for such things as servicing additional program interruptions, managing and allocating real and external page storage, maintaining tables used by DAT hardware, and testing for paged or nonpaged mode of program operation.

- I/O time is required for paging operations. The amount of paging I/O time required is related to the number of page faults that occur and the speed of the paging I/O device(s) used. In OS/VS2 environments, the total I/O time required for paging includes some I/O time that is also required in OS MVT environments to load transient control program routines.

- Direct access storage is required for external page storage. The amount required depends on the amount of virtual storage that is to be supported and the way in which the particular operating system organizes and manages external page storage. (See the optional programming systems supplements for external page requirements by device type.)

- The amount of real storage required by the resident (fixed) control program is increased by the amount of real storage needed for additional routines and code that are included specifically to support a demand paged virtual storage environment.

The effect this additional use of hardware resources has on the performance of a given system configuration depends on the resource requirements of the job stream and the current utilization of system resources. To the degree that the additional required CPU and I/O time can be overlapped with existing CPU and I/O time that is currently not overlapped, system throughput is not affected. System throughput will be affected by the increase in CPU and I/O time that cannot be overlapped.

When a virtual storage operating system is used with an existing system configuration, for example, and the same job stream is processed, performance is affected by the use of any new performance enhancements these operating systems provide as well as by an increase in resource utilization that is required to support a virtual storage environment. When a Model 168 replaces a Model 165, performance is also affected by the fact that the Model 168 has a faster internal performance than the Model 165.

Figure 30.15.1 conceptually illustrates possible system performance when a virtual storage operating system is installed on a Model 168 with the same amount of real storage and the same I/O device configuration as the replaced Model 165.

Panel 1

Sample existing CPU and I/O utilization and overlap for a Model 165.

EXISTING SYSTEM THROUGHPUT MAINTAINED

Panel 2

Some of the additional CPU and I/O time required is overlapped with previously unoverlapped I/O and CPU time (points A). Additional CPU and I/O time that cannot be overlapped (point B) is offset by a reduction in the amount of CPU and I/O time required to process the same job stream. Results are achieved in the same elasped time.

Panel 3

Additional CPU and I/O time required (dotted lines) is overlapped and offset by operating the system at a higher level of multiprogramming to achieve greater overlap. Results are achieved in the same elapsed time.

EXISTING SYSTEM THROUGHPUT IMPROVED

Panel 4

Unoverlapped CPU and I/O time required is exceeded by reductions in previously used CPU and I/O time. Better overlap of previously used CPU and I/O time is also achieved. Same results are achieved in less elapsed time.

Panel 5

A higher level of multiprogramming is used to perform more work and achieve better overlap of CPU and I/O time. More results are achieved in the same elasped time.

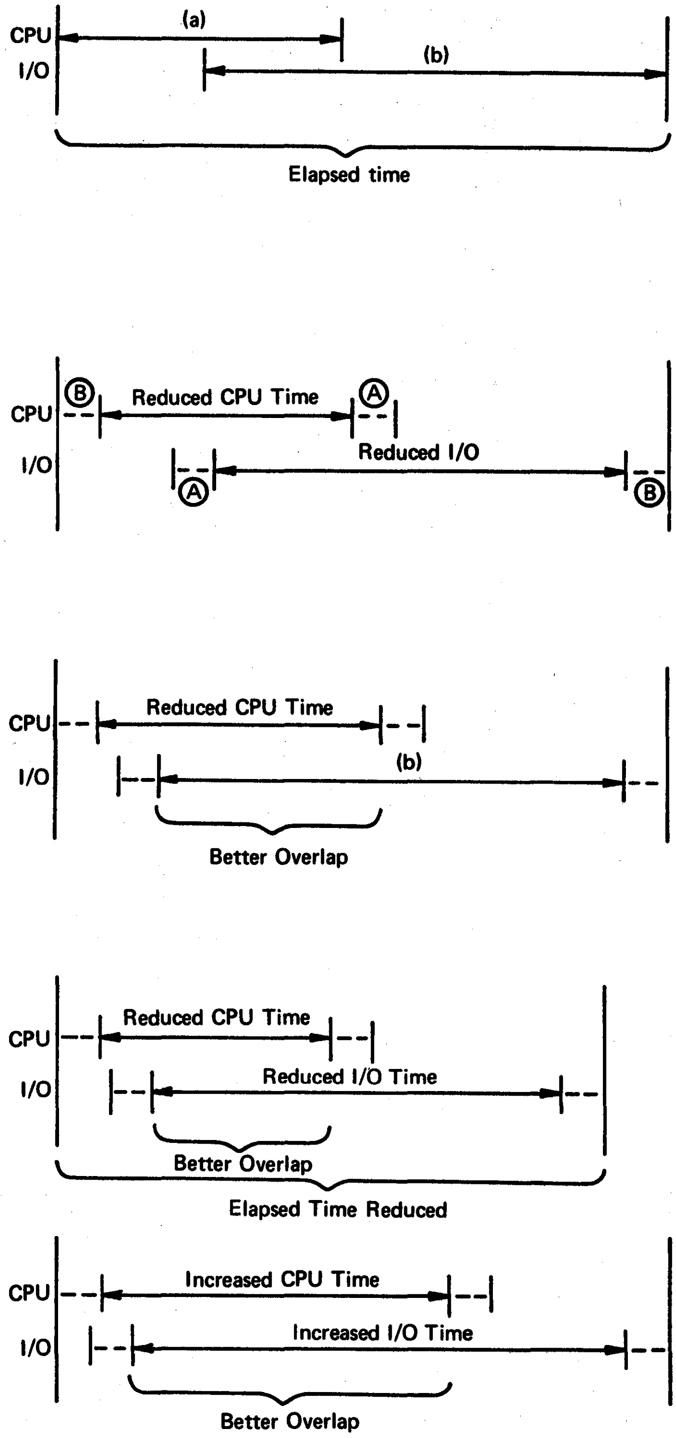


Figure 30.15.1. Possible system performance when a virtual storage operating system is used with a Model 168 with the same I/O configuration and real storage size as the replaced Model 165

A sample throughput for a Model 165 is shown in panel 1. (It is not meant to represent any specific Model 165 throughput.) Panels 2 and 3 illustrate the conditions under which existing performance can be maintained and the last two illustrate the conditions under which existing performance can be improved.

Existing throughput is maintained if both of the following occur:

1. A portion of the additional CPU and I/O time required to support a virtual storage environment is overlapped with CPU and I/O time that previously was not overlapped, as shown by points A in panel 2.

2. The amount of additional CPU and I/O time that cannot be overlapped (shown by points B in panel 2) is offset by reductions in previously used CPU and I/O time that occur as a result of the faster internal performance of the Model 168 and use of new performance features of the virtual storage operating system, as shown in panel 2. The unoverlapped CPU and I/O time may also be offset by a combination of the faster internal performance of the Model 168 and the achievement of better overlap as a result of operating the system at a higher level of multiprogramming to process the same work (as shown in panel 3).

Existing system throughput can improve if (1) unoverlapped CPU and I/O time required to support a virtual storage environment is exceeded by reductions in previously used CPU and I/O time and/or if previously used CPU and I/O time are better overlapped (as shown in panel 4) or (2) a higher level of multiprogramming is used to perform more work and provide better CPU and I/O overlap in the same elapsed time (as shown in panel 5).

NEW FACTORS THAT AFFECT SYSTEM PERFORMANCE

In addition to the factors that affect system performance in a nonvirtual storage environment, the performance of a system in a virtual storage environment is affected by the relationship of the following factors: the speed and number of paging devices, the speed of the CPU, the size of real storage, the structure of the programs in the job stream, and the way in which real storage is organized and allocated by the virtual storage operating system. The interrelationship of each of these factors and their individual effect on performance, except for the last factor listed, are as follows (page replacement algorithms are not discussed):

Speed and Number of Paging Devices. A certain amount of I/O time is required to read in (or write out) a page using a given direct access device type. This time is a function of device type characteristics--seek time, rotation time, and data transfer rate. Assuming one page-in performed at a time, no page-outs, and no contention for the paging device or its channel, a maximum paging rate, in terms of the number of page faults that can be serviced per time interval, could be calculated for a given device type. This rate could be improved by certain programming techniques, such as use of rotational position sensing when it is present and initiation of multiple page-in and page-out requests with a single channel program. (Various techniques are implemented in OS/VS1 and OS/VS2.) The maximum paging capability of a given system can be increased by various means, such as using more than one paging device or using a faster paging device.

The paging characteristic of a virtual storage environment is the feature that permits an operating system to support virtual storage that is larger than real storage. The paging activity of a system begins to adversely affect system performance, however, once the CPU is in the position of frequently having to wait for paging I/O operations to complete. When requests for paging operations are permitted to occur faster than the paging rate the system can sustain, such that the system can do little or no processing except that related to paging, the system is in a paging-I/O-bound situation and is said to be thrashing. When a thrashing condition exists, little or no productive work can be accomplished unless paging activity is reduced.

In order to prevent thrashing, the System/370 virtual storage operating systems monitor the activity of the system to determine when paging activity becomes excessive. At this point, the OS control program performs task deactivation. This involves placing a task (OS/VS2) or partition (OS/VS1) in deactivated status and releasing the page frames currently allocated to the task or partition. These page frames are then available for allocation to other tasks to reduce paging activity. Later, when paging activity becomes sufficiently low, the deactivated task or partition is reactivated.

CPU Speed. An improperly balanced relationship between CPU speed and paging device speed can also cause the system to become I/O-bound as a result of paging. A Model 168 can execute a certain number of instructions during the time required to service a page-in request using a given direct access device type. A Model 168 can execute many more instructions during a page-in from a 2305 Model 2, for example, than can a Model 158. As long as there is useful work for the CPU to perform while paging operations occur, the system is not kept waiting for paging I/O. However, if the concurrently operating programs are constantly executing instructions faster than the pages they require can be brought into real storage, an excessively high paging rate can develop and task deactivation will be the result. In general, therefore, the larger-scale System/370 models require faster paging devices to handle a particular page fault rate than do the smaller-scale models.

Real Storage Size. The amount of real storage present in a system affects the number of page faults that occur when a given job stream is processed. If the amount of real storage present in the system is equal to the total amount of virtual storage being used by the concurrently executing tasks, no page faults occur for programs that have been fetched and initiated. When the amount of real storage present is less than the amount of virtual storage being used, page faults occur. The total number of page faults that occur for a given job stream is affected by the ratio of virtual storage used to real storage available.

Assuming the amount of virtual storage used in a given system remains the same, the virtual-to-real storage ratio can vary. This occurs while a given system experiences variations in the amount of real storage actually available for paging as the amount of fixed real storage changes during job stream processing. The real storage available for paging at any point in time is the difference between the amount of real storage in the system and the total amount of long- and short-term fixed real storage. For IBM-supplied virtual storage operating systems, the total amount of fixed real storage at any given time is the sum of the:

- Resident (fixed) control program size, which does not vary after IPL

- Amount of long-term fixed real storage required for control blocks, which can change as the level of multiprogramming changes in OS/VS1 and OS/VS2 environments

- Amount of short-term fixed real storage required for outstanding I/O operations that have virtual channel programs, which fluctuates with the I/O activity of the system

- Amount of long-term fixed real storage required by the job steps executing in nonpaged mode, if any

- Amount of long-term fixed real storage required by programs that operate in paged mode but that have a portion of their partition or region always fixed (TCAM in OS/VS1 and OS/VS2, for example)

As the virtual-to-real storage ratio of a job stream increases, so usually does the page fault rate. In general, the page fault rate increases slowly for a while. At some point, the increase in page faults begins rising rapidly as the virtual-to-real storage ratio continues to increase. Figure 30.15.2, shown later, illustrates the general relationship between the number of page faults and the virtual-to-real storage ratio.

The amount of real storage available to process a given job stream also varies when a given job stream is processed on systems with various amounts of real storage, such as when a smaller-scale system is used to back up a larger-scale system.

The degree to which reducing the real storage available for paging affects the page fault rate depends on the paging activity pattern of the programs in a job stream. Therefore, the virtual-to-real storage ratio at the point at which a given number of page faults occurs will usually vary by job stream. The point can also be different for systems with similar paging activity patterns and the same amount of real storage installed, but with different amounts of long-term fixed real storage.

As the virtual-to-real storage ratio increases because of a reduction in the real storage available (or an increase in the amount of virtual storage used) and the page fault rate increases, more demand is placed on the paging devices. If too small an amount of real storage is present in a system, this situation can cause the page fault rate to exceed the permissible rate and task deactivation will occur. In general, therefore, in order to obtain a certain level of performance, a configuration that supports a given job stream and virtual storage size may require more real storage when a relatively slower paging device is used than when a faster paging device is used.

Program Structure. The total amount of virtual storage a program uses is not nearly so significant a factor in system performance as the way in which virtual storage is used. That is, the pattern and frequency of reference to pages in a program have more effect on the number of page faults that occur than the total size of the program. For example, assume a case in which a program has a 100K virtual storage design point. If the program can be structured to execute as a series of logical phases of four or five pages each and the pages of each logical phase reference only each other, no more than four or five page frames (8K to 10K or 16K to 20K of real storage, depending on page size) need be dynamically available to the program at one time and paging activity occurs only as the program progresses from one logical phase to the next. However, assume the program is structured so that during its execution each page of instructions constantly references a large number of different pages of instructions and data for very short durations on a random basis. An excessively high paging rate could occur if only four or five page frames were dynamically available to such a program at any time.

As indicated previously, most types of programs have a natural locality of reference characteristic, so that they can be structured to operate as a series of logical phases. In the simplest case, for example, a program can logically consist of an initialization phase, a main phase, one or more exception handling phases, and a termination phase. The total amount of virtual storage referenced in each logical phase usually varies but, generally, the amount is less than the total size of the program. In addition, the pages that are part of (referenced in) a given logical phase can usually be described as active or passive.

For the purpose of the discussion in this subsection, an active page is defined as one with a high probability of being referenced multiple times during execution of the logical phase, while a passive page has a low probability of being referenced more than once during execution of the phase. A logical phase experiences the least amount of paging activity as it executes when its active pages remain in real storage during its execution and its passive pages are paged in when required. A program uses real storage most efficiently when the active instructions and data in each logical phase are contained within the fewest number of pages possible.

The locality of reference characteristic does not apply to certain types of programs. For example, it does not apply to any program that is designed to optimize its performance at execution time by using the total amount of storage it has been allocated. This characteristic is usually true of sort/merge programs that initialize themselves to use all the storage made available to them in their partition or region during the sorting passes. The reference pattern for such a sort/merge is random and encompasses all the storage (and, therefore, all the pages) the program is assigned.

RELATIONSHIP BETWEEN VIRTUAL STORAGE SIZE AND SYSTEM PERFORMANCE

Assuming other required system resources are available, a given configuration can support a given virtual storage size and provide satisfactory performance when paging activity is kept at an acceptable level. Minimal paging activity occurs when enough real storage is present in the system to contain most or all of those pages of concurrently executing programs that are active at any given time. Paging activity then is required primarily for passive pages. Active pages are paged in (and later paged out as required) as the set of active pages for each program changes from one logical phase to another. The paging device(s) present must be capable of handling the demand for pages that results from the range of paging activity of the system.

As the amount of virtual storage used in a given system increases, the number of active and passive pages that the system must handle increases also. The ratio of active to passive pages will vary for a given increase in virtual storage, depending on how the additional virtual storage is used. As long as enough real storage is present to contain all or most of the increased number of active pages, the increase in paging activity required to support the additional virtual storage will be needed primarily for passive pages and should be relatively small. As soon as the use of more virtual storage causes the number of concurrently active pages to constantly exceed the capacity of real storage, the paging activity increase required to support the additional virtual storage becomes relatively large. As more and more active pages must be handled, paging activity could exceed the maximum paging capability of the system if task deactivation did not occur.

Figure 30.15.2 illustrates the increase in page faults that generally occurs as more virtual storage is used in a given system configuration. The curve begins at the point at which the amount of virtual storage used is equal to the amount of real storage present (virtual-to-real-storage ratio is 1). Paging activity begins as soon as the amount of virtual storage used exceeds the real storage present. As the virtual-to-real-storage ratio increases, so does paging activity. The system moves from passive paging activity (primarily paging of passive pages) into active paging (paging active pages in and out more of the time) and approaches the maximum paging capability of the system. As indicated previously, Figure 30.15.2 also illustrates the increase in page faults that generally occurs as less real storage is made available to support a given virtual storage size. The increase in page faults also causes the virtual-to-real storage ratio to increase.

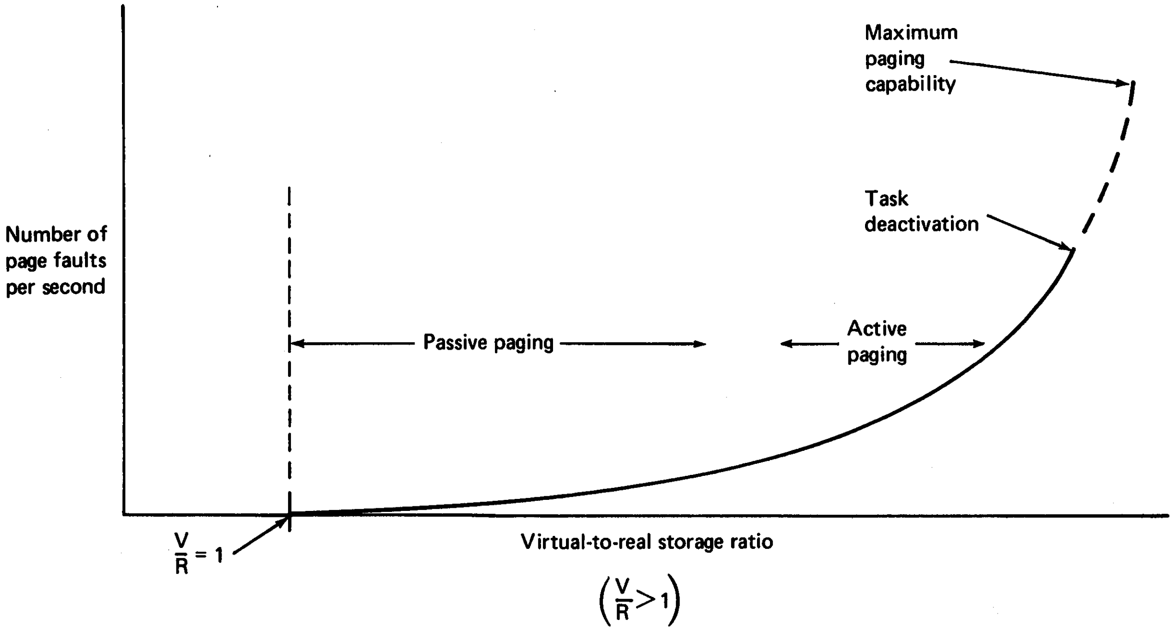


Figure 30.15.2. General effect on page faults of increasing the ratio of virtual storage used to real storage present in the system

Figure 30.15.3 illustrates how the paging factor only generally affects system performance. Figure 30.15.5, shown later, illustrates system performance taking into account all factors. The curve shows the performance of the system when passive and active paging are occurring, relative to the virtual-to-real storage ratio. The use of virtual storage can be increased with little or no adverse effect on performance as long as paging remains in the passive area. This is true because in the passive paging area there is a relatively small amount of paging and a high probability that all or most paging processing (CPU and I/O time) can be overlapped with other processing. As paging activity increases, there is a higher probability that CPU processing will be held up waiting for a paging operation to complete. As the CPU enters the wait state more frequently to wait for paging I/O and less paging I/O is overlapped, the paging factor causes performance to degrade more rapidly.

The actual virtual-to-real storage ratio at the time active paging begins in Figures 30.15.2 and 30.15.3 is a variable and depends on the way in which virtual storage is used, that is, active-to-passive page ratio of concurrently executing tasks.

Figure 30.15.4 illustrates the way in which the paging factor only can affect system performance in a given configuration, based on the active-to-passive page ratio. If the ratio of active to passive pages for executing tasks is relatively high most of the time, as shown in curve 1, the virtual-to-real storage ratio at the point at which active paging begins will be relatively low. Performance drops very rapidly in this case as more virtual storage is used, because the increased paging processing (I/O and CPU time) cannot be overlapped with other processing. This situation may apply to an installation initially when a switch from a nonvirtual storage to a virtual storage environment is made and more virtual storage is used, since existing programs were structured for optimum performance in a given partition or a region size rather than for optimum performance in a virtual storage environment.

If the active-to-passive page ratio for the system is low, as shown in curve 3, the virtual-to-real storage ratio can be relatively high when active paging begins. The performance curve stays flatter longer as virtual storage is increased when the active-to-passive page ratio is low. This situation can apply to an installation in which all executing programs are structured to minimize real storage requirements and page faults. An installation that continues executing all or most existing programs as they are presently designed and that structures new applications for optimum performance (low active-to-passive ratio) may be more common. Such installations may experience a virtual-to-real storage ratio somewhere between the low and the high extremes possible for a given job stream, as shown in curve 2.

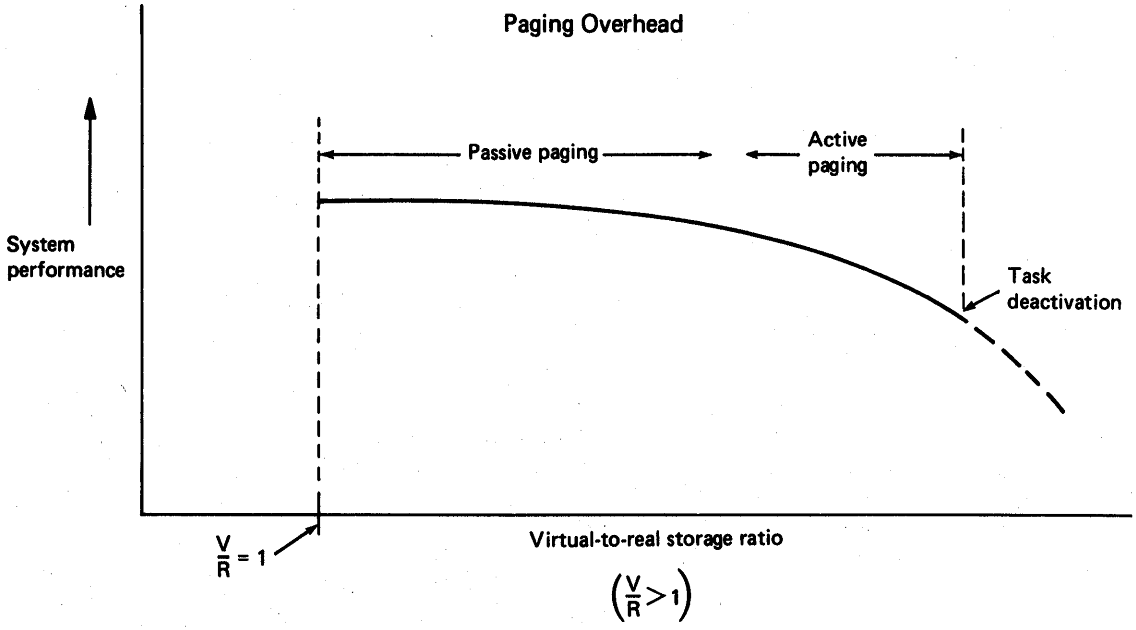


Figure 30.15.3. General effect on system performance of the paging factor only

The amount of virtual storage used in a system can be increased in several ways. First, the size of existing application programs can be increased by the addition of new functions. Second, the level of multiprogramming or multitasking can be increased, assuming other required resources, such as CPU time and I/O devices, are available. Third, the size of existing application programs can be expanded by (1) restructuring programs with a planned overlay or a dynamic structure to take them out of these structures and (2) combining two or more job steps within a job into one logical job step. The active-to-passive ratio of the additional pages the system must handle will usually be higher when the level of multiprogramming is increased than when existing jobs are restructured.

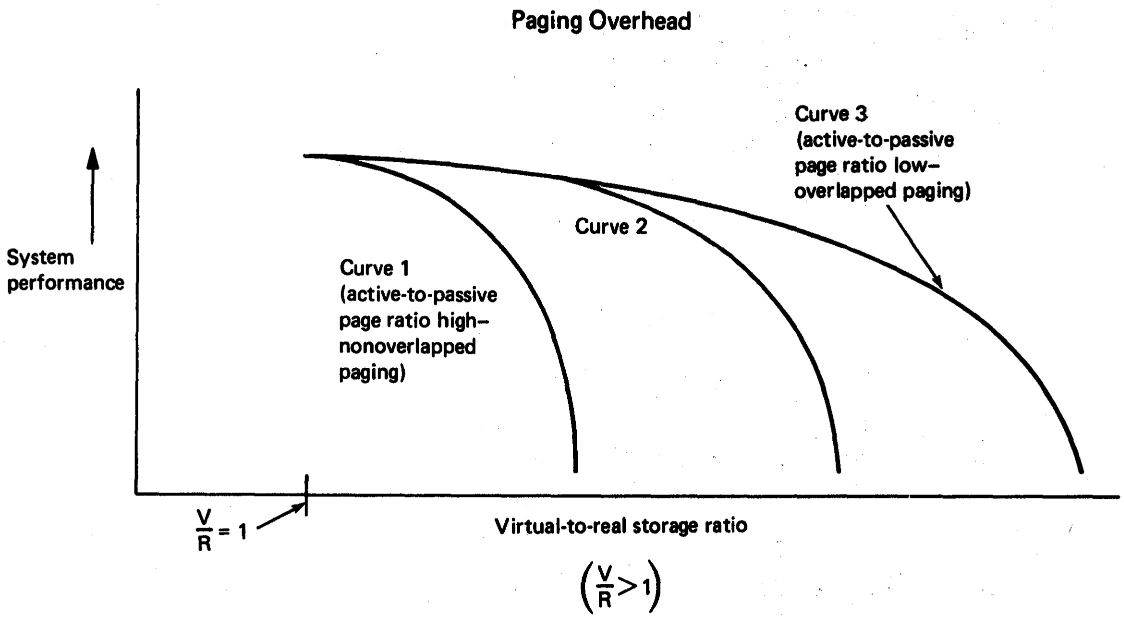


Figure 30.15.4. General effect of the paging factor on system performance with various active-to-passive page ratios

The way in which an installation should view the amount of virtual storage used and system performance for a given configuration, taking all performance factors into account, is illustrated in Figure 30.15.5. The increased use of virtual storage is beneficial to system performance up to a point. Thereafter, additional virtual storage can be used to handle additional functions at a variable cost in system performance. In reality, the virtual-to-real storage ratio and the page fault rate vary during system processing as the amount of virtual storage used (out of the total amount supported) and the amount of real storage available for paging vary. Best overall system performance is achieved when paging activity falls most of the time in the area identified on the curve as the operating range. More significant performance reduction begins when active paging is experienced.

Occasional active paging on an exception basis should be acceptable. More frequent active paging can be performed to achieve a desired function that does not justify changing the system configuration. However, when paging activity in a system is constantly at the point at which task deactivation occurs, system configuration changes should be made to improve system performance. Such changes might be the addition of more real storage, the addition of more or faster paging devices, or installation of a faster CPU. A history of the paging activity of the system can be maintained by recording the paging statistics provided by OS/VS1 and OS/VS2.

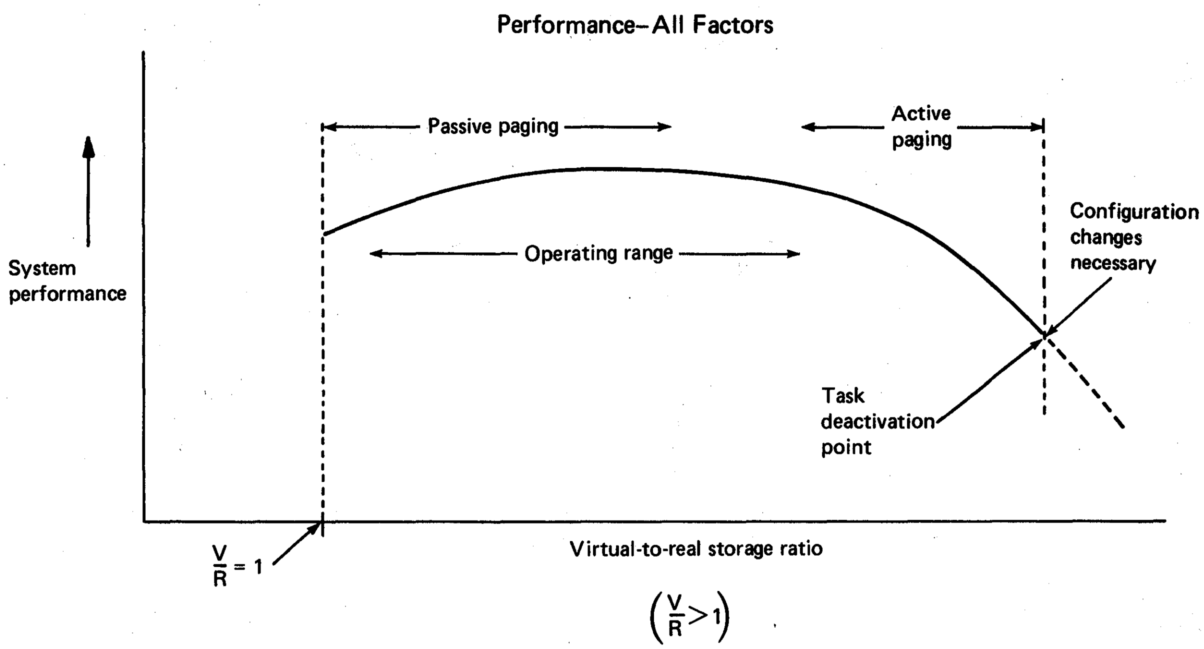


Figure 30.15.5. General system performance curve for a virtual storage environment

INCREASING SYSTEM PERFORMANCE IN A VIRTUAL STORAGE ENVIRONMENT

The IBM-supplied virtual storage operating systems are designed to provide an acceptable level of performance when existing problem programs are run without modification. However, given the additional resource requirements of virtual storage support and the new factors that affect system performance in a virtual storage environment, once a virtual storage operating system is installed (either on an existing configuration or a larger configuration) certain steps can be taken to improve performance or to achieve optimum performance. The benefit to be achieved by taking any one of the steps outlined must be evaluated on an installation basis, taking the specific configuration and operating environment into account. Some steps, for example, are more practical for large configurations than for small configurations. The following can be done:

- Use larger I/O buffers. This step is practical primarily for sequential data sets and can be used most effectively when previous real storage limitations prevented the use of larger buffer sizes in general and, in particular, optimum buffer sizes for disk data sets. In addition to reducing the total I/O time required to process a data set, as would occur in a nonvirtual storage environment, increasing buffer size reduces the number of I/O requests required to process the data set and, thereby, reduces the CPU time required for channel program translation and page fixing. This technique should be used taking into account the amount of real storage present in the system. If the buffer size of several data sets that are being processed concurrently is increased considerably or made large initially, the amount of real storage that must be short-term fixed increases considerably also and potentially increases the number of active pages. This may adversely affect system performance if the system has a relatively limited amount of real storage available for paging.

- Increase the page-fault-handling capability of the system when paging activity constantly causes task deactivation. This can be accomplished by (1) using a direct access device for paging that is faster than the currently used paging device, (2) allocating more direct access devices for paging to enable more overlap of paging activity, or (3) reducing or eliminating contention for the existing paging device(s). Contention for the paging device can be relieved by using dedicated paging devices, or reducing the amount of other I/O activity on the channel to which the paging device is attached, or dedicating a channel to paging. Alternatively, the same paging device configuration can be maintained while page fault occurrence is decreased by the addition of real storage.

- Use code that does not modify itself. Use of this type of code can reduce the amount of page-out activity required. Such code can be produced using OS PL/I and the OS Assembler Language.

- Execute programs in nonpaged mode only when actually required. Use of nonpaged mode should be limited because the amount of real storage available for paging operations during the operation of a nonpaged program is reduced by the size of the program and can affect system performance. If a nonpageable program is to be present in a system for an extended period of time or at all times, it should be considered part of the fixed real storage requirement so that the amount of real storage actually available for paging can be more accurately determined.

- Structure new application programs to operate efficiently in a paging environment. This is done by structuring programs to achieve a reasonable balance between page faults and real storage requirements. The extent to which this is done can vary widely by installation. The benefits that can be obtained should be evaluated in light of the additional programmer effort required. In this respect, deciding on the degree to which programs should be structured for efficient operation in a paging environment is similar to deciding how a high-level language should be used. The emphasis can be on most efficient program execution, which can require more programmer effort, or on most efficient use of programmer time, which can result in less efficient programs. Alternatively, there can be a tradeoff between programmer time and efficient programs (only the most frequently used programs are optimized, for example).

  Many of the general program structure techniques discussed do not require a large amount of additional effort or knowledge on the part of programmers--only that they adopt a particular programming style. All of the suggested techniques can be used by Assembler Language programmers. Some can be used with certain high-level languages and not with others. More of the suggested techniques can be used in PL/I programs than in other high-level language programs.

  Two major steps can be taken to structure programs to use real storage most efficiently and to incur the smallest possible number of page faults. The first is to adopt a certain programming style, one aspect of which is similar to the style that has been encouraged with System/360 and System/370, namely, that of modular programming. The second is to package program code and data within page boundaries. The objective of improving programming style is to construct a program that consists of a series of logical processing phases each of which contains a relatively small number of active pages. The objective of packaging code within pages is to group active code together to avoid crossing page boundaries in such a way that more real storage than is really necessary is required to contain the active pages of a logical phase.

In order to cause references to active instructions and data to be localized, the following general rules should be applied to programs:

1. A program should consist of a series of sequentially executed logical phases or--in System/370 programming terminology--a series of subroutines or subprograms. The mainline of the program should contain the most frequently used subroutines in the sequence of most probable use, so that processing proceeds sequentially, with calls being made to the infrequently used subroutines, such as exception and error routines. This structure contrasts with one in which the mainline consists of a series of calls to subroutines. Frequently used subroutines should be located near each other. Infrequently used subroutines that tend to be used at the same time whenever they are executed should be located near each other also.

2. The data most frequently used by a subroutine should be defined together so that it is placed within the same page, or group of pages, instead of scattered among several pages. If possible, the data should be placed next to the subroutine so that part or all of the data is contained within a page that contains active subroutine instructions (unless the routine is to be written in such a way that it is not modified during its execution). This eliminates references to more pages than are actually required to contain the data and tends to keep the pages with frequently referenced data in real storage.

3. Data that is to be used by several subroutines of a program (either in series or in parallel by concurrently executing subtasks) should be defined together in an area that can be referenced by each subroutine.

4. A data field should be initialized as close as possible to the time it will be used to avoid a page-out and a page-in between initialization and first use of the data field.

5. Structures of data, such as arrays, should be defined in virtual storage in the sequence in which they will be referenced, or referenced by the program in the sequence in which a high-level language stores them (by row or by column for arrays, for example).

6. Subroutines should be packaged within pages when possible. For example, avoid starting a 1500-byte subroutine in the middle of a 2K page so that it crosses a page boundary and requires two page frames instead of one when it is active. Subroutines that are smaller than page size should be packaged together to require the fewest number of pages, with frequently used subroutines placed in the same page when possible. This applies to large groups of data as well. The linkage editor supplied with OS/VS1 and OS/VS2 has new control statements that can be used to cause CSECTs and COMMON areas to be aligned on page boundaries, and to indicate the order in which CSECTs are placed in the load module. This linkage editor facility can be used with certain high-level language programs that contain multiple CSECTs (such as PL/I and ANS COBOL) as well as with Assembler Language programs.

- Use the OS PL/I Optimizing Compiler instead of OS PL/I F. The code produced by this language translator has characteristics that make it more suited to a virtual storage environment than the code produced by PL/I F. First, generated code is grouped into functionally related segments, by PROCEDURE and DO group, for example, which can help reduce paging. When PL/I allocates buffers and I/O control blocks, they are packed together and can potentially require fewer pages than if no attempt was made to define them together. Reentrant code can be produced by the OS PL/I Optimizing Compiler, and its library routines are reentrant. This reduces page-out requirements. User-written reentrant PL/I routines that are required by concurrently executing problem programs can be made resident in virtual storage and shared to reduce real storage and paging requirements for active pages of these routines.

- Use the shared library feature of the OS PL/I Optimizing Compiler and the COBOL Library Management Facility of the OS ANS COBOL language translator to make library modules resident in virtual storage so they can be shared by concurrently executing problem programs. Pages containing active library modules will tend to remain in real storage and thereby reduce paging and real storage requirements for these modules.

- Restructure existing application programs to incur as few page faults as possible, use the least possible amount of real storage, and take advantage of the program structure facilities that a virtual storage environment offers. This can be accomplished by (1) using the techniques described above, (2) taking planned overlay and dynamic structure programs out of these structures, and (3) combining into one logical step two or more steps of a job that would have been one job step if the required real storage were available. The last technique can eliminate redundant I/O time that is currently used to read the same sequential input file into two or more job steps and to write intermediate results from one job step in one or more sequential data sets for input to the next job step.

- Increase the level of multiprogramming in the system. This can be accomplished by (1) performing more peripheral I/O operations concurrently (more readers and writers), (2) operating more regions or partitions concurrently, or (3) increasing the use of multitasking (structuring a TCAM message processing program to use multitasking to enable several different types of transactions to be processed concurrently, for example).

  System throughput can be improved in a virtual storage environment if a higher level of multiprogramming causes more CPU and I/O time to be overlapped, which results in more effective utilization of available system resources. The larger the number of tasks in the system under these conditions, the less chance there is for the CPU to enter the wait state because no task is ready to execute. Better utilization of real storage in a virtual storage environment can enable more tasks to be present in the system.

  In order to achieve performance gains by increasing the level of multiprogramming, the potential for more overlap of CPU and I/O time must exist in a system and/or the potential must exist for reduction of I/O time via increased overlapping of channel activity and reductions in unoverlapped seek time (that can result from new system performance enhancements). The required hardware resources, such as CPU time, real storage, I/O devices, and direct access storage, must be available as well. The most critical resource in this situation is available CPU time. As the percentage of CPU utilization gets higher, there is less potential for increasing throughput via increasing the level of multiprogramming.

The information presented in this subsection is designed to enable the reader to more fully understand the way a system operates in a virtual storage environment and the facts that influence system performance. Understanding the environment and knowing the actions that can be taken to increase system performance will enable each installation to quantify the amount of effort that is to be devoted to optimizing the performance of a virtual storage operating system.

SECTION 40: VIRTUAL MACHINES

This section discusses the basic concepts, general operation, and advantages of virtual machines, as defined and implemented in Virtual Machine Facility/370. No previous knowledge of virtual machines is assumed. The virtual machine concept is a logical extension of the virtual storage concept. Therefore, comprehension of dynamic address translation hardware and virtual storage concepts, terminology, and advantages, as discussed in Sections 30:05 and 30:10, is assumed.

VM/370 consists of the Control Program (CP), Conversational Monitor System (CMS), Remote Spooling Communications Subsystem (RSCS), and Interactive Problem Control System (IPCS) components. CP supports the concurrent operation of multiple virtual machines. CMS, operating in a virtual machine under CP control, provides conversational time sharing facilities to a single user. RSCS, operating in a virtual machine under CP control, provides for the transmission of data between remote users and virtual machines via binary synchronous communication lines. IPCS, operating in a CMS virtual machine, provides interactive problem management, problem determination, and problem isolation.

VM/370 is the successor to CP-67/CMS. Virtual machine support was first provided by IBM in CP/67. In the CMS time sharing environment in which CP-67/CMS was primarily used, the major advantage of the virtual machine facility was that it enabled each CMS user to appear to have a complete System/360 (Model 22 to 75) at his disposal and to be isolated from all other CMS users. Each CMS user had access only to his own virtual machine and, therefore, could not inadvertently interfere with the operation of other CMS virtual machines. VM/370 also provides these facilities and can be used in nondedicated time sharing environments to provide other advantages as well.

The information presented in this section is prerequisite reading for the optional Virtual Machine Facility/370 Features Supplement, which can be inserted as Section 110 of this guide. The VM/370 supplement discusses the features and operation of the components of VM/370, as well as performance considerations for a virtual machine environment and the types of installations that can benefit most from the use of VM/370.

40:05 DEFINITION AND GENERAL OPERATION

A virtual machine is a functional simulation of a complete computer system, including a virtual CPU, virtual storage, virtual channels, virtual I/O devices, and a virtual operator's console, that appears to the user to be a real machine. In a VM/370 environment, a virtual machine is the functional equivalent of a System/370 (Models 135 to 168) and its associated I/O devices.

The control program (CP) component of VM/370, executing in a real machine (System/370 Models 135 through 168 with dynamic address translation hardware), supports concurrent operation of multiple virtual machines using multiprogramming techniques that enable real machine resources to be shared by multiple virtual machines. Each virtual machine is dedicated to a single user and isolated from other virtual machines. None of the components of one virtual machine can be accessed by a program that is executing in another virtual machine except via the controlled sharing facilities that are provided by CP.

The operation of a virtual machine and scheduling of the work it performs are handled by an operating system rather than by CP. That is, each virtual machine has an operating system executing in it that allocates machine resources and schedules the execution of problem programs just as if the operating system were executing in a real machine. In order to initiate operations in a virtual machine, the user must log on the virtual machine and IPL an operating system in it. The logon procedure establishes a connection with CP and the existence of a specific virtual machine for this user. A logon is performed using a console or terminal device of the type that CP supports as a virtual operator's console.

The virtual operator's console is the means by which the user controls the operation of his virtual machine and communicates with the operating system executing in it. CP provides a set of commands that (1) simulate the system control panel of the virtual machine, (2) provide for alteration of a virtual machine configuration, (3) request various services from CP for a virtual machine, and (4) control operation of the real machine. When a CP command is entered via the virtual operator's console, CP receives control and performs the required functions. Communication between the user and the operating system is accomplished using the operating system command language and the virtual operator's console. CP performs any simulation required to make the real I/O device the operator is using as a virtual operator's console appear to be the primary console device type that is defined for the operating system.

In a VM/370 environment, a virtual operator's console is frequently called a remote terminal because, in most cases, a terminal device type is actually used aş the virtual operator's console device. However, the real I/O device that is used as the virtual operator's console may be a System/370 console device as well as a local or a remote terminal.

VM/370 supports execution of any one of the following System/360 and System/370 programming systems in a virtual machine:

- CMS component of VM/370
- RSCS component of VM/370
- DOS Version 3, DOS Version 4, or DOS/VS
- APL 360-DOS
- OS PCP, MFT, or MVT
- OS ASP Version 3
- OS/VS1
- OS/VS2 SVS (Releases 1, 1.6, and 1.7)
- OS/VS2 MVS (Releases 2 and up) in uniprocessor mode only
- PS44
- VM/370

Any number and combination of the above operating systems can execute concurrently in a VM/370 environment, subject to the availability of the required real machine resources, including multiple copies of the same operating system (OS/VS1 executing in more than one virtual machine, for example). With a few exceptions, all the facilities that are supported by these operating systems when they execute in a real machine can be used when the operating system executes in a virtual machine in a VM/370 environment. Figure 40.05.1 conceptually illustrates the real and virtual machine environment that is supported by VM/370.

Each virtual machine that is to be supported by CP must be user defined and stored in the VM/370 directory. The size of virtual storage, the virtual I/O devices to be used, the options to be used, and a virtual console are usually specified. Virtual machine configurations can be different from each other and, within certain limitations, different from that of the real machine in terms of these specifications.

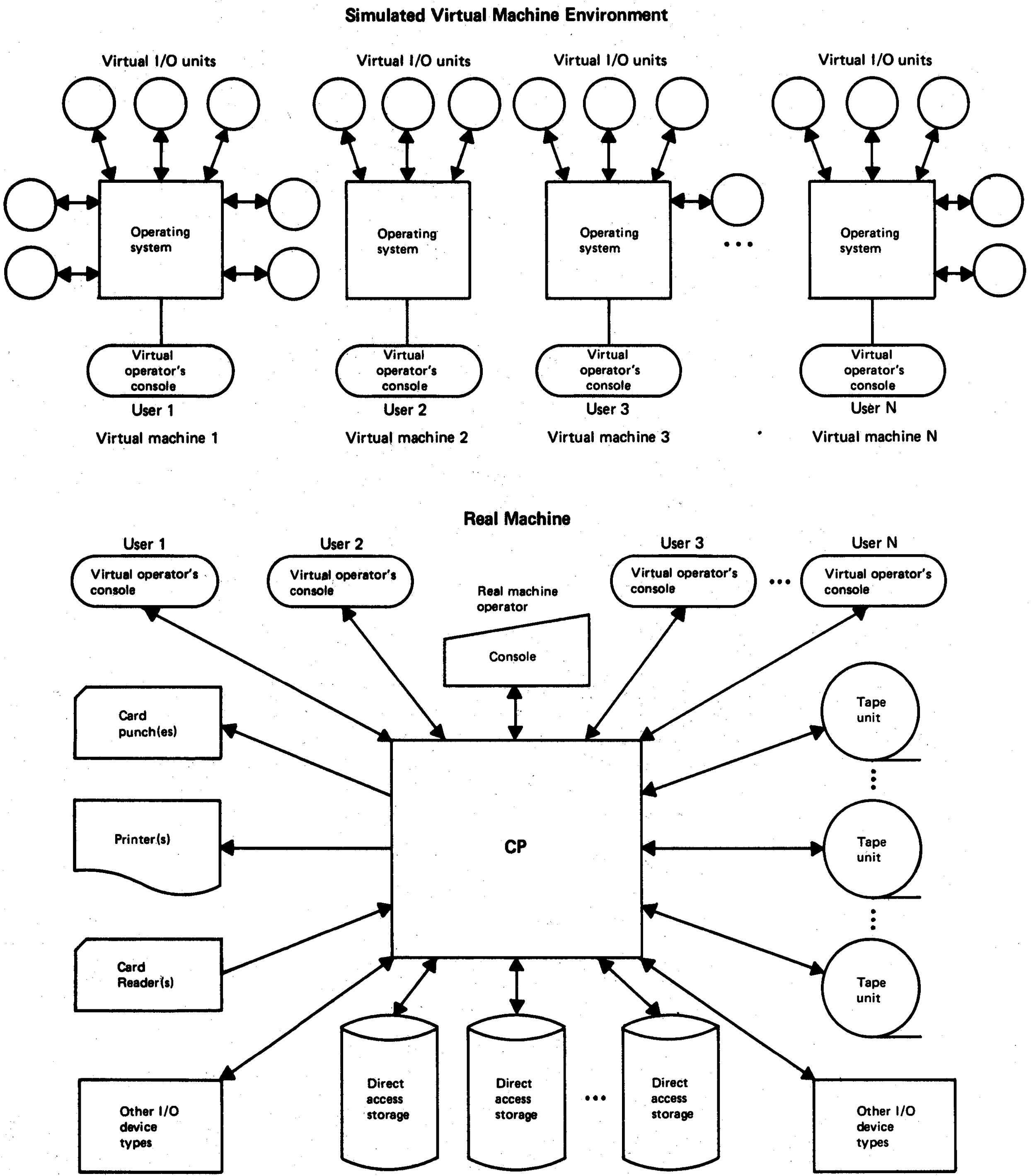


Figure 40.05.1. Conceptual illustration of the real and virtual machine environment that is supported by VM/370

Virtual CPU Simulation

CP is resident in real storage during operation of the real machine. It controls the operation of the real machine, schedules the execution of virtual machines, and simulates virtual machine hardware components using the hardware components of the real machine. In order to be able to perform its functions and isolate virtual machines from each other, CP must have exclusive control over the status and modes of operation of the real machine, as does the control program of an operating system. Hence, CP always executes with the real machine in supervisor state and receives control after all real machine interruptions.

Virtual machines always operate with the real machine in problem state. Therefore, any time any program that is executing in a virtual machine issues a privileged instruction, an interruption occurs in the real machine. CP receives real CPU control and takes the required action. This may involve simulating execution of the privileged instruction for the virtual machine or returning real CPU control to the control program in the virtual machine for which the interruption occurred so that the interruption can be processed by that control program. In this manner, CP maintains control of the real machine. In addition, CP simulates the existence of both a supervisor state and a problem state in the virtual machine while, in reality, the virtual machine operates only in problem state.

CP gives control of the real CPU to operating virtual machines on a time-shared basis to simulate the existence of multiple CPUs. A virtual machine can execute any System/370 instruction except READ DIRECT and WRITE DIRECT, which are part of the Direct Control feature; the multiprocessing instructions; and SET CLOCK, which is treated as a NOP because CP controls the setting of the time-of-day clock. In addition, the DIAGNOSE instruction is reserved for communication between executing operating systems and CP.

The System/370 instructions and CPU features that are used by the control and problem programs executing in a virtual machine must be present in the CPU of the real machine in which CP executes. CP does not simulate the existence of System/370 instructions and CPU hardware features that are not present in the real machine. A virtual CPU can appear to be executing either with BC mode or EC and DAT modes specified, depending on the mode required by the operating system executing in it. However, EC and DAT modes are always specified in the real CPU when a virtual CPU is executing since address translation is required to support the existence of virtual storage for the virtual machine.

Virtual Storage Simulation

Each virtual machine can have up to 16,777,216 bytes of virtual storage, which is the maximum virtual storage size for System/370. The existence of virtual storage for a virtual machine is simulated by CP using DAT hardware and external page storage, as is done in a virtual storage environment (discussed in Section 30).

Operating system programs that are executing in a virtual machine (both control and problem programs) are paged in and out of real storage in the real machine on a demand paged basis as they execute. Real storage allocation, external page storage allocation, and paging operations are handled entirely by CP and are transparent to the control and problem programs that are executing in the virtual machines. In this manner, CP provides one virtual storage for each virtual machine, and real storage in the real machine is shared by concurrently operating virtual machines. The implementation of virtual storage in a virtual machine environment is conceptually illustrated in Figure 40.05.2.

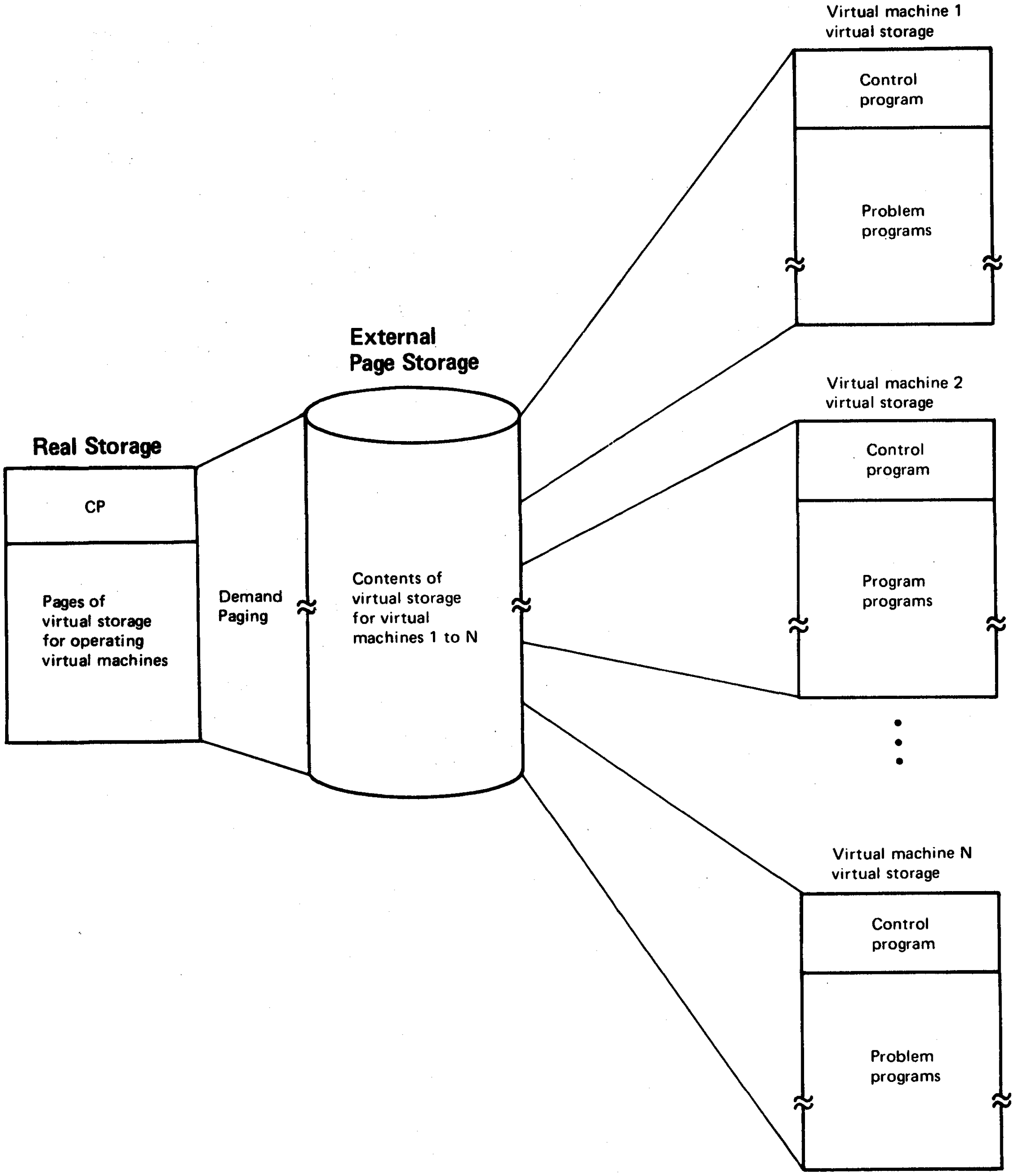


Figure 40.05.2. Conceptual illustration of the implementation of virtual storage in a virtual machine environment

The virtual storage defined for a virtual machine always appears to be real storage to the operating system that is executing in the virtual machine. In effect, an operating system that does not support virtual storage, such as DOS Version 4 or OS MFT, has virtual storage support provided by CP when such an operating system executes in a virtual machine and, therefore, offers the functional advantages of a virtual storage operating system.

When executing in a virtual machine, an operating system that does support virtual storage uses the virtual storage defined for the virtual machine as real storage in order to simulate the existence of the virtual storage it is designed to support. As shown in Figure 40.05.3, the virtual storage operating system builds a segment table and page tables to translate addresses in the virtual storage it supports to addresses in the virtual storage defined for the virtual machine, which the operating system assumes is real storage. CP always builds and maintains a segment table and page tables for each virtual machine. These tables are used to translate addresses in the virtual storage of the virtual machine to addresses in real storage in the real machine.

When a virtual storage operating system is executing in a virtual machine, CP constructs and maintains a third set of tables using the contents of the other two sets of tables. The third set of tables, a shadow segment table and shadow page tables, are the tables that are actually used for address translation when the virtual machine operates. The shadow tables are used to translate addresses in the virtual storage the operating system supports to addresses in real storage in the real machine.

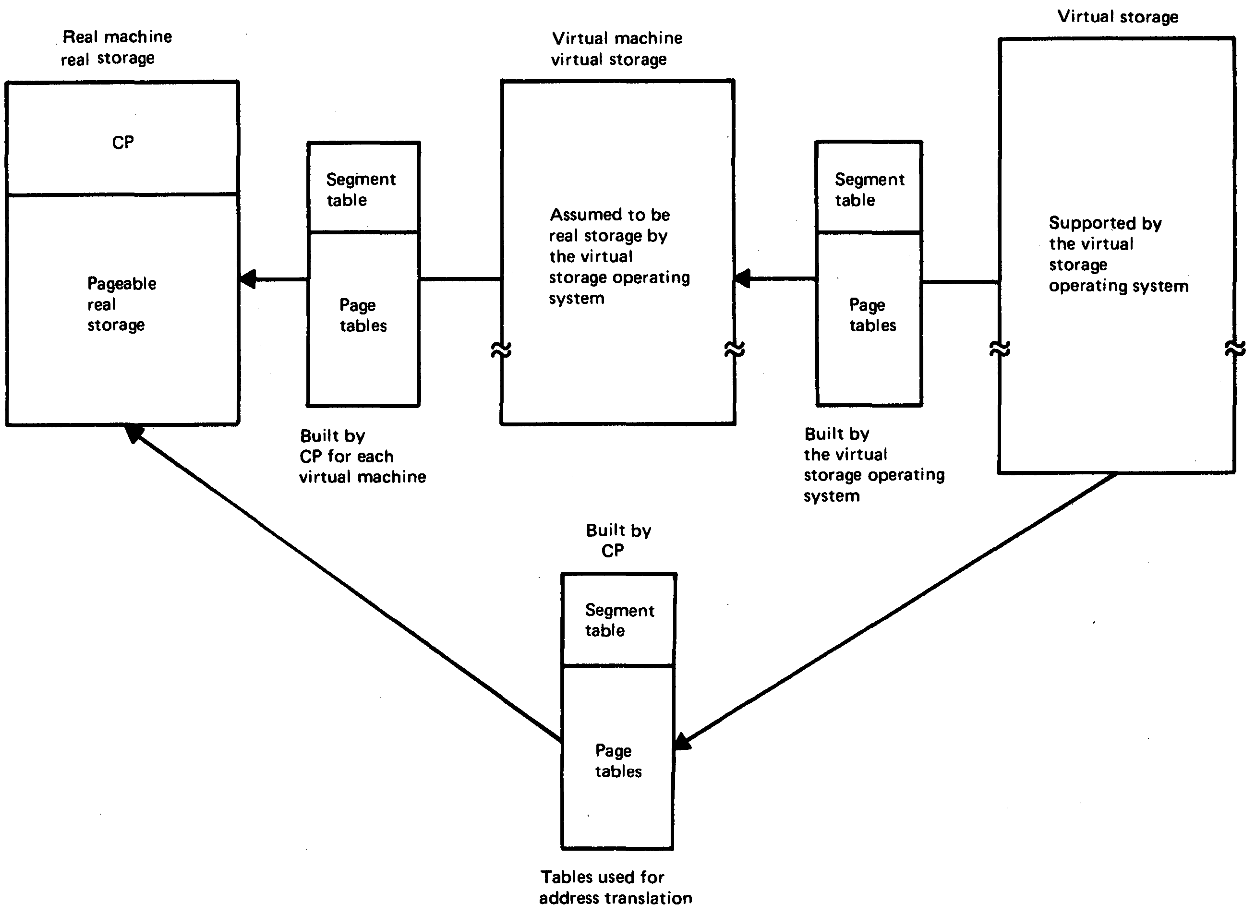


Figure 40.05.3. Segment tables and page tables built when a virtual storage operating system executes in a virtual machine

Virtual I/O Component Simulation

The virtual channels, control units, and I/O devices defined in each virtual machine configuration are simulated by CP using real channels, control units, and I/O devices that are of the same type. While each virtual I/O device defined must have a real I/O device counterpart in the real machine configuration, there does not necessarily have to be a one-to-one correspondence. In addition, the I/O device addresses assigned to virtual I/O devices need not be the same as the addresses of their real I/O device counterparts. CP also allows a virtual direct access device to be simulated by only a portion of a real direct access device volume. Such a virtual direct access device is called a minidisk. Support of a minidisk facility enables one real direct access device to simulate the existence of several virtual direct access devices of the same type and thus provides more efficient use of available direct access storage.

Virtual I/O devices are always simulated on a real I/O device of the same device type unless the spooling facility of CP is used. (CP also allows 2311 disk storage to be simulated using 2314/2319 disk storage and the minidisk facility.) The local spooling capability of CP provides data transcription between unit record devices and direct access storage devices and is functionally similar to DOS POWER, OS readers and writers, OS HASP, and OS/VS JES. In effect, the CP spooling facility enables virtual unit record devices (card readers, card punches, and printers) to be simulated using direct access storage. CP also provides console spooling and a remote spooling facility.

The virtual I/O devices in a virtual machine configuration are logically controlled by the operating system that is executing in the virtual machine rather than by CP. That is, all the data management routines of the operating system (physical record processing, logical record processing, and error recovery routines) execute as usual. Therefore, a virtual machine I/O configuration can include any I/O device types that are supported by the operating systems that will execute in the virtual machine, as long as real I/O device counterparts exist in the real machine I/O configuration as required.

CP controls only the scheduling and actual initiation of virtual machine I/O operations in the real machine. When a START I/O instruction is issued by an operating system control program that is executing in a virtual machine, a privileged operation interruption occurs and CP receives real CPU control. CP translates the virtual I/O device address to its counterpart real I/O device address and, for minidisks, converts virtual cylinder addresses to corresponding real cylinder addresses, as required. CP also performs the necessary channel program translation and page locking operations and queues the I/O request if it cannot be started.

After the I/O operation is started, CP returns the condition code to the operating system control program that initiated the I/O request so that appropriate action can be taken. When the I/O operation completes and causes an I/O interruption, CP receives CPU control, gathers I/O status information, and attempts to restart the available real I/O components. CP presents the status data to the operating system control program via a simulated I/O interruption for the virtual machine in which the operating system is executing.

CP completely controls operation of the real I/O devices that are required for its own execution, such as paging and spooling devices. This includes determining the need for I/O operations, scheduling and initiating I/O requests, handling I/O interruption processing, and performing error recovery procedures.

Virtual Machine Assist Feature

The Virtual Machine Assist feature is available as an RPQ for the Model 168. This feature is designed to improve total system performance in a VM/370 environment and can also improve the performance achieved by certain operating systems that operate under CP control in a virtual machine. The virtual machine assist feature performs the same functions as some of the most frequently used virtual machine simulation routines of CP. Use of the virtual machine assist feature can result in virtual machine performance improvement through elimination of CP processing that would otherwise cause an operating system to experience throughput degradation when it executes in a virtual machine instead of a real machine. Total system performance improvement is achieved if a higher level of multiprogramming can be maintained as a result of the elimination of certain CP processing.

The virtual machine assist feature is controlled by mask bits in control register 6. When the virtual machine assist feature is enabled, certain types of real machine interruptions that occur when a virtual machine has real CPU control cause the virtual machine assist hardware feature to gain control to simulate the required virtual machine function. The virtual machine assist feature is entered when one of the following occurs:

- A privileged instruction program interruption occurs that is caused when a virtual machine issues an INSERT PSW KEY, INSERT STORAGE KEY, LOAD PSW, LOAD REAL ADDRESS, RESET REFERENCE BIT, SET PSW KEY FROM ADDRESS, SET STORAGE KEY, SET SYSTEM MASK, STORE CONTROL, STORE THEN AND SYSTEM MASK, or STORE THEN OR SYSTEM MASK instruction. The virtual machine assist feature simulates execution of the privileged instruction, and operation of the virtual machine continues with execution of the instruction after the privileged instruction.

- An SVC instruction except SVC 76 is issued by a virtual machine. PSW switching for the virtual machine is simulated by the virtual machine assist feature.

- A page translation program interruption occurs in a virtual machine in which a virtual storage operating system is executing. The virtual machine assist feature updates the appropriate shadow page table if possible.

The virtual machine assist hardware feature performs the same functions as the counterpart simulation routines in CP, with a few exceptions. The virtual machine assist feature does not handle certain special situations for a few of the privileged instructions supported. The unsupported special situations are those that occur infrequently and that would require the inclusion of a considerable amount of additional hardware. When these special situations occur, the appropriate simulation routine of CP is entered to perform the required functions.

The amount of throughput improvement that occurs for an operating system when the virtual machine assist feature is used depends on the extent to which the operating system utilizes the functions the virtual machine assist feature supports. If the increase in run time an operating system experiences when it executes in a virtual machine is caused to a large extent by the CP processing that is required to simulate functions supported by virtual machine assist hardware, a relatively significant performance gain can be expected. The virtual machine assist feature can be of the most benefit, for example, to operating systems that support virtual storage (DOS/VS, OS/VS1, and OS/VS2).

The virtual machine assist feature is supported by VM/370 as of Release 2. Additional details regarding the operation of the virtual machine assist feature and the support provided by VM/370 are discussed in _Virtual Machine Facility/370 Features Supplement_.

40:10 GENERAL ADVANTAGES OF A VIRTUAL MACHINE ENVIRONMENT

The advantages of VM/370 complement those of virtual storage operating systems. Like a virtual storage environment, a virtual machine environment is designed primarily to support new functions rather than increase system performance. Essentially, CP is a simulator. Traditionally, simulators have been used to provide a desired function at the expense of performance. The new functions provided by virtual machines (1) can increase the rate of new application development and (2) expand operational capabilities over those provided by virtual storage. The CMS component of VM/370 supplements these two major advantage areas of a virtual machine environment by supporting time sharing facilities, such as online program development, conversational program execution and problem solving, and interactive text processing.

The following indicates the way in which the virtual machine environment that is supported by the CP component of VM/370 aids the installation of new applications and identifies the new operational features such an environment supports. The functions and specific advantages of CMS are discussed in the VM/370 supplement.

Increasing New Application Development

Since virtual machine support includes support of a virtual storage environment for each virtual machine, all the capabilities virtual storage provides that aid new application development are present in a virtual machine environment as well. (These capabilities are discussed at the end of Section 30:05.) By enabling multiple operating systems to execute concurrently in one real machine, the virtual machine environment supported by CP also provides the following new capabilities:

- Testing of new programs can be more extensive and completed sooner by the elimination of dedicated testing periods. While a virtual storage environment can eliminate most program testing restrictions that result from real storage size limitations, the isolation that is provided by executing a program in a virtual machine eliminates the need to test programs that can cause total system termination in a dedicated environment. For example, system-oriented routines written by system programmers and teleprocessing programs, which usually are tested only during scheduled dedicated testing periods, can be tested while production work is in progress. This can eliminate the need to establish testing periods during second or third shift and, by reducing individual test turnaround time, enables more of this type of testing to be accomplished in a given time period.

- Testing of new programs can be completed sooner through the use of console debugging, when necessary. Using the CP commands that simulate system control panel functions, the programmer can use any console debugging facility that is available on a real machine, such as setting address stops, examining and altering general registers, displaying and altering virtual storage, etc., without interfering with production work. CP also provides other debugging services, such as an extensive set of traces, that can be invoked by CP commands. Console debugging, which can enable difficult-to-locate program errors to be detected more quickly than with desk debugging, is usually not permitted in a nonvirtual machine environment, except as a last resort, or is scheduled for nonproduction periods. Program testing turnaround time can be significantly reduced through the use of console debugging.

- Transition from one release of an operating system to another release or from one operating system to another can be accomplished more quickly because of the capability of executing multiple operating systems concurrently. A new release of an operating system can be generated and tested in one virtual machine while production work continues in another virtual machine using the existing release. Existing application programs and system-oriented programs that must be modified or newly written (to use a new facility or new language translator, for example) can be tested during production processing as well. The multiple virtual machine facility also enables an installation to execute programs that are dependent on a back release (because the release is user modified, for example) concurrently with each new release of that operating system or with an entirely new operating system (such as a back release of a DOS version operating concurrently with OS).

- CMS can be used to perform online program development concurrently with the processing of production work using either OS or DOS. Significant gains in programmer output can be realized through writing, compiling, and testing programs using an online terminal in a conversational manner. This enables new applications to become operational sooner. When CMS is used, each programmer has his own virtual machine with CMS executing in it. Therefore, the occurrence of a programming or operational error in one virtual machine can cause termination of that virtual machine only. Other programmers and production work are not affected.

Expanded Operational Capabilities

In addition to the new operational facilities a virtual storage environment provides (discussed in Section 30:05), a multiple virtual machine environment offers the following capabilities:

- Operating system maintenance can be performed concurrently with production work. PTF's can be applied and tested using one virtual machine without the possibility of causing the abnormal termination of another virtual machine that is processing production work.

- Operator training can be done using a virtual machine, which eliminates the need to dedicate the entire real machine to this function. Multiple operators can be trained while production work is in process without the possibility of terminating real system operations through an operator error.

- A system can be backed up by another system that not only has less real storage but that also has real I/O devices with different addresses, fewer direct access devices, and fewer channels, as long as sufficient I/O devices of the required type are available.

- New channel and direct access device configurations can be simulated using a virtual machine for the purpose of evaluating the load on the new I/O configuration before it is installed on the real machine. Similarly, ASP configurations consisting of two or more machines can be simulated in a virtual machine environment using only one real machine. This enables an installation without ASP installed to determine the activity of such a configuration and gain experience in its operation before the second system is installed or before making the decision to install ASP. The ASP user can also experiment with different ASP configurations.

As the above indicates, a virtual machine environment, as supported by VM/370, offers several unique capabilities that can be of benefit to small, intermediate, and large System/370 users. In most cases, VM/370 can be used to best advantage as complementary programming system support in Model 168 installations in which a version of OS is used as the primary programming system. VM/370 can be used in the same system as the OS or OS/VS operating system or in a separate support system. A discussion of the types of installation environments in which VM/370 will be most frequently used is contained in Virtual Machine Facility/370 Features Supplement.

SECTION 50: I/O DEVICES FOR MODELS 1 AND 3

50:05 I/O DEVICE SUPPORT

All I/O devices, consoles, and telecommunications terminals that can be attached to the Model 165 can be attached to Models 1 and 3 of the Model 168. However, all I/O devices supported by OS MFT and MVT are not also supported by OS/VS1 and OS/VS2, respectively. (See the optional programming systems supplements for I/O device support.)

Model 65 devices that are not part of the standard Model 165 I/O configuration are not part of the standard Model 168 I/O configuration. The integrated storage controls feature and several I/O devices can be attached to the Model 168 but not to the Model 165 (see the table in Section 70:05).

The I/O devices discussed in this section attach to a Model 168 (Models 1, 3, and A3) but not to a Model 165.

50:10 3333 DISK STORAGE AND CONTROL MODEL 11 AND 3330 DISK STORAGE MODEL 11

A 3330-series string that is attached to a Model 168 can contain 3333 Model 11 Disk Storage and Control and 3330 Model 11 Disk Storage modules, which do not attach to the Model 165. The drives in these modules offer twice the capacity of the drives in Model 1 and 2 modules. Model 11 of the 3333 consists of two independent drives, device-oriented control functions, and power for itself and the drives that can be attached to it, as does Model 1 of the 3333. Model 11 of the 3330 consists of two independent drives without the device-oriented control functions that are part of a 3333, as does a 3330 Model 1.

In a Model 168 configuration, the 3333 Model 11 attaches to 3830 Storage Control Model 2 and Integrated Storage Controls. It must be the first module in each 3330-series string that is attached to these control units. The 3330 Model 11 attaches only to 3333 modules, Models 1 and 11. Up to three 3330 modules, in any combination of Models 1, 2, and 11, can be attached to a 3333 Model 1 or 11 module.

With one exception, Model 11 3330-series drives are functionally like Model 1 and 2 drives. The drives in 3330 and 3333 Model 11 modules have a standard write format release feature that is not provided for 3330 Model 1 and 2 and 3333 Model 1 drives. This feature enables a Model 11 drive to disconnect from a 3333/3330 Model 11, 3830 Model 2, or the ISC while the drive is erasing to the end of the track after a record has been written with a formatting write command. This facility frees the control unit and channel for the initiation of another I/O operation.

The removable 3336 Model 11 disk pack is used with 3333 and 3330 Model 11 drives. Like a 3336 Model 1, a 3336 Model 11 has 19 recording surfaces. However, the Model 11 disk pack has 808 data cylinders, instead of 404, for a maximum capacity of 200 million bytes. The Model 11 disk pack also has seven alternate cylinders, like a Model 1. Hence, the maximum capacity of a 3330-series string of all Model 11 drives is 1600 million bytes.

Model 11 3336 Disk Packs are interchangeable across all 3330 Model 11 and 3333 Model 11 drives but cannot be used with Model 1 and 2 3330-series drives. The 3336 Model 11 Disk Pack has a physical interlock so that it cannot be mounted on a 3330 Model 1 or 2 drive or a 3333 Model 1 drive. The 3336 Model 1 Disk Pack has a physical interlock so that it cannot be mounted on a Model 11 drive. The 3336 Model 1 Disk Pack can be converted to a Model 11.

Table 50.10.1 compares Model 1, 2, and 11 drive characteristics. Table 50.10.2 compares 3336 Model 1 and 11 Disk Pack characteristics.

Table 50.10.1. Capacity and timing characteristics of 3330-series drives

Characteristic	3330-series Model 1 or 2 drive	3330-series Model 11 drive
Capacity in thousands of bytes (full-track records)	100,018	200,036
Seek time (ms)		
Maximum	55	55
Average	30	30
Average cylinder-to-cylinder	10	10
Time channel busy searching when SET SECTOR is used (ms)		
Minimum	.120	.120
Maximum	.380	.380
Rotation time (ms)	16.7	16.7
Rotation speed (rpm)	3600	3600
Data transfer rate (KB/sec)	806	806

Table 50.10.2 3336 Model 1 and 11 Disk Pack characteristics

Characteristic	3336 Model 1	3336 Model 11
Number of disks per pack	12	12
Number of recording disks	10	10
Number of recording surfaces	19	19
Disk thickness in inches	.075	.075
Disk diameter in inches	14	14
Disk pack weight in pounds	20	20
Disk pack maximum capacity in millions of bytes	100	200
Full track capacity in bytes	13,030	13,030
Cylinders per pack	404 plus 7 alternates	808 plus 7 alternates
Tracks per cylinder	19	19
Tracks per pack	7676	15,352

ATTACHMENT VIA INTEGRATED STORAGE CONTROLS

Optionally, one Integrated Storage Controls (ISC) feature can be installed on a Model 168 to attach 3330-series, 3340, 3344, or 3350 disk storage to one or two 2880 Block Multiplexer Channels. Attachment of these disk devices via 3830 Storage Control is possible as well. The following discusses attachment to the ISC of 3330-series strings only.

The Integrated Storage Controls feature includes dual direct access storage controls, each of which operates independently of the other and is functionally like 3830 Storage Control Model 2 except for the following:

- The Integrated Storage Controls feature is contained in the main frame of the Model 168 and is powered by it.

- The Two-Channel Switch, Additional feature (that provides four-channel switching) cannot be attached to the storage controls in the ISC feature.

Both logical storage controls in the ISC feature can be attached to the same 2880 channel or they can be attached to two different 2880 channels connected to the Model 168. Each logical storage control can have attached a maximum of four 3330-series strings of up to eight drives each. The 32 Drive Expansion and Control Store Extension optional features (field installable) must be installed in the ISC in order to attach more than two strings to each logical control. Therefore, up to 64 drives (eight strings) can be attached to the Model 168 via the ISC feature. The first module in each 3330-series string must be a 3333 Disk Storage and Control Model 1 or 11 unit.

The 3330-series drives attached to ISC operate just as if they were attached via 3830 Storage Control Model 2. That is, when multiple requesting is used, each logical storage control within the ISC can handle up to 32 channel programs concurrently, one on each of its drives, and only one of the 32 drives can be transferring data at a time. When a malfunction occurs, diagnostics can be run on one logical storage control and its drives, while normal operations take place on the other logical storage control in the ISC.

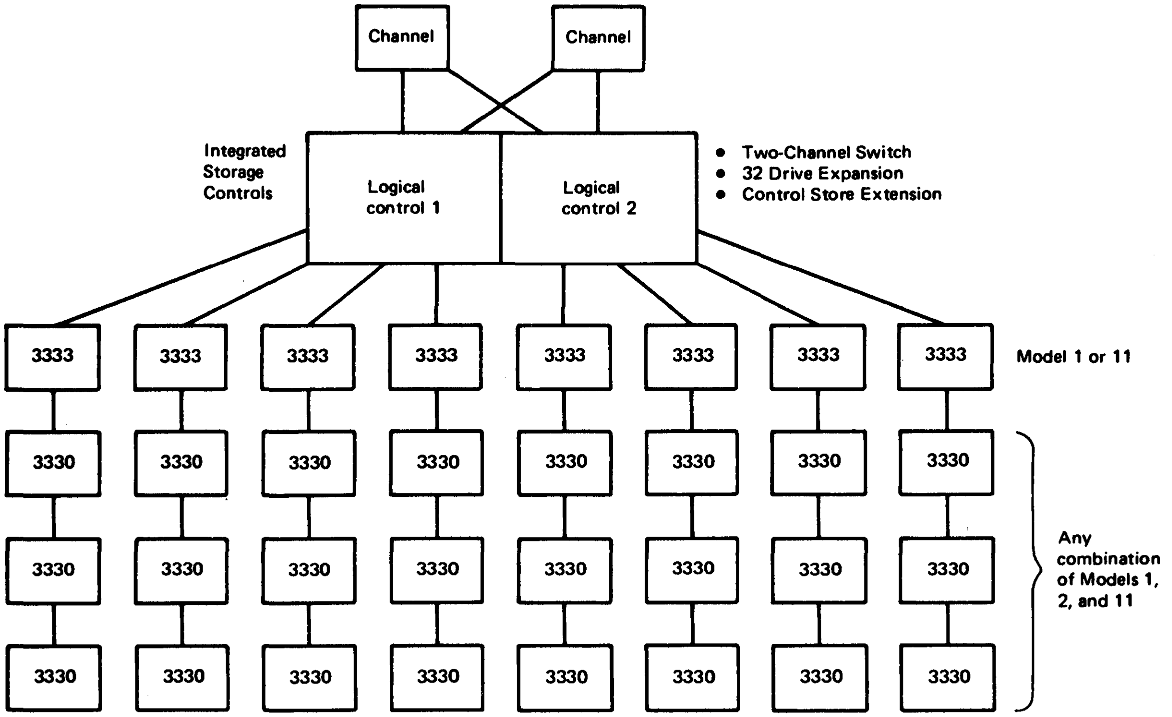
The ISC feature provides lower-cost attachment of 3330-series disk storage than 3830 Storage Control Model 2 when two storage control units are required, and floor space is saved since the ISC is in the Model 168 CPU. See Table 50.15.3 for a summary of the capabilities of the 3830 Models 1 and 2 and ISC.

The Two-Channel Switch optional feature is also available for the ISC feature. When installed, this feature provides a two-channel switching capability for both of the logical storage controls. The Two-Channel Switch feature permits each integrated storage control unit to be attached to two channels in the same Model 168 or to one channel in the Model 168 and one channel in another System/370. Figure 50.10.1 summarizes the 3330-series string configurations that are possible for the Model 168 ISC. Intermixing 3330-series and 3340 strings on an attachment is discussed in Section 50:15.

The 3333 String Switch optional feature can be installed on a 3333 Model 1 or 11 that is attached to the 3830 Model 2 or ISC. This field-installable feature enables the 3333 and all its attached 3330s (a 3330-series string) to be connected to two control-unit-type attachments instead of only one. The attachments can be any combination of two of the following:

- 3830 Storage Control Model 2
- Integrated Storage Controls for Models 158 and 168 (or the two logical controls in one ISC)
- Integrated Storage Control for the Model 145
- 3345 Storage and Control Frame Models 3, 4, and 5 for the Model 145
- 3330/3340 series IFA for the Model 135

The two attachments to which a 3333 with the 3333 String Switch feature is connected can be attached to the same or different channels in the same CPU, or to channels in two different CPUs. In addition, channel switching features can be installed on one or both of these attachments.



- One to four strings of from two to eight drives each connected to each logical control. Each logical control connected to one or two channels in the same or different CPU's.
- 3333 String Switch can be added to any or all 3333's to connect a 3333 to a second attachment in the same ISC, the same CPU, or another CPU except Models 115 and 125.

Figure 50.10.1. Permissible 3330-series string configurations for the Model 168 Integrated Storage Controls feature

The 3333 String Switch is functionally similar in its operation to the Two-Channel Switch. A switch can be set to allow the 3330-series string to be accessed via both attachments, one at a time. In effect, this setting provides two control unit paths to the string. String switching is accomplished dynamically under program control. Alternatively, the switch can be set to dedicate the string to one attachment or the other so that the string can be accessed only via that attachment.

Figure 50.10.2 illustrates 3333 string switching for four 3330-series strings. In the configuration shown, all strings can be accessed via two channels and two control units. Channel switching, string switching, and 32 Drive Expansion features can be used to enhance the availability of 3330-series disk storage and to extend backup capabilities when two System/370 systems (the same or different models) are present in an installation.

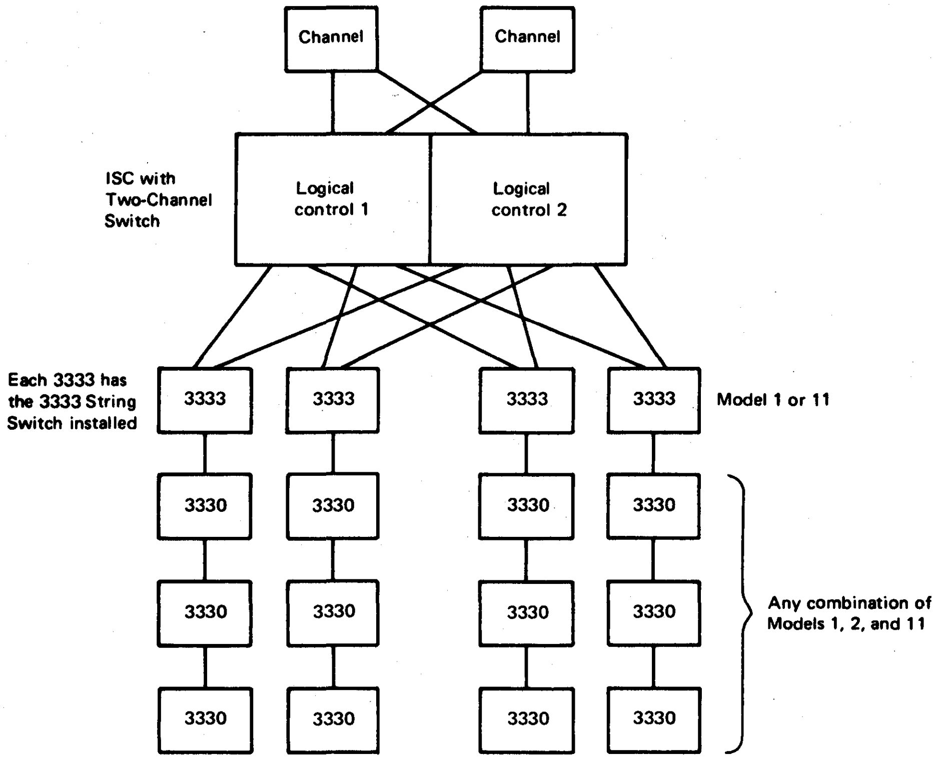


Figure 50.10.2. Sample 3330-series string configuration with string switching

Optionally, the staging adapter feature can be installed on the ISC to permit attachment of the 3850 Mass Storage System to the ISC. The ISC provides the same functions for the 3850 as 3830 Storage Control Model 3. The staging adapter permits the addressing capability of each of the four ISC paths to be expanded to a maximum of 64 unique addresses. When the staging adapter is installed, the control store extension feature must also be installed and 3340 disk storage cannot be attached to the ISC.

50:15 THE 3340 DIRECT ACCESS STORAGE FACILITY

3340 DISK STORAGE DRIVES AND THE 3348 DATA MODULE

The 3340 Direct Access Storage Facility is an intermediate capacity, modular, high-performance direct access storage subsystem that consists of 3340 Disk Storage and Control Model A2 and 3340 Disk Storage Models B1 and B2. A 3340 string can consist of from one to four units and is connected to a 2880 Block Multiplexer Channel in a Model 168 (Model 1, 3, or A3) configuration via 3830 Storage Control Model 2 or Integrated Storage Controls in the Model 168 CPU.

A 3340 string for the Model 168 can consist of from two to eight drives. A 3340 Disk Storage and Control Model A2 must be the first unit in a 3340 string. The 3340 Model A2 consists of two drives, drive-oriented control functions, and power for itself and the 3340 drives attached to it. In a Model 168 configuration, the 3340 Model A2 attaches to 3830 Storage Control Model 2 and a logical control in the ISC. Up to three units, any combination of 3340 Disk Storage Models B1 and B2, can be attached to a 3340 Model A2. The 3340 Model B2 consists of two drives and does not contain the power and device-oriented control functions that are part of the 3340 Model A2. The 3340 Model B1 contains one drive and no control functions. Functionally, all 3340 drives are alike regardless of whether they are part of a Model A2, B2, or B1 unit.

Figure 50.15.1 shows a 3340 string of five drives that includes one 3340 Model A2, one 3340 Model B2, and one 3340 Model B1. An operator control panel is located on the top of each 3340 drive. This panel contains the three-digit hexadecimal address of the drive, the switches required to operate the drive, and status indicator lights. The address of a 3340 drive is wired on a logic board in the 3340 unit.

The removable 3348 Data Module is used for data storage. Unlike the removable 2316 and 3336 disk packs that are the storage medium for 2314 and 3330-series disk storage, respectively, the 3348 Data Module is a sealed cartridge that contains a spindle, access mechanism, and read/write heads in addition to disks on which data is written and read. The cover of the data module, which is shock-absorbing and nonflammable, is never removed from the cartridge. The 3340 disk storage drive contains only the mechanical and electrical components that are required to house, load, air-filter, and drive the 3348 Data Module.

The 3348 Data Module is shown in Figure 50.15.2. The access mechanism in a 3348 Data Module is an L-shaped carriage that moves back and forth on a cylindrical shaft mounted within the data module. When the data module is not loaded, the access mechanism is latched in the home position so that it cannot move. In this position, the access mechanism is located such that the read/write heads rest on nondata areas on the disk surfaces.

Three models of the 3348 Data Module, all of which are the same physical size, are available. The 3348 Model 35 has a maximum capacity (assuming full-track records) of approximately 35 million bytes that are accessed by movable read/write heads. The 3348 Model 70 has a maximum capacity of approximately 70 million bytes that are accessed by movable read/write heads. The 3348 Model 70F also has a maximum capacity of 70 million bytes of which approximately 502,000 bytes maximum (60 logical tracks) are accessed by fixed read/write heads and the balance by movable read/write heads.

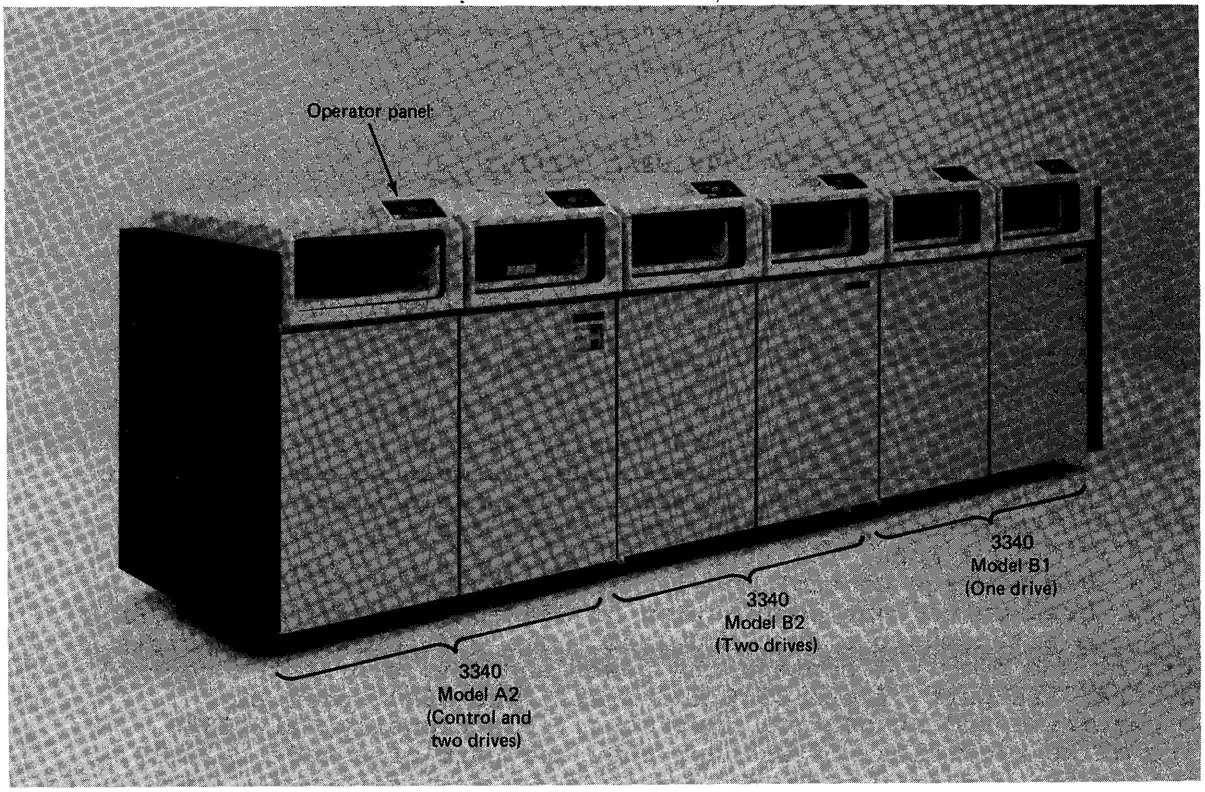


Figure 50.15.1. A five-drive 3340 string with 3340 Model A2, B2, and B1 units

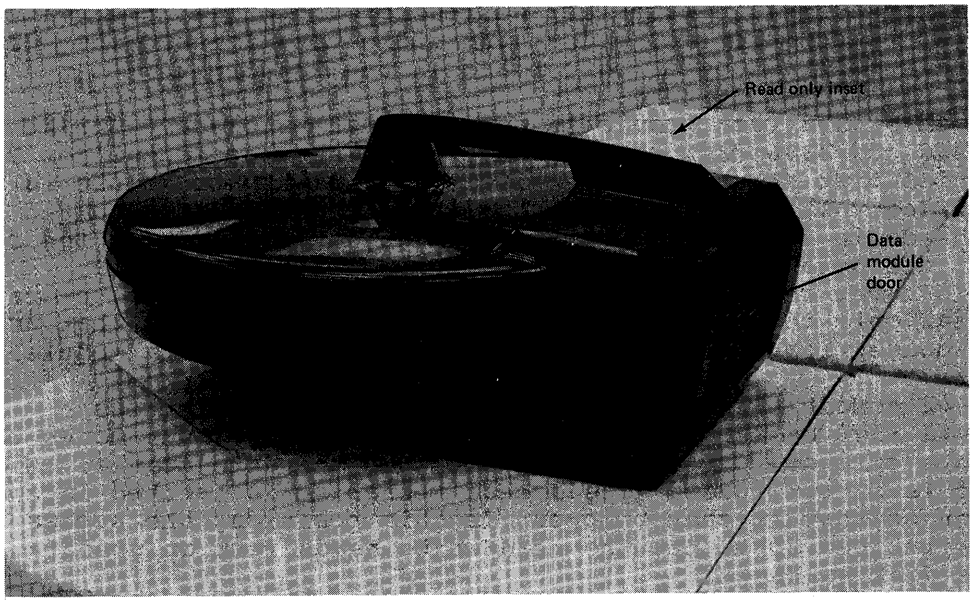


Figure 50.15.2. The 3348 Data Module

A purchased 3348 Model 35 can be upgraded to a Model 70 at the plant of manufacture. The upgrading of a 3348 Model 35 or 70 to a Model 70F and the alteration of a Model 70 to a Model 35 are not available as data module conversions.

The 3348 Model 70F can operate only on a 3340 drive (Model A2, B2, or B1) that has the optional field-installable Fixed Head feature installed. When installed on a 3340 A2 or B2 unit, the Fixed Head feature is installed on both drives. The presence or absence of this feature in a 3340 drive can be determined by programming at any time by issuing a SENSE command and inspecting the Fixed Head feature bit in the sense bytes read. The Fixed Head feature and the the Two-Channel Switch Additional feature (for four-channel switching) are mutually exclusive for the same 3340 string.

A Model 70F data module can be mounted on a 3340 drive that does not have the Fixed Head feature installed and made ready without any notification of the error by the hardware. However, the first I/O operation issued to the 3340 drive causes an intervention-required unit check condition and the drive is taken out of ready status. When this situation occurs in an OS/VS environment, a message is given to the operator and the affected job must be canceled in order to recover. To avoid such situations, it is recommended that 3340 units with and without the Fixed Head feature not be mixed within a string. If one 3340 unit has the feature, all should have the feature.

Models 35 and 70 of the 3348 Data Module can be used with any 3340 drive (Model A2, B2, or B1) whether or not it has the Fixed Head feature installed. No indication is given if a Model 35 or 70 is placed in a 3340 drive with the Fixed Head feature. In such cases, the fixed head capability of the drive is not utilized.

The 3340 Direct Access Storage Facility is unlike other System/370 direct access storage in that the capacity of an individual 3340 drive is determined by the model of 3348 Data Module mounted on the drive rather than by the model of the drive itself. The capacity of the 3348 Data Module that is mounted on a 3340 drive can be determined by programming at any time by issuing a SENSE command and inspecting the data module size bits in the sense bytes read.

The capability of having two capacity options per drive means the capacity of a 3340 string can be increased by using larger capacity data modules on existing drives as well as by adding drives to the string. A 3340 string can vary in capacity from 70 million bytes (two Model 35 data modules) to a maximum capacity of 560 million bytes (eight Model 70 or 70F data modules) in 35- and/or 70-million-byte increments (assuming full-track records).

Reliability and the Sealed Cartridge Design

The sealed cartridge design of the 3348 Data Module, the advanced design used for the read/write heads in the data module, and improvements in the physical design of the 3340 drive make the 3340 Direct Access Storage Facility more reliable than previously announced direct access storage devices for System/370, as explained below. No preventive maintenance is scheduled for a 3340 facility because of its reliability features.

Reliability is improved by the removal of head-to-disk alignment problems. Each read/write head within a 3348 Data Module is dedicated to certain tracks on one data surface. Therefore, each head reads only the data it wrote previously, regardless of the 3340 drive that is used. Since common head alignment across all 3340 drives is not required, the critical alignment tolerances that are normally necessary to achieve data interchangeability among drives are not needed for 3348 Data Modules. It is the less critical alignment tolerances for the read/write heads in a 3348 Data Module that minimize the chance of errors caused by incorrect alignment of a head to its dedicated tracks.

There is also less chance of damaging read/write heads. If a data module is dropped, the only read/write heads that can be affected are those in that data module. If a disk pack is damaged, it can cause damage to the read/write heads in more than one drive if it is moved from drive to drive in an attempt to find a drive that can read the pack. The outside covers of a 3348 Data Module are made of a highly durable material that is designed to enable a data module to withstand more severe blows without damage than can a disk pack.

Reliability is improved because the exposure of the disk surfaces in a 3348 Data Module to outside contamination is greatly reduced when compared to the contamination exposure of a disk pack. A 3348 Data Module is opened only when it is mounted on a 3340 drive and only when the drive cover is closed. Contamination on disk surfaces can be a major cause of head and disk damage.

In addition, the possibility of head crashes is minimized by the improved flying characteristics of the read/write heads in a data module. The low mass of the read/write heads and the low loading force used enable the heads to fly over the rotating disks at a very low height. This near contact (or proximity) recording capability of the read/write heads in the 3348 permits smaller bits to be written, which increases the recording density that can be achieved.

The recording density in bits per inch of a track in a 3348 Data Module is approximately 2.5 times greater than the recording density of a track in a 2316 pack (10 percent greater than 3330-series Model 11 density and more than two times greater than 3330-series Model 1 and 2 density). The advanced head design used for the 3348 Data Module enables greater density to be achieved together with improved reliability.

Reliability of the 3340 Direct Access Storage Facility is also improved because many critical mechanical parts have been eliminated, such as a complex head load/unload mechanism. In other cases, electronic functions have replaced mechanical functions. While the 3340 drive contains more electronics than the 2314, higher density logic cards are used in the 3340, which results in significantly fewer logic cards. (A 3340 drive also contains approximately one-third the number of logic cards as a 3330-series drive.)

The sealed cartridge design implemented in the 3348 Data Module provides several advantages in addition to improved reliability, such as simplified data module loading and unloading. Operations that are required for disk pack loading and unloading (tightening the pack on the spindle, cover removal, cover replacement, untightening the pack for removal) are not required for a 3348 Data Module. In addition, the possibility of hub wear or hub damage as a result of loading and unloading operations is eliminated for a 3348 Data Module.

After the top cover of the 3340 drive to be used is raised, the operator places the data module in the exposed drive shroud recess. After closing the cover, the operator initiates automatic loading of the module by putting the start/stop switch on the operator panel of the drive in the start position. This causes the cover of the drive to be locked, which is indicated by a light on the operator panel, and the data module to be loaded.

The following occurs during data module loading. The shroud containing the seated data module moves to the back of the 3340 drive where the voice coil motor is located. While the data module is in motion, the data module door in the rear of the 3348 is rolled down. Electrical, mechanical, and filtered air connections between the 3348 Data Module and the 3340 drive are then made through the open data module door. The access mechanism is then unlatched and the disks are brought up to rotational speed. The access mechanism is moved to physical track 0. This entire loading process requires approximately 20 seconds. When the loading process is completed, the ready light on the operator panel is turned on to indicate the 3348 Data Module is ready for processing.

To unload a data module, the operator places the start/stop switch in the stop position. The unloading procedure consists of a reversal of the operations performed during loading. The access mechanism moves to the home position in the data module, where it is latched, disk rotation is stopped, the data module is disconnected from the drive, the data module door is closed, and the data module moves to the front of the drive. The cover-locked indicator light is turned off as soon as the unloading procedure is completed. Unloading requires approximately 20 seconds. The cover of the 3340 drive can be raised as soon as the cover-locked indicator light is turned off and the 3348 Data Module can then be removed.

The possibility of contaminating the disk surfaces of a data module during loading and unloading operations is minimized because the data surfaces are exposed to the air within the closed 3340 drive through the open data module door for only slightly more than one second. Further, as soon as a seal between the 3340 drive and the 3348 Data Module has been made, the filtered air system displaces the air within the data module several times to remove any contaminants that may have entered via the open data module door.

The sealed cartridge also offers two other unique features. First, a read-only function (not available for the 2314) is provided on a data module basis rather than a drive basis (as implemented for 3330-series disk storage). The read-only function is enabled for a 3348 Data Module by turning an inset in the handle of the 3348 (see Figure 50.15.2) to the read-only position before placing the data module in the 3340 drive. This inset causes the read-only switch that is part of each 3340 drive and the read-only indicator on the operator panel to be turned on when the 3348 is loaded in a 3340 drive.

When the read-only function is enabled for a 3348 Data Module and an attempt is made to write on the data module, an interruption occurs and IBM-supplied programming support terminates the program that issued the write. The advantage of this approach is that once the read-only inset in a 3348 Data Module is set to inhibit writing, the data module can be used with any 3340 drive at any time and the operator need not remember to turn on a read-only switch on the drive.

Second, external label handling is improved. An external label can be placed on a 3348 Data Module after it is removed from the 3340 drive. Placing an external label on the top surface of a disk pack instead of on the cover, to avoid mislabeling a disk pack by placing the wrong cover on it, can be done only when the disk pack is mounted on a drive. In addition, since the outside cover is never removed from a data module, the volume identification label on the cover is legible through the front window of the cover of the 3340 drive even when the data module is loaded and being accessed.

Layout of Tracks, Cylinders, and Read/Write Heads in 3348 Data Modules

The layout of physical and logical tracks on a data surface of any model 3348 Data Module and the relative position of the read/write heads for a data surface are shown in Figure 50.15.3. A data surface contains 700 physical tracks with a small space between the first 350 physical tracks and the second 350 physical tracks. There is also unused space after the second group of 350 physical tracks. Two logical tracks, one even-numbered and one odd-numbered, are written on each physical track. A logical track has a maximum capacity of 8368 data bytes (for full-track records).

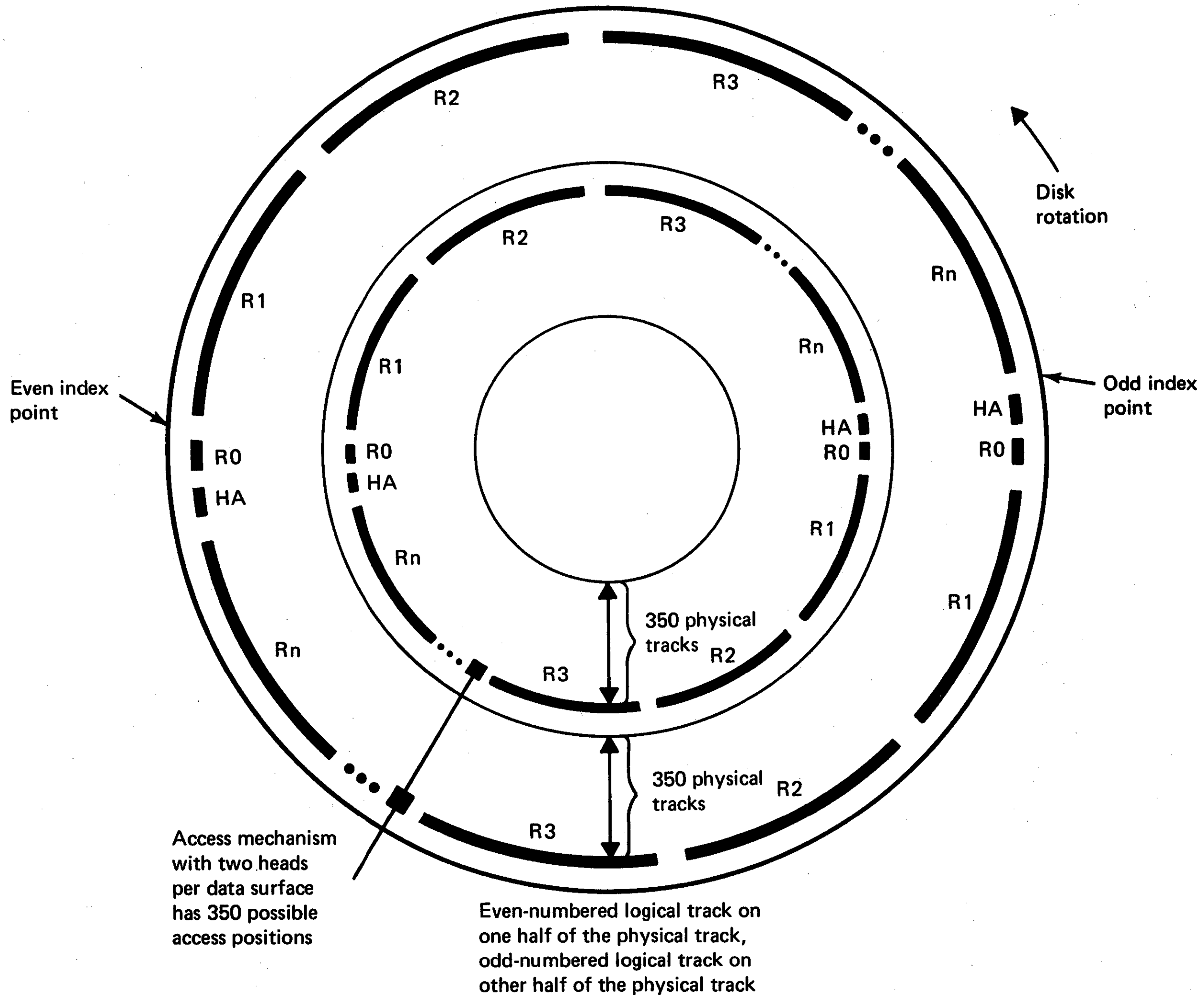

R2
R3
Disk rotation
R2
R3
Rn
R1
Rn
R1
Even index point
Odd index point
R0
HA
R0
HA
HA
R0
HA
R0
Rn
R1
R1
Rn
350 physical tracks
R2
Rn
R3
350 physical tracks
R2
R3
Access mechanism with two heads per data surface has 350 possible access positions
Even-numbered logical track on one half of the physical track, odd-numbered logical track on other half of the physical track

Figure 50.15.3. Location of physical and logical tracks and read/write heads on a data surface in a 3348 Data Module

There are two read/write heads associated with each data surface. They are positioned a little more than 350 physical tracks apart, as shown in Figure 50.15.3. While starting and stopping the data module, the read/write heads are positioned over the unused portions of the data surface.

The access mechanism can be placed at any one of 350 access positions on the data surface. Therefore, an outermost head on the access mechanism can access physical tracks 0 to 349 on its associated data surface while an innermost head can access physical tracks 350 to 699. At any of the 350 possible access mechanism positions, two physical tracks (four logical tracks) can be accessed on a data surface. However, only one read/write head in a data module can be active at a time.

The bottommost surface in all 3348 Data Modules is used as the servo surface. This surface contains information for the servo system that is used to control seek operations, positioning of the heads over tracks, data clocking (the synchronization of data with rotational speed during writing operations), index generation, and signal generation required by the RPS feature. Functionally, the 3340 servo system is like that used in 3330-series drives. However, design improvements, such as elimination of the electromechanical tachometer, have been made.

The required servo information is prerecorded on the servo surface of each 3348 Data Module at the plant of manufacture and is read by a servo read head at the bottom of the access mechanism. The servo information on this surface cannot be read or written using 3340 commands. The servo surface on a 3348 Model 70F Data Module also contains the 60 logical tracks that are read by the fixed heads.

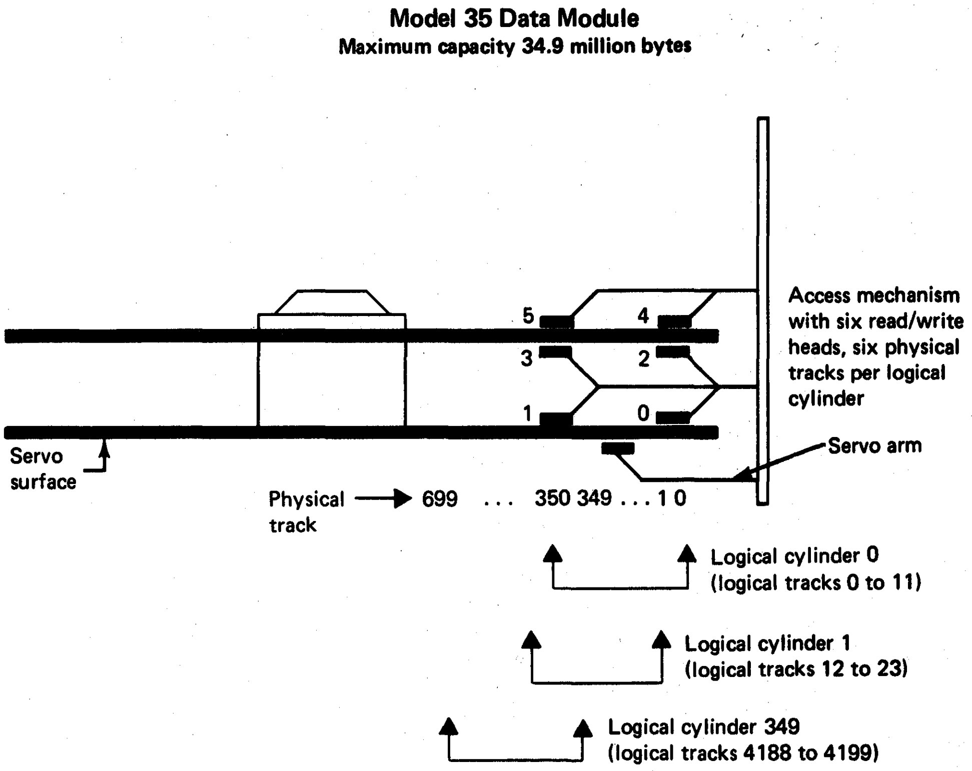
The access mechanism in a 3348 is driven by a voice-coil motor. This motor and the servo system provide fast, precise access mechanism positioning, which minimizes head settling time.

Figure 50.15.4 shows the layout of cylinders and read/write heads for the 3348 Model 35 Data Module. A Model 35 contains two recording disks. Three of the data surfaces on the two recording disks are used for data recording in a Model 35 Data Module. The three data surfaces are accessed by six read/write heads (0 to 5). The six physical tracks that can be accessed at any given position of the access mechanism constitute a logical cylinder and contain twelve logical tracks. Head 0 accesses logical tracks 0 and 1, head 1 accesses logical tracks 2 and 3, etc.

A four-byte field (CCHH) is used to address the logical tracks in a 3348 Data Module. The two-byte CC (cylinder address) field specifies the logical cylinder address, which can be 0 to 348 for the primary and alternate logical tracks of a Model 35 Data Module. The two-byte HH field, which normally specifies the actual head address (for 2314 and 3330-series drives, for example), specifies the number of the logical track within the logical cylinder, a value from 0 to 11, instead of a head address of 0 to 5. The drive selects the appropriate head, using the logical track number.

In Figure 50.15.4, the access mechanism is shown positioned at logical cylinder 0, where physical tracks 0 and 350 on each of the three data surfaces can be accessed. There are 350 logical cylinders in the Model 35 Data Module. The first 348 are used for data, logical cylinder 348 is the alternate cylinder, and logical cylinder 349 is the CE cylinder. The CE cylinder is designed to be used only by the CE for testing the read/write capability of a 3340 drive. It contains a prewritten area for read testing and an area in which write tests can be performed.

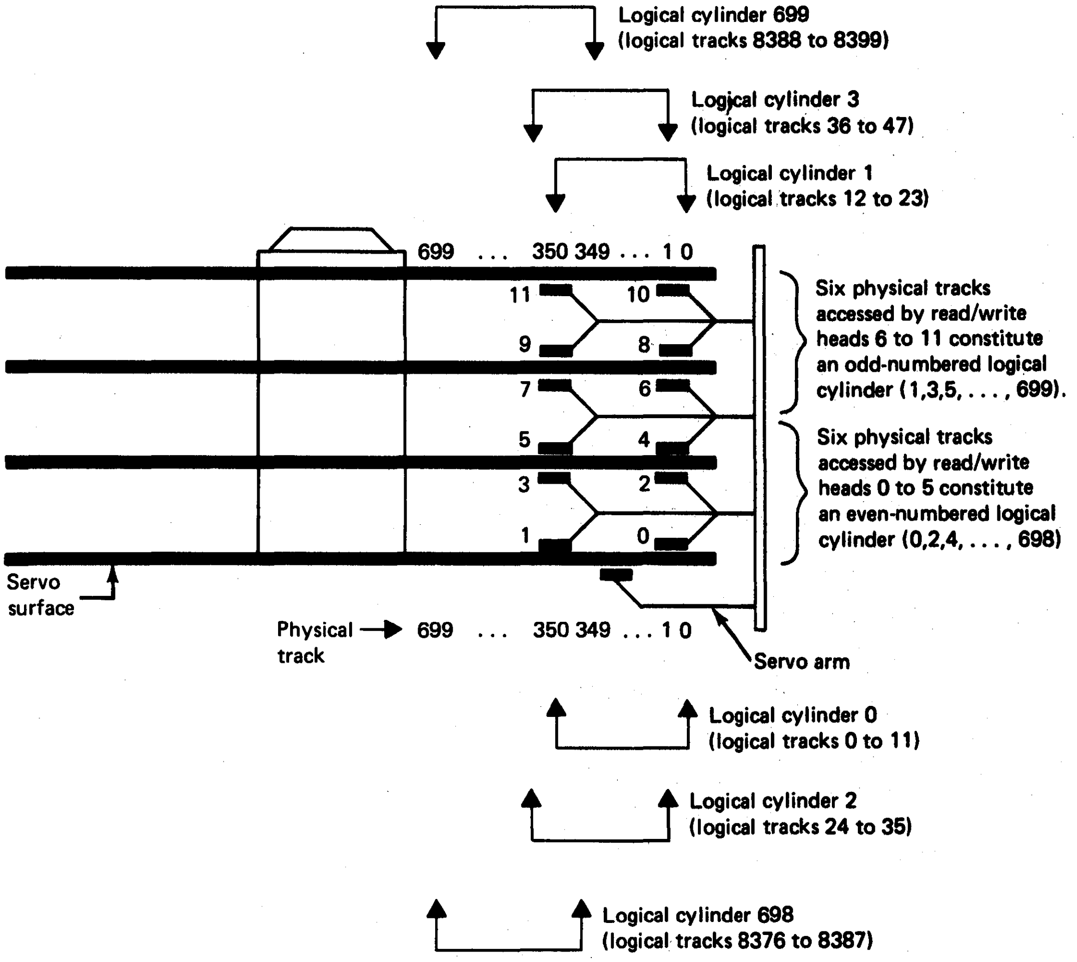
Figure 50.15.5 shows the layout of cylinders and read/write heads for the 3348 Model 70. A Model 70 contains four recording disks. Six data surfaces on the four recording disks, each of which is accessible by two read/write heads, are used for data recording in the Model 70. As for the Model 35, the six physical tracks that can be accessed by the lower six read/write heads (0 to 5) at a given position of the access mechanism constitute a logical cylinder of twelve logical tracks. In a Model 70, however, the logical cylinders addressed by read/write heads 0 to 5 are all even-numbered (0, 2, 4, ..., 698). The six physical tracks that can be accessed by the upper six read/write heads (6 to 11) at a given position of the access mechanism also constitute a logical cylinder of twelve logical tracks. The logical cylinders addressed by read/write heads 6 to 11 are all odd-numbered (1, 3, 5, ..., 699). Thus, on a Model 70 two logical cylinders (24 logical tracks) can be accessed at each of the 350 possible access mechanism positions.


Model 35 Data Module
Maximum capacity 34.9 million bytes
5
4
3
2
1
0
Access mechanism with six read/write heads, six physical tracks per logical cylinder
Servo arm
Servo surface
Physical track
699 ... 350 349 ... 1 0
Logical cylinder 0 (logical tracks 0 to 11)
Logical cylinder 1 (logical tracks 12 to 23)
Logical cylinder 349 (logical tracks 4188 to 4199)

Number of recording disks	2	
Number of data surfaces	3	
Number of read/write heads	6	
Number of physical tracks per physical cylinder	6	
Number of physical tracks per logical cylinder	6	
Number of logical tracks per logical cylinder	12	
Number of logical cylinders per data module	350	
Number of logical tracks per data module	4200	(4176 data) (12 alternate) (12 CE)
Number of access mechanism positions	350	
Number of logical cylinders accessed per access mechanism position	1	

Figure 50.15.4. Cylinder and read/write head layout for a 3348 Model 35 Data Module

Model 70 Data Module
Maximum capacity 69.8 million bytes


Logical cylinder 699
(logical tracks 8388 to 8399)
Logical cylinder 3
(logical tracks 36 to 47)
Logical cylinder 1
(logical tracks 12 to 23)
699 . . . 350 349 . . . 1 0
11
10
9
8
7
6
5
4
3
2
1
0
Six physical tracks accessed by read/write heads 6 to 11 constitute an odd-numbered logical cylinder (1,3,5, . . . , 699).
Six physical tracks accessed by read/write heads 0 to 5 constitute an even-numbered logical cylinder (0,2,4, . . . , 698)
Servo surface
Physical track
699 . . . 350 349 . . . 1 0
Servo arm
Logical cylinder 0
(logical tracks 0 to 11)
Logical cylinder 2
(logical tracks 24 to 35)
Logical cylinder 698
(logical tracks 8376 to 8387)

Number of recording disks	4	
Number of data surfaces	6	
Number of read/write heads	12	
Number of physical tracks per physical cylinder	12	
Number of physical tracks per logical cylinder	6	
Number of logical tracks per logical cylinder	12	
Number of logical cylinders per data module	700	
Number of logical tracks per data module	8400	(8352 data) (24 alternate) (24 CE)
Number of access mechanism positions	350	
Number of logical cylinders accessed per access mechanism position	2	

Figure 50.15.5. Cylinder and read/write head layout for a 3348 Model 70 Data Module

There are 700 logical cylinders in the Model 70 data module. The first 696 (0-695) are used for data. Logical cylinders 696 and 697 are used as alternate logical cylinders while logical cylinders 698 and 699 are CE cylinders. The method of addressing a logical track in a Model 70 data module is the same as described for a Model 35. The CC value can vary from 0 to 697 for data and alternate logical cylinders while the HH value can vary from 0 to 11.

Figure 50.15.6 shows the layout of cylinders and read/write heads for the 3348 Model 70F. This model is identical to the Model 70 except for the following. Seven surfaces, six data surfaces and the servo surface, on the four recording disks are used for data recording. Logical cylinders 1 to 5 are recorded on the servo surface. They are written on 30 physical tracks that are accessed by 30 fixed read/write elements, which are mounted on a plate under the servo surface, as shown in Figure 50.15.6. The first six physical tracks contain logical cylinder 1, the second six physical tracks contain logical cylinder 2, etc. Logical cylinders 0 and 6 to 699 are recorded on the six data surfaces just as in a Model 70 data module.

Addressing a logical track in a Model 70F data module using a CCHH field is the same as described for the Model 70. When a command is received that addresses a logical track in logical cylinders 1 to 5 of a Model 70F, the 3340 drive automatically selects the fixed read/write element associated with the specified logical track instead of the movable head. Therefore, a Model 70F and a Model 70 data module can be accessed using the same 3340 channel programs. This means no special programming support is required to use a Model 70F instead of a Model 70.

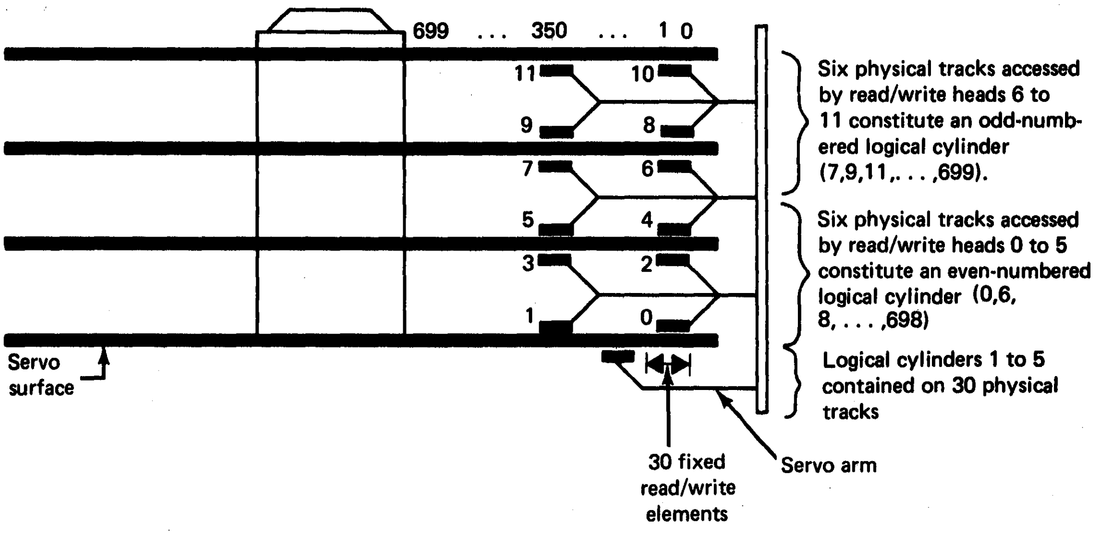
The physical tracks that contain logical cylinders 1 to 5 in a Model 70 are not used in a Model 70F and cannot be accessed by the user or a customer engineer because of the way in which head selection is performed. Hence, the data capacity of Models 70F and 70 is the same. Seek time for logical cylinders 1 to 5 in a Model 70F is zero. Seek times for logical cylinders 0 and 6 to 695 in a Model 70F are the same as Model 70 seek times.

A data set or file can be contained both in logical cylinders 1 to 5 of a Model 70F data module and in logical cylinders that are accessed by movable heads. A 3340 drive, however, can perform only one operation at a time. Therefore, a seek, search, or data transfer operation involving a fixed head in a Model 70F data module cannot be performed at the same time a movable head is involved in a seek, search, or data transfer operation.

The best performance gains can be achieved when Model 70F data modules are used by assigning the fixed head logical tracks to small active system data sets (such as the page data set, system catalog, TCAM message queue), small active user data sets, large active data sets that can be segmented (OS/VS1 page data set, partitioned data sets, ISAM index levels, for example), and data sets with major activity concentrated at the beginning of the data set (such as the OS/VS job queue).

The assignment of such data sets to the fixed head logical tracks in a Model 70F data module is a user responsibility. OS/VS DD statements for these data sets must specifically request by actual address locations within the fixed head logical cylinders. Note also that the device type code in the device table that is generated in the control program during a system generation (OS/VS UCB table) does not differentiate between 3340 drives with and without the Fixed Head feature. Therefore, if generic device type assignment by device type (3340) is used in a configuration that contains 3340 drives with and without the Fixed Head feature, either type drive can be selected by the operating system.

Model 70F Data Module
Maximum capacity 69.8 million bytes



Six physical tracks accessed by read/write heads 6 to 11 constitute an odd-numbered logical cylinder (7,9,11,. . . ,699).

Six physical tracks accessed by read/write heads 0 to 5 constitute an even-numbered logical cylinder (0,6, 8, . . . ,698)

Logical cylinders 1 to 5 contained on 30 physical tracks

Number of recording disks	4	
Number of data surfaces	6	plus servo surface
Number of read/write heads	12	movable
	30	fixed
Number of physical tracks per physical cylinder	12	
Number of physical tracks per logical cylinder	6	
Number of logical tracks per logical cylinder	12	
Number of logical cylinders per data module	700	
Number of logical tracks per data module	8400	(8352 data - 60 fixed head and 8292 movable head) (24 alternate) (24 CE)
Number of movable head access mechanism positions	350	
Number of logical cylinders accessed per access mechanism position	2	except for first 3 positions

Figure 50.15.6. Cylinder and read/write head layout for a 3348 Model 70F data module

The assignment of a 3340 drive with the Fixed Head feature can be assured in an OS/VS environment by specifying a user-defined device class name for such 3340 drives at system generation and using this name (instead of UNIT=3340) in the appropriate DD statements.

Alternate tracks that are accessed by fixed heads are not provided for logical cylinders 1 to 5 in a Model 70F data module. Logical cylinders 696 and 697, which provide alternate tracks for the logical tracks accessed by the movable heads, also provide alternate tracks for the logical tracks in logical cylinders 1 to 5. This approach is taken because the probability a fixed head track in logical cylinders 1 to 5 will develop a defect is lower than that for movable head tracks and the possibility of a defect occurring in a movable head track is very low (for the reasons discussed later).

The low probability of defects occurring in fixed head logical cylinders 1 to 5 of a Model 70F data module results in part from the fact that these cylinders are recorded on the servo surface, which is a specially manufactured surface because of its primary function. In addition, the fixed head tracks are recorded on the outer edge of the servo surface, which results in a lower bit density for these tracks. The width of a fixed head physical track is six times greater than that of a movable head track on a data surface.

If an uncorrectable error does occur on a fixed head logical track in a Model 70F data module, the logical track should be flagged and an alternate track should be assigned. This can be done using the IEHATLAS, IEHDASDR, or IBCDASDI utility of OS/VS. IEHDASDR or IBCDASDI should then be used to test the flagged fixed head track to determine whether the track is really defective. If the track is found not to be defective, the flag is removed and the assigned alternate track is released. If the track is defective, the data module can be returned to the plant of manufacture for repair if the loss of performance resulting from using an alternate movable head track instead of the fixed head track is not acceptable.

The physical and capacity characteristics of 3348 Data Modules and the 2316 disk pack are given in Table 50.15.1. Table 50.15.2 gives the timing characteristics of the 3340 direct access storage facility and the 2314 facility.

Track Formatting and Data Module Initialization

Self-formatting records consisting of count, key, and data or count and data areas are written on the logical tracks of a 3348 Data Module just as on the tracks of a 2316 pack. However, each home address, count, and key area written on a 3348 track has a six-byte detection code field appended to it for data validity checking by the 3830 Model 2 or integrated storage control. The detection code used can detect all single-error bursts of eleven bits span or less.

A six-byte correction code field is appended to each data area written on a 3348 track. The correction code used has the same detection capability as the detection code and the capability of correcting single-error bursts of three bits span or less. The actual error correction procedure must be performed by programming (error recovery routines) using corrective bits that are supplied by the control unit as discussed later.

Table 50.15.1. Physical and capacity characteristics of 3348 Data Modules and the 2316 Disk Pack

Characteristic	3348 Model 35	3348 Model 70	3348 Model 70F	2316
Number of data disks per data module/pack	2	4	4	11
Disk diameter in inches	14	14	14	14
Number of surfaces used per data module/pack	3 data 1 servo	6 data 1 servo	6 data 1 servo and data	20 data
Number of read/write heads per recording surface	2	2	2 plus 30 read/write elements for the servo surface	1
Number of cylinders per data module/pack	348 plus 1 alternate and 1 CE	696 plus 2 alternates and 2 CE	696 plus 2 alternates and 2 CE	200 plus 3 alternates
Number of logical tracks per cylinder	12	12	12	20
Number of data tracks recorded per data module/pack	4,176	8,352	8,352	4,000
Full track capacity in bytes	8,368	8,368	8,368	7,294
Cylinder capacity in bytes	100,416	100,416	100,416	145,880
Maximum capacity in bytes per data module/pack	34,947,768	69,889,536	69,889,536 (502,080 in logical cylinders 1 to 5, 69,387,456 in logical cylinders 0 and 6 to 695)	29,176,000
Data module/pack weight in pounds	17	19.5	20	15

Table 50.15.2. Timing characteristics of the 3340 direct access storage facility and the 2314 facility

Characteristic	Models 35 and 70	Model 70F		2314
		Cylinders 1-5	Cylinders 0, 6-699	
Seek time (ms)				
Maximum	50 (350 cyl-Model 35) (700 cyl-Model 70)	0	50 (700 cylinders)	130
Average	25 (350 cyl-Model 35) (700 cyl-Model 70)	0	25 (700 cylinders)	60
Cylinder to cylinder				
Model 35	10			25
Models 70, 70F	Even to next odd - 0	0	0	
	Even to next even - 10	0	10	
	Odd to next even or odd - 10	0	10	
Rotation time (ms)	20.2	20.2	20.2	25
Rotation speed (rpm)	2964	2964	2964	2400
Data transfer rate (KB/sec)	885	885	885	312
Sectors per track	64	64	64	--
Sector time (microseconds)	316	316	316	--
Load time (secs) (time to ready status after mounting)	20	20	20	60
Unload time (secs)	20	20	20	15

The home address and count areas written on a logical track in a 3348 contain two new fields in addition to the same fields as are written in home address and count areas on 2316 tracks. The home address and each count area on a 3348 logical track contain a two-byte skip defect field and a two-byte physical address field in front of the flag byte. The automatic surface defect skipping capability of the 3340 allows valid data to be written before and after a surface defect on a logical track. The skip defect bytes are used to indicate the location of the center of the surface defect relative to the index point of the logical track. Bits in the flag byte field indicate whether the surface defect is located in the next count, key, or data area.

Surface defect skipping is implemented by including in each logical track of a 3348 Data Module a reserved area called a surface defect gap in which no data is written. If a logical track has no surface defects, the surface defect gap is located at the end of the logical track. If there is a surface defect, the surface defect gap is placed over the defective portion of the logical track at the time of manufacture. One or more surface defects that together occupy an area of up to 16 bytes in length per logical track can be handled by the defect skipping technique while the stated full logical track capacity of 8368 bytes is maintained.

The error detection and correction code capabilities of the 3340 facility permit successful recovery from an error within the data portion of a physical record even when it contains a surface defect gap.

Partial initialization of all 3348 Data Modules is performed at the plant of manufacture. A home address record and track descriptor (R0) record are written on each logical track in the data module. If a single skippable defect is found during the analysis of the surface of a logical track, the appropriate SD bytes and flag byte are written in the home address to indicate this fact. If no surface defect is found, the SD bytes are written as zeros.

The SD bytes and flag byte are supplied in the count area field in virtual storage only for a WRITE HOME ADDRESS command. When R0 is written during data module initialization and thereafter whenever a formatting write is performed, the SD and flag bytes for the count area to be written on disk are supplied by the control unit, which reads them from the record immediately preceding the record to be written.

When a record is written with a formatting write command on the portion of a logical track that contains an identified surface defect, the defect gap area is maintained in the defective portion of the logical track and data is written before and after the defect gap as appropriate. Whenever a nonformatting write or a read is issued for this record, the surface defect gap is automatically skipped over by the hardware without programming assistance or any error notification, just as if no surface defect existed.

The OS/VS IBCDASDI, IEHDASDR, or IEHATLAS utilities can be used to assign an alternate track if a physical track becomes defective during its use in an installation. If data cannot be read from a 3348 Data Module and recovery of this data is critical, the data module can be returned to the plant of manufacture where recovery will be attempted.

The two physical address bytes in home address and count areas on a 3348 logical track contain the physical cylinder and track address of the logical track on which they are written. When a seek command is issued, the control unit converts the logical cylinder and track address specified by the seek command to a physical cylinder and track address that is actually used by the drive in the seek operation. This physical address is saved in the control unit for later use in seek verification.

The physical address bytes are automatically written and read by the control unit and are not processed by programming. That is, when a home address or count area is written, the physical address bytes are automatically supplied by the control unit and are not contained in the home address or count area field in virtual storage that is indicated by the write command. Similarly, when a home address or count area is read, the control unit reads the physical address bytes but they are not placed in the home address or count field area in virtual storage.

The physical address bytes are used by the control unit for seek verification during normal operations and by the 3340 microdiagnostic routines. When a home address or count area is processed during a read, search, or clock operation, the physical address bytes read are compared with the most recent seek address (physical cylinder and track address) that was saved in the control unit when the last seek command was issued. If the two physical addresses are not equal, the command is terminated and a unit check condition results. Seek check is indicated in the sense bytes.

ATTACHMENT VIA 3830 STORAGE CONTROL MODEL 2

The 3830 Storage Control Model 2 unit contains the control functions required to operate one or two 3340 strings of from two to eight drives each. If the 32 Drive Expansion and Control Store Extension optional features are installed on a 3830 Model 2, up to four 3340 strings of from two to eight drives each can be attached to it. These two features are field-installable. A maximum of two of the 3340 strings attached to the 3830 Model 2 can contain 3344 units.

Cabling between the 3830 Model 2 and the 3340 Model A2 can be a maximum of 150 feet in length. The 3830 Model 2 attaches to a 2880 Block Multiplexer Channel in the Model 168 configuration via cabling up to 150 feet in length. Figure 50.15.7 shows a Model 168 configuration with 3340 strings attached via 3830 Storage Control Model 2. Intermixing 3340 and 3330-series strings on an attachment is discussed later in this subsection.

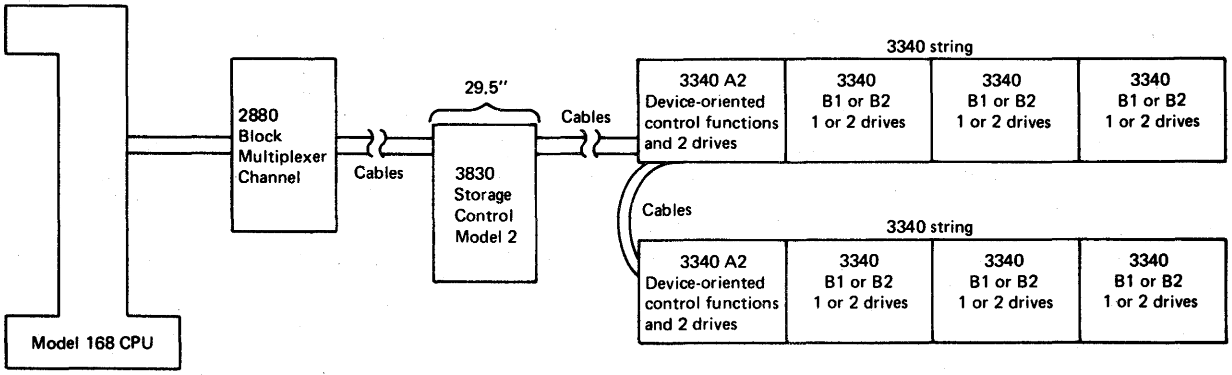

Model 168 CPU
2880 Block Multiplexer Channel
Cables
29.5"
3830 Storage Control Model 2
Cables
Cables
3340 string
3340 A2 Device-oriented control functions and 2 drives
3340 B1 or B2 1 or 2 drives
3340 B1 or B2 1 or 2 drives
3340 B1 or B2 1 or 2 drives
3340 string
3340 A2 Device-oriented control functions and 2 drives
3340 B1 or B2 1 or 2 drives
3340 B1 or B2 1 or 2 drives
3340 B1 or B2 1 or 2 drives

Figure 50.15.7. A Model 168 configuration with 3340 disk storage attached via 3830 Storage Control Model 2

Standard features of the 3830 Model 2 when used with 3340 disk storage are record overflow, multiple requesting, and rotational position sensing. The command retry facility of the 3830 Model 2 that is implemented for 3330-series drives is not implemented for 3340 drives. When multiple requesting is used, the 3830 Model 2 can control concurrent operation of up to 32 channel programs (when 32 Drive Expansion is installed), one on each of its drives. Only one of the 2 to 32 drives attached to a 3830 Model 2 can be transferring data at a time.

Rotational position sensing is an optional field-installable feature for 3340 units. It must be installed on each unit (both drives in an A2 or B2 3340 unit) that is to use the standard rotational position sensing capability of the 3830 Model 2. For performance reasons (see Section 60:10 in A Guide to the IBM System/370 Model 165, GC20-1730), it is recommended that the RPS feature be installed on all of the 3340 units in a given string or on none of the units in the string. The presence or absence of the RPS feature in a 3340 drive can be determined by programming at any time by issuing a SENSE command and inspecting the RPS feature bit in the sense bytes read.

If a SET SECTOR command is issued to a 3340 drive that does not have the RPS feature installed, no operation is performed, track orientation is lost, and channel end and device end status are presented. If a READ

SECTOR command is issued to a 3340 drive without RPS installed, a sector value of zero is returned together with channel end and device end status. Thus, channel programs containing sector commands can operate on 3340 drives that do not have RPS installed.

The 3830 Model 2 supports all the 2314 commands (except the file scan commands) in addition to new commands not available for the 2314, such as RPS and diagnostic commands. The command set for the 3340 is the same as that for 3330-series disk storage.

The Two-Channel Switch feature, identical in function to the same feature for the 2314 facility, can be installed on a 3830 Model 2 to allow it to be attached to two channels. The Two-Channel Switch, Additional feature can be added to this configuration to permit the 3830 Model 2 to be attached to four channels. A maximum of two of the four channels can be present in the same system. The channels to which a 3830 Model 2 with one or both of these features is connected must each have one control unit position and, if block multiplexing is to be used, eight nonshared subchannels available. An enable/disable switch on the 3830 Model 2 can be set to dedicate the 3830 to any subset of the two to four channels.

The optional String Switch feature can be installed on 3340 Model A2 drives. This field-installable feature enables the 3340 Model A2 and its attached Model B2 and B1 units to be connected to two control-unit-type attachments instead of only one. The attachments can be any two of the following:

- 3830 Storage Control Model 2
- Integrated Storage Control for the Model 145 or 148
- 3345 Storage and Control Frame Models 3, 4, and 5 for the Model 145
- Integrated Storage Controls for Models 158 and 168 (or the two logical controls in one ISC)
- 3330/3340-series IFA for the Model 135 or 138
- Direct Disk Attachment of a Model 115 Model 2 or Model 125 Model 2

Except for the Direct Disk Attachment, the two attachments to which a 3340 Model A2 with the String Switch feature is connected can be attached to the same or different channels in the same CPU, or to channels in two different CPUs. In addition, channel-switching features can be installed on one or both of the attachments (except for the Direct Disk Attachment).

For Model 2 of Models 115 and 125, the String Switch enables two strings of 3340 drives to be attached to any System/370 model (except a Model 115 Model 0 or Model 125 Model 0) and a Model 115 Model 2 or Model 125 Model 2.

The String Switch feature for 3340 disk storage is functionally similar in its operation to the Two-Channel Switch. A switch on the 3340 Model A2 can be set to allow the 3340 string to be accessed via both attachments, one at a time. In effect, this setting provides two control unit paths to the string. Switching is accomplished dynamically under program control. Alternatively, the switch can be set to dedicate the string to one attachment or the other so that the string can be accessed only via that attachment.

Figure 50.15.8 illustrates string switching for two 3340 strings attached to a 3830 Model 2 unit. In the configuration shown, both strings can be accessed via two channels and two control units. Channel switching, string switching, and 32 Drive Expansion features can be used to enhance the availability of 3340 direct access storage facilities and to extend backup capabilities when two System/370 systems (the same or different models) are present in an installation.

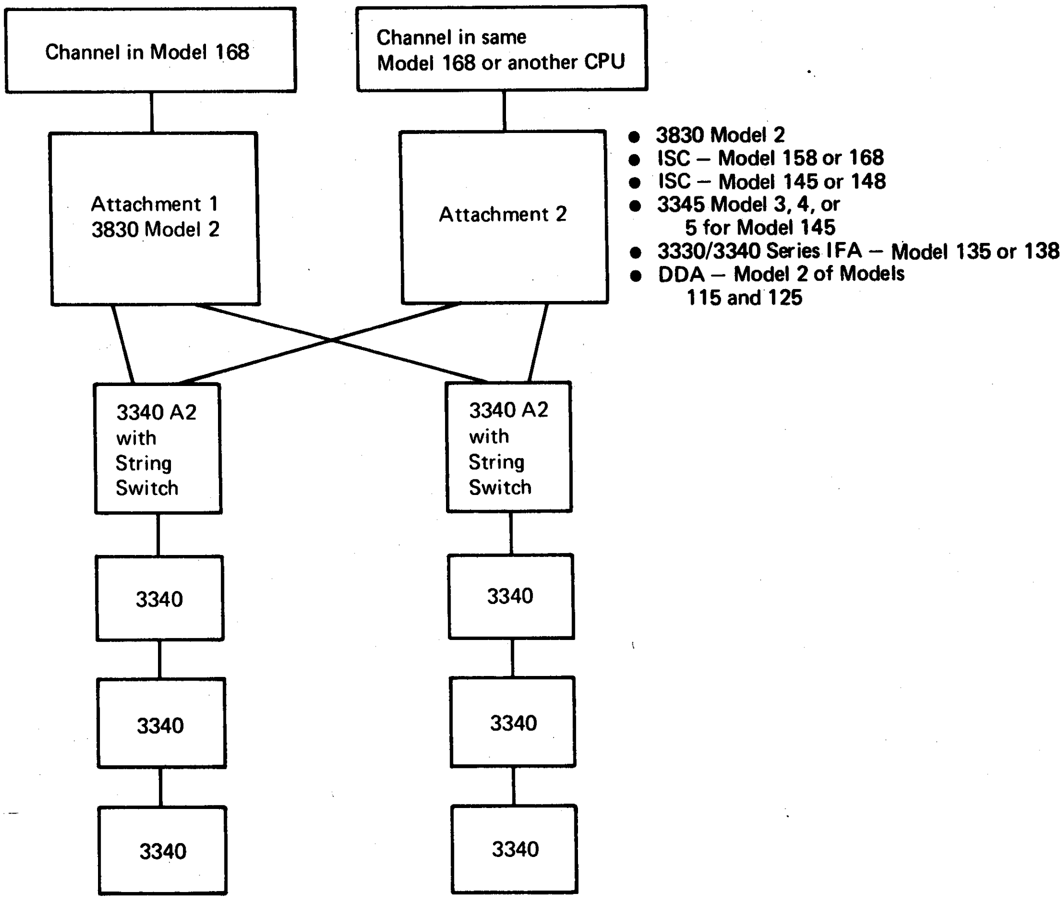

Channel in Model 168
Channel in same Model 168 or another CPU
Attachment 1 3830 Model 2
Attachment 2
3830 Model 2
ISC – Model 158 or 168
ISC – Model 145 or 148
3345 Model 3, 4, or 5 for Model 145
3330/3340 Series IFA – Model 135 or 138
DDA – Model 2 of Models 115 and 125
3340 A2 with String Switch
3340 A2 with String Switch
3340
3340
3340
3340
3340
3340

Figure 50.15.8. String switching for 3340 facilities attached to a 3830 Model 2

The 3830 Model 2 control unit is microprogram-controlled. Read/write monolithic storage contained in the control unit is used for microprogram residence. The 3830 Model 2 also contains a device that reads interchangeable disk cartridges. This device is used for microprogram backup storage and for storage of nonresident diagnostics for the 3340 string. During a 3830 Model 2 power-on sequence, the functional microprogram is loaded from the device into control storage within the 3830 Model 2 control unit. Therefore, microcode engineering changes can be installed merely by replacing the current disk cartridge with another that contains the new microprogram.

The 3830 Model 2 incorporates error detection, correction, and logging features that are designed to improve its availability and serviceability. For the 3340, the 3830 Model 2 provides the following facilities that are not implemented in System/360 direct access devices:

- I/O error routine correction of recoverable data errors on read operations with data supplied by the control unit in sense bytes. When the 3830 Model 2 detects a correctable data error during the reading of the data portion of a physical record, it generates the information necessary to correct the erroneous bytes. The sense bytes presented by the 3830 Model 2 contain a pattern of corrective bits and a displacement value to indicate which of the bytes transferred to processor storage contain the errors. The disk error recovery program need only EXCLUSIVE OR (logical operation) the corrective bit pattern with the error bytes in the input area in processor storage to correct the errors.

- Statistical usage recording by the 3830 Model 2. Statistical usage counters for each drive in a 3340 string are continuously maintained by the 3830 Model 2. These counters indicate the number of bytes read/searched, number of seeks issued, and number of command and data overruns for each device. When a counter reaches its threshold or a data module is removed from a drive, the 3830 Model 2 indicates the condition via a unit check when the next I/O operation is initiated to the drive or a data module is made ready on the drive. Counter data can be obtained and counters can be reset by issuing a READ AND RESET BUFFERED LOG command.

- Inline diagnostic testing of a malfunctioning drive. (Inline diagnostics are provided only for 2314 facilities.) A 3830 Model 2 control unit can execute diagnostic tests on a malfunctioning drive while normal operations take place on the remaining drives in the string. Diagnostic tests can be loaded into a transient area of the control storage of the 3830 Model 2 and executed on the malfunctioning drive. This can be done in an online environment using OLTEP or the CE panel on the 3830 Model 2. OLTSEP can be used in a standalone environment. This inline testing allows CE diagnosis and repair of most 3340 drive failures without the necessity of taking the entire 3340 string out of the system configuration.

A 3340 drive can be placed in CE mode (offline to the system) by means of a switch that is located inside the rear door of the drive so that maintenance functions can be performed. To take the 3340 drive out of CE mode and return it to online status, the attention pushbutton must be pressed. This also causes the access mechanism to move to physical track 0.

ATTACHMENT VIA INTEGRATED STORAGE CONTROLS

Optionally, one Integrated Storage Controls feature can be installed on a Model 168 to attach 3330/3344, 3330-series, or 3350 disk storage to one or two block multiplexer channels. Attachment of 3340/3344, 3330-series and 3350 disk storage via 3830 Storage Control is possible as well. The following discusses attachment of 3340 strings only.

The Integrated Storage Controls feature includes dual direct access storage controls, each of which operates independently of the other and is functionally like 3830 Storage Control Model 2 except for the following:

- The Integrated Storage Controls feature is contained in the main frame of the Model 168 and is powered by the Model 168 CPU.

- The Two-Channel Switch, Additional feature (that provides four-channel switching) cannot be attached to the logical storage controls in the ISC feature.

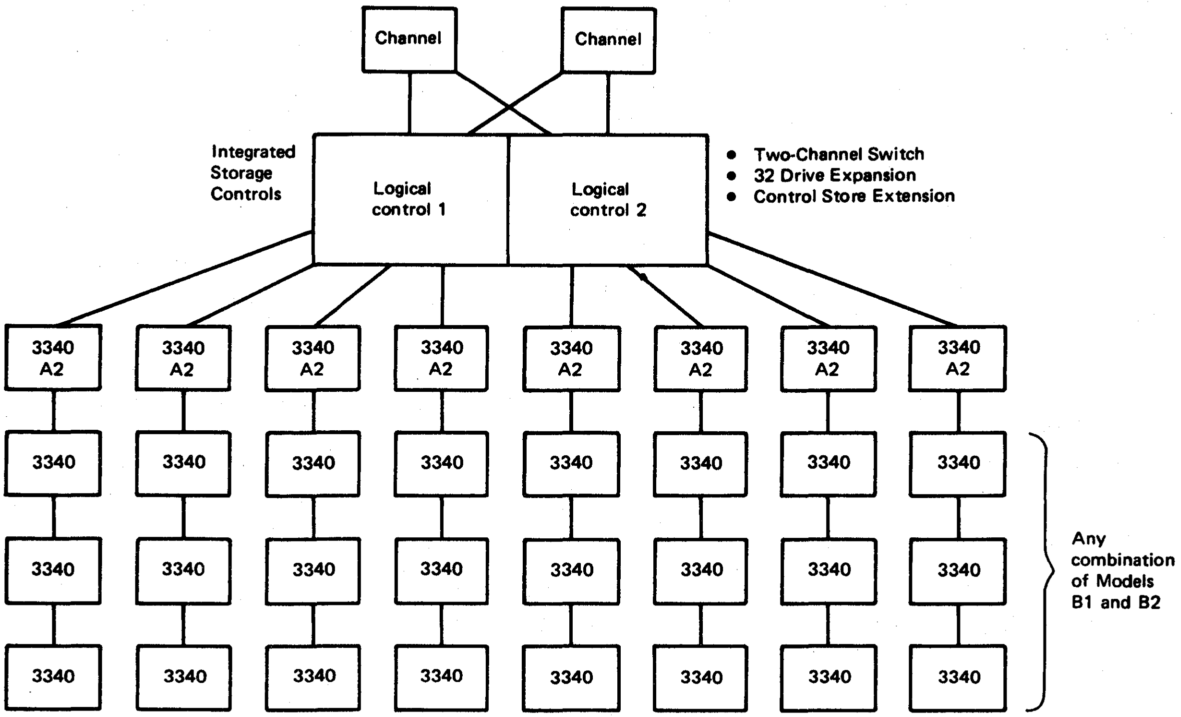
Both logical storage controls in the ISC feature can be attached to the same channel, two different channels in the Model 168 configuration, or a channel in the Model 168 configuration and a channel in another System/370. Each logical storage control can have attached a maximum of four 3340 strings of up to eight drives each. The 32 Drive Expansion and Control Store Extension optional features (field installable) must be installed in the ISC in order to attach more than two strings to each logical control. Therefore, up to 64 drives (eight strings) can be attached to the Model 168 via the ISC. The first unit in each 3340 string must be a 3340 Model A2. A maximum of two of the 3340 strings attached to the ISC can contain 3344 units.

The 3340 drives attached to the ISC operate just as if they were attached via 3830 Storage Control Model 2. That is, when multiple requesting is used, each logical storage control within the ISC can handle up to 32 channel programs concurrently, one on each of its drives, and only one of the 32 drives can be transferring data at a time. When a malfunction occurs, diagnostics can be run on one logical storage control and its drives while normal operations take place on the other logical storage control in the ISC.

Intermixing 3340 and 3330-series strings on the ISC is discussed below. Figure 50.15.9 summarizes the 3340 string configurations that are possible for a Model 168 ISC.

The ISC feature provides lower-cost attachment of 3340 disk storage than 3830 Storage Control Model 2 when two storage control units are required, and physical space is saved since the ISC is in the Model 168 CPU.

The Two-Channel Switch optional feature is also available for the ISC. When installed, this feature provides a two-channel switching capability for both of the logical storage controls. The Two-Channel Switch permits each logical storage control to be attached to two channels in the same Model 168 configuration or to one channel in the Model 168 configuration and one channel in another System/370. Two switches are provided that can be set to dedicate a logical storage control to one channel or the other, or to enable the storage control to be accessed by both channels.


Channel
Channel
Integrated Storage Controls
Logical control 1
Logical control 2
3340 A2
3340
Any combination of Models B1 and B2

- Two-Channel Switch
- 32 Drive Expansion
- Control Store Extension

- One to four strings of from two to eight drives each connected to each logical control. Each logical control connected to one or two channels in the same or different CPU's.
- String Switch can be added to any or all 3340 Model A2 units to connect a 3340 A2 to a second attachment in the same ISC, the same CPU, or another CPU except Model 0 of Models 115 and 125.

Figure 50.15.9. Permissible 3340 string configurations for the Model 168 Integrated Storage Controls feature

The String Switch optional feature can be installed on a 3340 Model A2 that is attached to the ISC. This field-installable feature enables the 3340 Model A2 and all its attached 3340s (a 3340 string) to be connected to two control-unit-type attachments instead of only one, as discussed for the 3830 Model 2.

Figure 50.15.10 illustrates string switching for four 3340 strings that are attached to the same ISC. In the configuration shown, all strings can be accessed via two channels and two control units. Channel switching, string switching, and 32 Drive Expansion features can be used to enhance the availability of 3340 disk storage and to extend backup capabilities when two System/370 systems (the same or different models) are present in an installation.

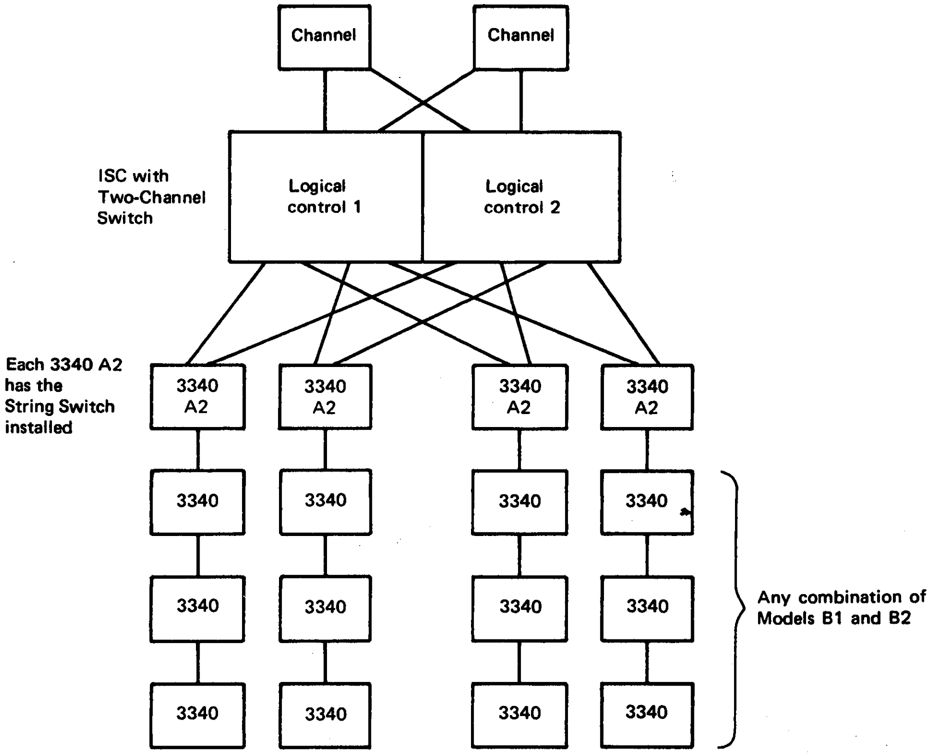


Figure 50.15.10. String switching for 3340 facilities attached to one ISC

INTERMIXING 3340 AND 3330-SERIES STRINGS ON AN ATTACHMENT

Optionally, the 3333/3340 Intermix feature can be installed on 3830 Storage Control Model 2 and Integrated Storage Controls in the Model 168 CPU. When present, this field-installable feature permits both 3340 and 3330-series strings to be attached to a 3830 Model 2 or ISC. Each string must contain all 3340 drives (no 3344 units) or all 3330-series drives. A 3340 string that contains 3344 units cannot be intermixed with 3330-series strings attached to the ISC.

The intermix feature requires installation of the Control Store Extension feature on the 3830 Model 2 or ISC and can coexist with other optional features for these units and their strings (channel switching, 32 Drive Expansion, string switching, and fixed head features).

SUMMARY

The hardware features of the 3340 and 2314 direct access storage facilities are summarized in Table 50.15.3. Table 50.15.4 compares the capabilities of the 3830 Model 1, 3830 Model 2, and Model 168 Integrated Storage Controls for both 3340 and 3330-series disk storage.

When compared with the 2314 facility, the 3340 facility offers the following major advantages:

- Faster access to data
  Data transfer rate almost three times that of the 2314
  Seek times approximately 40% of those of the 2314 for movable head accesses
  Zero seek time provided by the fixed heads in a 3348 Model 70F Data Module
  Rotational delay interval approximately 20% shorter than for the 2314

- Larger capacity per drive
  17% for the Model 35 data module
  175% for Model 70 and 70F data modules

- Two capacity options per drive for expanded growth flexibility

- Multiple requesting and rotational position sensing capabilities for use with block multiplexer channels

- Operational improvements
  Cover tightening/untightening and removal/replacement operations are eliminated, speeding up data module loading and unloading
  Load time to ready status for a mounted data module is three times faster
  Write protection is provided on a data module basis
  External labeling procedures are more flexible and leave less chance of erroneous data module labeling

- Significantly increased reliability
  Sealed cartridge design eliminates head-to-disk alignment problems, minimizes the possibility of disk surface contamination, and eliminates hub wear and damage
  Advanced head design makes head crashes a remote possibility and permits increased recording density without any loss of reliability

- Improved error handling capabilities
  Error correction data is provided by the hardware for use by programmed error recovery procedures
  Surface defect skipping reduces the need to use the error correction capability

- Improved availability and serviceability
  No preventive maintenance is scheduled, because of the reliability features of the 3340 and 3348
  Faster error isolation and correction is possible because the 3340 contains fewer circuit cards
  Expanded microdiagnostics can test more than 95% of the circuits in a 3340

Table 50.15.3. Summary of the hardware features of 3340 and 2314 disk storage facilities

Feature	3340 attached to 3830 Model 2 or ISC	2314 (A-Series)
Number of drives per string or facility	Two to eight in one drive increments	One to eight in one-drive increments. (A ninth can be included as a spare only.)
Number of strings or facilities per control unit	One to four (maximum of eight strings for ISC)	One maximum
Data medium used	Removable interchangeable data module (sealed cartridge)	Removable interchangeable disk pack
Read-only feature on drive or data medium	Yes, on data module	No
Removable address plugs on drive	No	Yes
Attachment of a string or facility to two control units in the same or a different CPU	Yes, via optional string switch feature. Only one data transfer operation permitted per string.	Yes, via 2844 Auxiliary Storage Control. Two concurrent data transfer operations per facility permitted.
Two-Channel Switch	Optional	Optional
Attachment of the control unit to four channels	Yes using the optional Two-Channel Switch and Two-Channel Switch, Additional features (3830 Model 2 only)	Yes using the optional Two-Channel Switch and 2844 Auxiliary Storage Control
Record Overflow	Standard	Standard
File Scan	Not available	Standard
Multiple track operations	Standard	Standard
Multiple requesting	Standard	Not available
Rotational Position Sensing	Optional (on 3340 drives)	Not available
Error correction data presented by control unit	Yes	No
Surface defect skipping	Yes	No

Table 50.15.3 (continued)

Feature	3340 attached to 3830 Model 2 or ISC	2314 (A-Series)
Writable storage in control unit loaded from a disk cartridge	Yes	No
Statistics logging by the control unit in its storage	Yes	No
Inline diagnostics executed under OLTEP or via the CE panel	Yes	Yes

Table 50.15.4. Summary of the features of 3830 Storage Control Models 1 and 2 and Integrated Storage Controls

Characteristic	3830 Model 1	3830 Model 2	ISC
Type of unit	Standalone	Standalone	Contained in Model 168 CPU
Power source	Contains own for itself and all the drives that can be attached to it	Contains own for itself only	Power control shared with Model 168 CPU
Attaches to	Block multiplexer channel	Block multiplexer channel	Block multiplexer channel
Devices attaching to it	3330 Models 1 and 2	3333 Models 1 and 11 (optionally with 3330 Model 1, 2, and 11 units attached) 3340 Model A2 (optionally with 3340 Model B1 and B2 units attached)	Same as 3830 Model 2
Number of drives in a string	1 to 8	2 to 8 for a 3330-series or 3340 string	Same as 3830 Model 2
Standard number of strings attachable	One maximum	Two maximum	Two maximum per logical control
32 Drive Expansion feature for attachment of two additional strings	Not available	Optional for a maximum of four strings	Optional for a maximum of four strings per logical control

Table 15.15.4 (continued)

Characteristic	3830 Model 1	3830 Model 2	ISC
3333/3340 Intermix feature for attachment of 3330-series and 3340 strings	Not available	Optional	Optional
Two-Channel Switch	Optional	Optional	Optional
Two-Channel Switch, Additional (for four-channel switching)	Optional	Optional	Not available
String switching capability	Not available	Yes, for 3330-series strings via optional 3333 String Switch feature. Yes, for 3340 strings via optional String Switch Feature.	Same as 3830 Model 2
Multiple requesting	Standard	Standard	Standard
Rotational position sensing	Standard	Standard on control unit (standard on 3330-series drives, optional on 3340 drives)	Same as 3830 Model 2
Multiple track operations	Standard	Standard	Standard
Record overflow	Standard	Standard	Standard
Command retry	Standard	Standard for 3330-series strings. Not available for 3340 strings.	Same as 3830 Model 2
Surface defect skipping	Not implemented	Implemented for 3340 strings. Not implemented for 3330-series strings.	Same as 3830 Model 2
Inline diagnostic tests	Standard	Standard	Standard
Error logging by control unit	Standard	Standard	Standard

SECTION 60: MULTIPROCESSING

60:05 GENERAL DESCRIPTION

INTRODUCTION

The System/370 Model 168 offers large-system users the advantages of shared storage multiprocessing, an advanced system function not available for System/370 Models 165 and 165 II. A Model 168 multiprocessing configuration, which is similar in architectural design to a System/360 Model 65 shared storage multiprocessing configuration, contains two multiprocessor models of the Model 168 that are connected via the 3068 Multisystem Communication Unit.

The two systems in a Model 168 multiprocessing configuration share their processor storage and operate under the control of a single operating system that is resident in the shared processor storage. One input queue and one task queue can be maintained for the configuration and both CPUs can be used to process each task (but not simultaneously).

IBM-supplied programming systems support of shared storage multiprocessing is provided in OS/VS2 Releases 2 and up, that is, in OS/VS2 Multiple Virtual Storage (MVS) support and by Time Sharing System/370 (TSS/370). Model 168 multiprocessing is compatible with Model 158 multiprocessing.

While Model 168 shared storage multiprocessing is very similar in design to Model 65 shared storage multiprocessing, Model 168 multiprocessing offers all the advantages provided by the new architectural features and I/O devices of System/370, and incorporates functional hardware and programming enhancements that reflect the knowledge gained from actual installation of Model 65 multiprocessing. Thus, Model 168 multiprocessing offers many advantages in addition to those provided by Model 65 multiprocessing and provides improved performance.

The price performance of Model 168 multiprocessing makes shared storage multiprocessing available to large-system users who desire the advantages provided by this type of system configuration but who previously could not justify the cost. Because Model 168 multiprocessing configurations are upward compatible with uniprocessor System/360 and System/370 models, nondisruptive growth is provided for installations with multiple systems (Models 65, 155, 158, and 168, for example) that require a shared storage multiprocessing environment.

The Model 168 can be part of a wide variety of multiprocessing configurations. A Model 168 shared storage multiprocessing configuration can be combined with other types of multiprocessing configurations. The breadth and flexibility of the multiprocessing configurations supported for System/370 enable an installation to combine multiple systems such that the particular advantages offered by different types of multiprocessing configurations can be obtained as desired and ease of growth is assured.

DEFINITION OF MULTIPROCESSING

A multiprocessing system is one in which two or more CPUs are interconnected and execute two or more tasks simultaneously, one in each CPU. Multiprocessing is a logical extension of multiprogramming, in which two or more tasks operate concurrently in a single CPU. In a multiprogramming environment, one task executes at a time and only I/O operations for two or more tasks can operate simultaneously. In a multiprocessing environment, both I/O operations and instruction execution for two or more tasks in the same or different programs can occur simultaneously, with each task executing in a different CPU.

The hardware connection of the CPUs in a multiprocessing configuration is the means by which CPUs communicate with each other in order to coordinate the activity of the multiprocessing system. A multiprocessing configuration can be tightly or loosely coupled or can include a combination of both loosely and tightly coupled processors.

A tightly coupled multiprocessing configuration is one in which (1) the processors share access to all the processor storage available in each system, (2) CPU-to-CPU communication is accomplished via the storing of data in shared storage and via direct CPU-to-CPU signals (both program- and hardware-initiated), and (3) a single control program is used. Model 168 and 158 shared storage multiprocessor systems are, therefore, tightly coupled multiprocessing configurations. A Model 168 Attached Processor System (discussed in Section 67) is also a tightly coupled multiprocessing configuration.

A loosely coupled multiprocessing configuration is one in which (1) CPUs are coupled via shared access to direct access storage or via channel-to-channel connections, (2) each CPU has its own control program, and (3) a single system scheduling and operational interface is optional.

Loosely coupled multiprocessing configurations for System/370 models are supported by ASP (Asymmetric Multiprocessing System) Version 3, JES2 Multi-Access Spool support in OS/VS2 MVS, and JES3 support in OS/VS2 MVS. ASP Version 3 supports from 2 to 33 systems connected via channel-to-channel adapters. One support system schedules the operation of up to 32 main systems.

The JES2 Multi-Access Spool facility supports from two to seven systems, each with its own control program resident, that share input and output work queues on direct access storage shared by the systems. JES3, a generally compatible extension of ASP, supports from two to eight systems connected via channel-to-channel adapters. Each of the systems in a JES3 configuration can be a uniprocessor system or tightly coupled multiprocessing system (that is, Model 158 multiprocessing system, Model 168 multiprocessing system, or Model 168 Attached Processor System).

The objective of coupling multiple systems to form a multiprocessing configuration is to obtain a configuration that combines advantages of a single processor environment with those of an uncoupled multiple processor environment.

A single processor environment offers the following advantages:

- Best price performance potential, since one large system is generally more economical than several small systems

- A single interface to the computing system for workload scheduling and operation of the system

- The ability to apply all the resources of the system to a given job step when necessary'

The advantages provided by an uncoupled multiple processor configuration are:

- The capability of adding to the configuration in smaller increments, that is, the addition of a smaller model rather than replacement of the existing model with the next larger model when additional computing power is required. The next larger model may provide additional computing power far in excess of that required.

- More economical growth possibilities for installations with purchased systems

- Growth possibilities for large-scale installations that have the largest model of the system already installed

- Enhancements to configuration availability (better probability that a system will be available for critical application processing), serviceability (maintenance can occur simultaneously with production processing), and reliability (protection of critical jobs from failures in noncritical jobs by processing them in separate systems)

THE MODEL 168 MULTIPROCESSING SYSTEM

A Model 168 multiprocessing system, as shown in Figure 60.05.1, consists of two multiprocessor CPU models of the Model 168 (3168 Processing Units) that are interconnected via the 3068 Multisystem Communication Unit plus the following for each Model 168 CPU: one standalone 3066 System Console, one standalone 3067 Power and Coolant Distribution Unit, standalone channels, and a motor generator set. The physical dimensions of the 3168, 3066, and 3067 units used in multiprocessor and uniprocessor configurations are the same. A field-installable multiprocessing feature must be installed on 3066 and 3067 units that are part of a Model 168 multiprocessing system. One CPU is designated as CPU A while the other is CPU B.

Asymmetric processor storage multiprocessing configurations (the connection of two multiprocessor systems with different amounts of processor storage) are permitted for the Model 168. The physical dimensions of the multiprocessor models of the Model 168 CPU are identical to those of the uniprocessor models.

When multiprocessor mode is in effect and OS/VS2 MVS multiprocessing support is used, the two CPUs in a Model 168 multiprocessing configuration share from 2 to 16 megabytes of processor storage, I/O devices that have channel- or string-switching features installed on their control units, a single control program, and a single set of work (input and output) queues. The standalone channels for each system are normally dedicated to that system.

The two systems in a Model 168 multiprocessing configuration can be reconfigured via the configuration control panel (located on the 3068 Multisystem Communication Unit) to permit each system to operate independently from the other, that is, in uniprocessor mode.

CPU Features

Multiprocessing is an optional feature for the Model 168 and is field-installable. That is, uniprocessor models of the 3168 CPU can be field-converted to multiprocessor models and attached to the 3068 unit. The multiprocessor models of the Model 168 have the same standard and optional features as the uniprocessor models. All standard and optional features can operate when multiprocessor mode is in effect. However, IBM-supplied programming systems support for the Model 168 supports shared storage multiprocessing operations only for EC and DAT modes. BC mode multiprocessing operations are not supported.

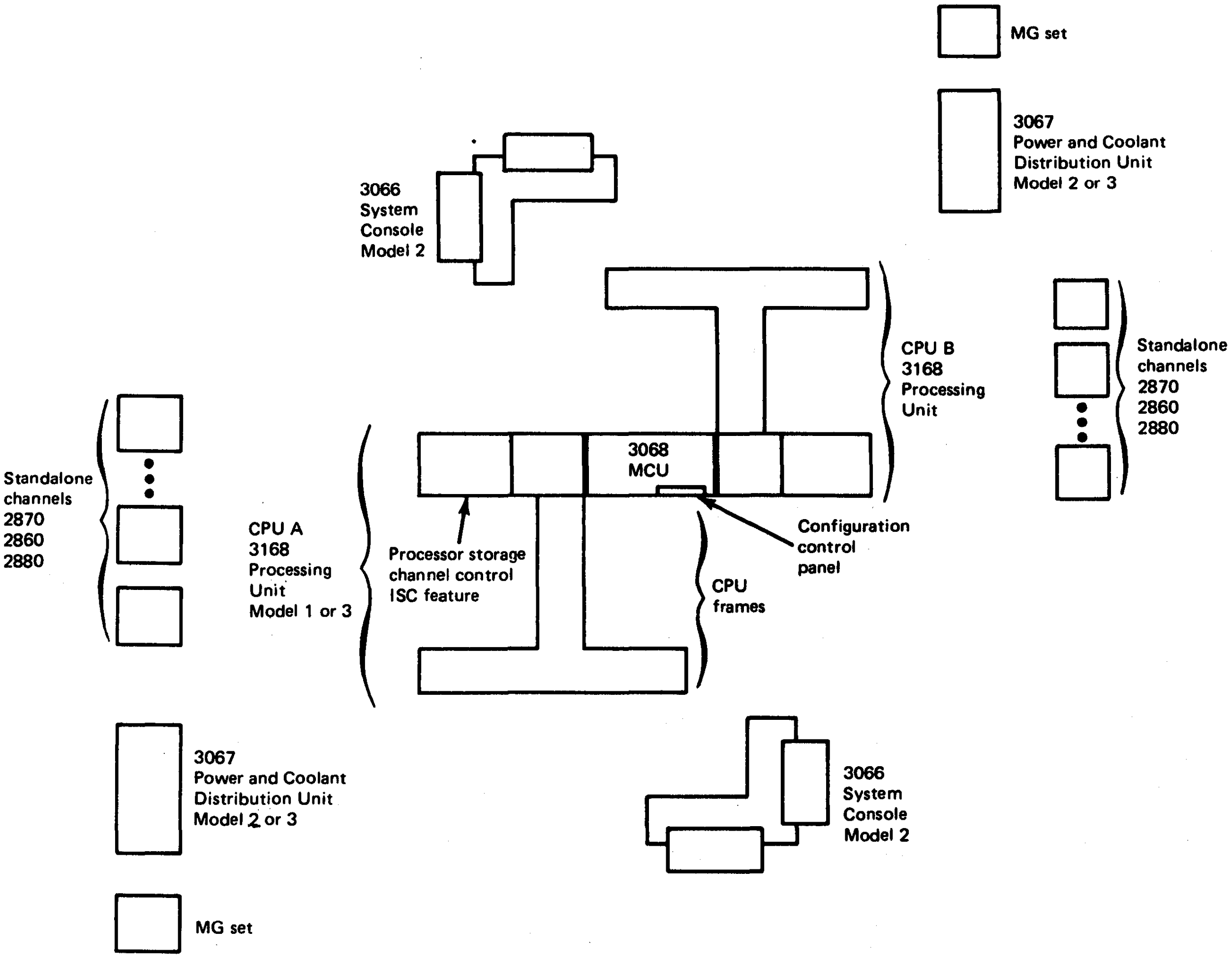


Figure 60.05.1. Model 168 Model 1 multiprocessing system

OS/VS2 multiprocessing support for the Model 168 does not require symmetric CPU features. That is, the two CPUs need not have the same optional features installed. When CPU features are not symmetric, the CPU affinity facility of OS/VS2 multiprocessing support can be used to ensure that a program requiring a feature present in only one CPU is processed only by that CPU. When CPU features are symmetric, both CPUs can be used in the processing of each program.

The AFFINITY macro can be included during OS/VS2 MVS system generation to specify the names of those programs that can be processed by only one CPU and the address of that CPU. This macro must be used for Model 168 multiprocessing systems when a 7000-series emulator feature is asymmetric. The AFFINITY macro need not be specified when the High-Speed Multiply feature is installed in only one of the two Model 168 multiprocessor systems, since the system without the feature still has the capability of processing multiply instructions. Nor does installation of the Buffer Expansion feature on only one system require specification of the AFFINITY macro.

Processor Storage

The processor storage used in Model 168 multiprocessor models is the same four-way, doubleword interleaved monolithic storage that is used in the uniprocessor models. The eight multiprocessor storage sizes for the Model 168 Model 1 are:

Model	Size in Bytes (K=1024)
MP1	1024K
MP2	2048K
MP3	3072K
MP4	4096K
MP5	5120K
MP6	6144K
MP7	7168K
MP8	8192K

The total amount of shared storage in a Model 168 multiprocessing configuration can vary from a minimum of 2048K bytes (two MP1 models) to a maximum of 16,384K bytes (two MP8 models) in 1024K-byte increments.

The total processor storage contained in each multiprocessor model of the Model 168 is logically divided into 1024K-byte units, called storage elements, in order to implement floating storage addressing. Floating storage addressing is implemented in multiprocessor models of the Model 168 so that the physical address range assigned to each element of processor storage can be varied as needed.

The address range of each 1024K-byte element of processor storage is specified using the floating storage address switches on the configuration control panel when either uniprocessor or multiprocessor mode is in effect (see discussion in Section 60:10 under "3068 Multisystem Communication Unit").

Since processor storage is four-way interleaved in multiprocessor models (as in uniprocessor models), the total processor storage available in the multiprocessing configuration is divided into four logical storages, each of which can be accessed simultaneously. Nonsimultaneous requests from the processors (the two 3168 Processing Units) to a nonbusy logical storage are handled on a first-come, first-served basis. When both processors simultaneously request access to the same logical storage, a priority scheme is used to determine which processor is given access.

First, the highest priority contender within each processor is established. This determination is made concurrently in the two processors. Within each processor, priority is resolved among requests from channel buffers and among requests from three CPU registers. These two priorities are resolved within a processor at the same time. The highest priority channel buffer is then given priority over the highest priority CPU register within a processor.

After the highest priority request in each processor is determined, priority is resolved between the two requests according to the rule that each processor has the capability of selecting a given logical storage on an alternate (every other) storage (80-nanosecond) cycle basis. That is, one processor can access a given logical storage every odd storage cycle while the other can access the same logical storage every even storage cycle. Note, however, that neither processor is permitted to have two successive accesses to a given logical storage if the other processor also has a request outstanding for that logical storage.

Thus, if one processor, say CPU A, is given storage access and accesses four consecutive logical storages beginning with, for example, logical storage 1 and has another request outstanding for logical storage 1, CPU A is not given access to logical storage 1 if the highest priority request in CPU B is also for logical storage 1 (even though the current cycle is still the alternate one during which CPU A can select logical storage 1). CPU B is given access to logical storage 1 on its alternate cycle (that is, one storage cycle later).

When the multiprocessing feature is installed, requests from a processor to its own physical processor storage (local requests) and to processor storage contained in the other processor (remote requests) are channeled to storage via the 3068 unit. Therefore, the same amount of time is required for a remote request as for a local request when multiprocessor mode is in effect and more time is required to access processor storage in a multiprocessor model of the Model 168 than in a uniprocessor model.

Table 60.05.1 lists the processor storage access and fetch times for Model 168 multiprocessor models when multiprocessor mode is in effect and uniprocessor mode is in effect with or without processor storage cross-configured. Processor storage is said to be cross-configured when uniprocessor mode is in effect for the two systems and one or more elements of processor storage physically contained in one system can be accessed only by the other system.

Table 60.05.1. Processor storage access and fetch times for multiprocessor models in nanoseconds

	Multiprocessor Mode Shared Storage	Uniprocessor Mode Cross- Configured	 Not Cross- Configured
Processor storage access time (from time of PSCF select to availability of data in the PSCF)	480	480	480
Processor fetch of eight bytes from processor storage (from time of request acceptance to availability of data in a processor register) assuming the logical storage is not busy	800-880*	800-880*	720

*The smaller time applies if the fetching processor has priority for selecting processor storage; otherwise, the larger time applies. In addition, the times indicated will increase if the logical storage is busy.

Channels

The maximum number of channels permitted per Model 168 CPU, twelve, can be installed for each Model 168 in a tightly coupled multiprocessing configuration for a total of 24 channels. Ideally, each CPU should have the same number of 2870 Multiplexer, 2860 Selector, and 2880 Block Multiplexer channels so that at least two channel paths to I/O devices, one from each Model 168 CPU, can be made available via installation of programmable channel- or string-switching features. Asymmetric channel configurations are supported so that a malfunctioning channel can be removed from the operational system configuration during processing.

Normally the standalone channels in a Model 168 multiprocessing system are dedicated to that system and cannot be accessed by the other Model 168 CPU in the multiprocessing configuration. However, channel reconfiguration hardware is included in the Model 168 multiprocessing feature. It is designed only to aid the recovery provided by the OS/VS2 multiprocessing control program after one CPU fails and must be logically removed from the operational multiprocessing configuration. When activated in a CPU, channel reconfiguration hardware enables that CPU to control the operation of the dedicated channels for the other CPU. Channel reconfiguration hardware is discussed in detail in Section 60:10.

I/O Devices

Any I/O device that can be attached to a Model 168 uniprocessor system can be included in a Model 168 multiprocessing configuration. While OS/VS2 multiprocessing support does not require symmetry for all I/O devices, for maximum availability the I/O device configuration should be as symmetric as possible. Ideally, the same I/O device configuration should be attached to each CPU and, where it is available, a programmable channel-switching or direct access device string-switching feature should be installed on each control unit to provide each CPU with access to the device.

A maximum of four channel paths per system to a given control unit is supported by OS/VS2 multiprocessing support. (Note that only two channel paths to a device are supported for uniprocessor systems in OS/VS2 MVS.) A control unit can be attached to only one channel in only one system, two channels in only one system, one channel in each system, or two channels in each system. The latter two configurations are preferable for multiprocessing systems. I/O devices connected to control units with a channel- or string-switching feature must have the same I/O address (channel, control unit, and device) in each system.

Note that the total number of I/O devices present in a Model 168 multiprocessing system cannot be greater than the maximum number of I/O devices present in a uniprocessor configuration because OS/VS2 MVS supports the same maximum number of UCBs (unit control blocks) for I/O devices in its multiprocessor support as in its uniprocessor support.

String-switching features are available for 3330-series, 3340/3344, and 3350 disk storage and for the 3330 strings in a 3850 Mass Storage System. The following are some of the more frequently used I/O devices for the Model 168 that have a programmable two-channel switch available for their control unit:

- 1403 Printer
- 2540 Card Read Punch
- 2400-series Magnetic Tape
- 3400-series Magnetic Tape
- 2314/2319 Disk Storage
- 2305 Model 1 and 2 Disk Storage
- 3330-series Disk Storage
- 3340 and 3344 Disk Storage

- 3350 Disk Storage

- 3800 Printing Subsystem

- 3850 Mass Storage System

The Channel Adapter Type 3 feature can be installed on a 3705 Communications Controller to permit it to be switched between the two systems in a multiprocessing configuration,under program control. This channel adapter is functionally equivalent to the programmable two-channel switch available for certain direct access devices (except for RESERVE/RELEASE functions). The 3704 Communications Controller has a nonprogrammable two-channel switch that permits it to be manually switched between two systems.

For other I/O devices, the 2911 Manual Switch or 2914 Switching Unit can be installed to provide a manual switching capability for their control units when the device is not present in both systems. Manual switching using the 2911 is normally done by the operator when the systems are not operating, such as before an IPL. This is also true for a 2914 unless additional RPQs are installed on the 2914 that enable switching to be done dynamically during system operation (switching becomes effective the next time the channel interface becomes inactive).

Shown below is a sample I/O configuration that operates at the near maximum aggregate data rate achievable (with negligible or no overrun) for a Model 168 tightly coupled multiprocessing configuration that includes I/O devices with a 3-megabyte data rate. The I/O configuration listed below is assumed to operate with multiprocessing mode and four-way interleaving active. Assuming the identical I/O configuration shown below in both systems, the aggregate data rate of a Model 168 multiprocessing configuration with 3-megabyte devices is 27.8 megabytes per second.

Channel Priority	Channel Type	Device Type	Data Rate (MB/sec)
1	2880	2305 Model 1 disk or 3420 Model 8 tape	3.0
2	2880	2305 Model 1 disk or 3420 Model 8 tape	3.0
3	2870	Miscellaneous	.56
4	2880	2305 Model 1 disk or 3420 Model 8 tape	3.0
5	2870	Miscellaneous	.66
6	2880	3330 disk or 3420 Model 6 tape	.8
7	2860	3420 Model 7 tape	.32
9	2880	3330 disk	.8
A	2880	3330 disk	.8
B	2860	3420 Model 7 tape	.32
C	2860	3420 Model 7 tape	.32
D	2860	3420 Model 7 tape	.32
	Total aggregate data rate per CPU		13.9

Note that a 2860 selector channel with 2301 drums or 3420 Model 8 tape units attached must have channel priority 1 assigned. For best performance, a 2880 channel with 2301 drums, 2305 Model 1 or 2 disk storage, or 3420 Model 8 tape units attached should be assigned priority 1, 2, or 4 (although 2880 channels with 2305 Model 2 or 3420 Model 8 tapes can be assigned other priorities as shown in Table 60.05.2). A 2870 byte multiplexer channel with four selector subchannels must be assigned priority 1 or 2.

Table 60.05.2 indicates the channel priorities the higher speed System/370 devices require in a Model 168 multiprocessing configuration. That is, each I/O device in the table can be attached only to a channel with one of the priorities indicated in its column. Each column also indicates the maximum number of channels to which the device can be attached without violating data rate constraints (three for the 2305 Model 1, five for the 2305 Model 2, etc.).

Permissible I/O device configurations are also shown in Table 60.05.2, which in turn indicates the I/O device configurations that can operate concurrently. In general, any other device type with similar characteristics and the same or a slower data rate than the listed device can also be assigned a channel with the indicated priority (3350 with an 1198-KB/sec data rate same as the 3420 Model 8, 3340/3344 with an 885-KB/sec data rate same as a 3330).

Table 60.05.2. Permissible configurations and channel priorities for highest speed System/370 I/O devices in a Model 168 tightly coupled multiprocessing configuration. (All 3420 tapes are assumed to be attached to 3803 Model 2 Control Units.)

Channel Priority	Device Type							
	2305 Model 1 3 MB/sec	2305 Model 2 1.5 MB/ sec	3420 Model 8 1.25 MB/sec		3330-series .8 MB/ sec	3420 Model 6 .8 MB/ sec	3420 Model 4 .47 MB/ sec	3420 Model 7 .32 MB/ sec
			2880	2860		2860	2860	2860
1	X	X	X	X	X	X	X	X
2	X	X	X	X	X	X	X	X
3		X	X	X	X	X	X	X
4	X	X	X	X	X	X	X	X
5		X	X	X	X	X	X	X
6			X	X	X	X	X	X
7			X	X	X	X	X	X
9			X		X	X	X	X
A						X	X	X
B							X	X
C						X	X	X
D							X	X

3068 Multisystem Communication Unit

The 3068 Multisystem Communication Unit contains most of the Model 168 multiprocessing feature hardware. The balance of the multiprocessing feature is contained in the two CPUs. The 3068 is divided into two physical units. One unit is associated with CPU A and the other with CPU B. A physical half of the 3068 unit receives its power and cooling from the 3067 for its associated CPU. The 3068 is also water-cooled.

The 3068 Multisystem Communication Unit also contains the configuration control panel. This panel provides the operator with the capability of establishing the hardware configuration (storage and I/O devices) that is to be online for each system, assigning address ranges to storage segments, establishing storage interleave mode, establishing system and time-of-day clock oscillator control, and configuring the two systems for uniprocessor mode or multiprocessor mode operations.

3066 System Console

During multiprocessing operations, the 3066 console on which IPL was performed is normally used as the operating system primary console. The other 3066 console can be used as an alternate or, when DIDOCS is used, additional console.

The two 3066 consoles should not be assigned the same I/O address since the OS/VS2 multiprocessing control program considers the failure of one console to be the failure of both consoles when both consoles have the same address.

The system control panel on a 3066 console used in a Model 168 multiprocessing configuration has a multisystem activity meter and associated rotary switch in addition to all the same pushbuttons, lights, and switches as a 3066 console for a uniprocessor model of a Model 168. These are the only items added for multiprocessing.

The multisystem activity meter logically combines the system activity meters on the two 3066 system control panels to display the average activity of the major elements of the multiprocessing configuration. The rotary switch associated with the multisystem activity meter determines the activity displayed. When it is set to the A-B OVLP position, the logical ANDing of the activity selected in the activity meter switches for both systems is displayed. When the switch is set to the A/B CALIBRATE position, the logical ORing of the activity selected for both systems is displayed.

The 3066 console for Model 168 uniprocessor models has the same functions and operates in the same way as the 3066 console for uniprocessor models except for additional communication functions required for multiprocessing operations. Specifically, when a system reset (with or without the system clear pushbutton pressed) or load (IPL with or without the system clear pushbutton pressed) is performed by pressing the system reset or load key on one multiprocessor system, a corresponding function is also automatically sent (broadcast) to the other system when multiprocessing mode is in effect, as shown below. The setting of the system clear pushbutton is propagated as well.

Function Selected by Operator on Local CPU	Function Performed on Local CPU	Function Broadcast to Remote CPU
System reset (normal)	Program reset	Program reset
System reset (clear)	System clear	Initial program reset
Load (normal)	Program reset	Program reset
Load (clear)	System clear	Initial program reset

The only other communication function provided in the console unit for multiprocessor mode operations is for the time-of-day clock. When the time-of-day clock security switch for either system is held in the enable set position, this setting is propagated to the other CPU so that the time-of-day clock in each system is enabled for setting.

ADVANTAGES OF MODEL 168 TIGHTLY COUPLED MULTIPROCESSING CONFIGURATIONS

The major advantages of a Model 168 tightly coupled multiprocessing configuration when compared with two uncoupled systems, each having half the resources of the total multiprocessing configuration, are the following:

- Improved availability
- Less complex operational requirements
- Improved resource utilization
- Operational flexibility
- Improved growth options
- Improved throughput possibilities

These advantages are made possible by hardware resource redundancy, extensive hardware reconfigurability (implemented both in hardware and the OS/VS2 MVS multiprocessing control program), tightly coupled hardware interconnection that permits the configuration to operate with one control program and one work queue, and the availability features that are basic to the design of the OS/VS2 MVS control program. In a tightly coupled multiprocessing environment, the most critical component is the control program, and CPUs are viewed as system resources to be allocated to tasks just as are I/O devices, processor storage, and programs.

Availability

Availability as it relates to a data processing installation is usually described as the percentage of scheduled time the system or an application is capable of processing. A system is available when both its hardware and programming system are capable of processing jobs. An application is available when it is capable of performing processing for its end users.

Unavailable time occurs for a system when, for example, a hardware or control program failure occurs and system recovery procedures are invoked, an operator error causes a failure and recovery is required, scheduled preventive maintenance is performed, engineering changes are applied to hardware, fixes are applied to programming systems, and diagnostics are performed to locate a hardware malfunction that prevents continued system operation.

The improved availability offered by a tightly coupled Model 168 multiprocessing configuration is the result of hardware component redundancy, hardware component reconfigurability, and availability features implemented in OS/VS2 MVS uniprocessor and tightly coupled multiprocessor support. Since there are two CPUs and, in general, CPU feature, channel, and I/O device symmetry as well as access to I/O devices by both CPUs, backup is usually available when a hardware component fails.

In addition, the reconfiguration capabilities supported by the OS/VS2 MVS multiprocessing control program enable any type of hardware resource (a CPU, certain storage element, channel, or I/O device) to be logically removed from the operational multiprocessing system on an individual basis (within certain limitations) without the necessity of terminating system operations and performing a re-IPL. The rest of the multiprocessing system can then continue to function with a minimum of performance degradation as only the malfunctioning component is removed.

In a tightly coupled multiprocessing environment, there is more likelihood of having a critical subset of the total system resources available for processing than in an environment with two uncoupled systems, since all processor storage can be accessed by both CPUs.

Availability is also enhanced by the capability of physically removing malfunctioning components from the configuration for deferred or concurrent maintenance without impacting the availability of the rest of the multiprocessing configuration. Once the operator issues a VARY OFFLINE command to indicate the component to be removed, the OS/VS2 multiprocessing control program issues a message as soon as it has finished using the component and has logically removed it from the operational system. Using the configuration control panel, the operator can then physically remove the component from the configuration while normal processing continues, provided the proper operational procedure is followed.

In a configuration with two uncoupled systems, only unallocated I/O devices and redundant channel paths can be logically removed from the operational system (varied offline) without stopping the system. When a CPU or a critical portion of processor storage is the malfunctioning component, all the hardware resources of the system are unavailable (except switched I/O devices) until repair has been completed.

While a configuration with two uncoupled systems does provide backup by offering component redundancy, a tightly coupled multiprocessing configuration provides other availability advantages. First, since there is a single work queue, instead of two independent job queues, in a tightly coupled multiprocessing configuration, recovery from a CPU failure is faster because the switchover from one system to the other of jobs in progress and queued jobs is eliminated.

Second, when a failure occurs in a tightly coupled multiprocessing configuration, the hardware components of the failing system are available to assist in the recovery. In the case of a CPU failure, the functional CPU is available to perform the recovery. These capabilities are not provided in a configuration with two uncoupled systems.

A tightly coupled multiprocessing configuration also provides a better method of handling preplanned scheduled maintenance activities for installations that operate 24 hours a day, since a CPU and its components can be physically removed from the multiprocessing configuration without the necessity of job cancellation and system restart.

Functions designed to increase system availability are basic to the structure of OS/VS2 MVS. For both uniprocessor and multiprocessor systems, OS/VS2 MVS is designed to attempt to reduce the frequency of errors that occur as the result of programming and to reduce the impact of both hardware and programming errors when they do occur, such that system terminations are avoided more frequently than when another operating system, such as OS MVT, is used.

Specifically, the implementation of one virtual storage for each user in OS/VS2 MVS decreases the chance that one user will inadvertently modify virtual storage of another user or critical portions of the control program. Many integrity features designed to prevent program errors are also basic to the system. Several other features that are unique to OS/VS2 MVS are provided to reduce the impact of errors.

First, functional recovery routines (FRRs) for certain system functions are provided. An FRR is a tailored recovery routine for a specific system function. It uses data stored during execution of the system function in an attempt to repair that function when a failure occurs, so that system operation can continue without a re-IPL.

Second, two primary system residence volumes, each on a different physical path to the system configuration, are maintained. If a permanent hardware error occurs in the path to the residence volume currently being used, OS/VS2 MVS automatically switches to the backup system residence device without termination of system operation.

Third, OS/VS2 MVS eliminates many system terminations that would otherwise occur as the result of subchannel errors and control unit lockups by issuing the System/370 CLEAR I/O instruction to reset byte and block multiplexer subchannels when necessary.

Fourth, OS/VS2 MVS supports operator-initiated recovery via the restart key. If the operator suspects the system is in a loop or uncoded wait state, instead of re-IPLing and restarting the system, he can press the restart key, which gives control to the recovery termination manager in OS/VS2 MVS. The recovery termination manager initiates normal recovery processing, using FRRs and other recovery routines to recover the operating system without a system termination.

OS/VS2 MVS multiprocessing support also includes additional recovery facilities designed to further improve the availability offered by a Model 168 tightly coupled multiprocessing configuration. When a hard CPU failure occurs in one CPU, the alternate CPU recovery (ACR) facility is invoked in the functional CPU to recover from the CPU failure so that system operation continues uninterrupted without the failing CPU. Channel reconfiguration hardware in the Model 168 is used to recover I/O operations in progress in the system with the failing CPU.

After CPU and I/O recovery have been performed in a multiprocessing environment, OS/VS2 MVS automatically varies offline the malfunctioning CPU and all channel paths to it. If the primary system console was attached to the failing CPU, an automatic switch to the console of the functioning CPU is made by OS/VS2 MVS.

Less Complex Operational Requirements

A Model 168 tightly coupled multiprocessing configuration has less complex operational requirements than two uncoupled Model 168 systems because it presents a single system image to the operator even though there are two systems in the configuration. The operator has one operational interface to the entire system, one job scheduling interface, and one point of control for all the resources in the configuration. In addition, the operator must communicate with and control only one control program instead of two.

Improved Resource Utilization

Resource utilization in a tightly coupled multiprocessing configuration is improved over that of two uncoupled systems because load leveling occurs between the two systems, there is a reduction in the amount of processor storage required by the resident control program, there is I/O device pooling, and the need for using Shared DASD support is eliminated.

Load leveling occurs for CPU processing and I/O processing because of the way in which OS/VS2 MVS can schedule task execution and I/O operations in a tightly coupled multiprocessing configuration. Load leveling reduces the peak and valley periods of CPU and I/O utilization that normally occur in two uncoupled systems, as follows.

The two CPUs are considered to be system resources that, when available, are allocated to ready tasks. Usually, either CPU is capable of processing each task in the system. Thus, as soon as a CPU becomes available, it is allocated to the highest priority queued ready task. Since there are on the average twice as many tasks in a multiprocessing configuration as in one system in a two uniprocessor environment, the chances that no task in the multiprocessing system is ready to execute and available CPU time will be unutilized are significantly reduced. An I/O operation is begun on any available channel path in the total configuration and is not limited to being started only on the channel path(s) in one system.

Since there is only one copy of the OS/VS2 MVS multiprocessing control program resident in processor storage in a tightly coupled multiprocessing configuration, more processor storage is available for paging (which can benefit performance) than in two uncoupled systems, each of which has an OS/VS2 MVS uniprocessor control program resident.

I/O devices with programmable channel- or string-switching features installed are pooled in a tightly coupled multiprocessing configuration for use by both CPUs. More than half the total number of switched I/O devices can be allocated to an individual job when necessary. The pooling of I/O devices and sharing of processor storage permits the execution of jobs with larger processor storage and I/O device requirements than is possible using one system in a configuration with two uncoupled systems.

The sharing of I/O devices and processor storage also enables the OS/VS2 MVS control program to automatically handle peak load situations within jobs and to balance the workload across systems. Manual balancing of the workload between two systems, as is required for two uncoupled systems, is not required for a tightly coupled configuration.

The pooling of I/O devices can reduce the total number of I/O devices required in a tightly coupled multiprocessing configuration when compared with the I/O device requirements for two uncoupled systems that are to handle the same large I/O job or peak load direct access storage requirements.

Since there is only one control program, there is no need to split any data base into two parts, one for each system, or to use Shared DASD support in order to share a data base between the two systems, as is required for two uncoupled systems. The use of Shared DASD support results in reduced throughput for two uncoupled systems because of the interference it introduces. This throughput reduction is not incurred in a tightly coupled multiprocessing configuration since there is only one control program and it can maintain the integrity of a shared data base without using Shared DASD support.

Operational Flexibility

A tightly coupled Model 168 multiprocessing configuration can be divided into two systems that operate in uniprocessor mode when this is required to handle special environment situations. For example, a uniprocessor mode system might be required for planned preventive maintenance, as a test system for a system programmer, or to run a programming system other than OS/VS2 MVS (such as VM/370).

The two systems in a multiprocessing configuration can be divided such that only the hardware components (processor storage and I/O devices) actually required for the special environment system are allocated to one CPU, leaving the balance of the hardware resources available for normal production processing. When two uncoupled systems are present in an installation, one total system (half the total hardware resources) must be allocated to the special environment system regardless of its actual processor storage and I/O device requirements (except for any switched I/O devices).

In addition to being able to tailor the resources of a special environment uniprocessor mode system, an installation can perform the reconfiguration dynamically in a multiprocessing configuration without the necessity of stopping the system, canceling or interrupting jobs, or quiescing the system.

In an installation with two uncoupled systems, production operations in one system must be stopped or allowed to quiesce before the system can be used for the special environment operation. Then production processing must be restarted after the special processing is completed. Thus, productive capability is lost during the time the system is being quiesced and later during restart operations. This productive capability is not lost in a tightly coupled multiprocessing configuration.

Note also that there is very little possibility of losing half of the hardware components of a tightly coupled multiprocessing configuration as a result of one failure. If a CPU in one system fails, processor storage and switchable devices in that system can still be used by the other processor. If all processor storage in one system fails, the CPU, channels, and I/O devices in that system are still available to the multiprocessing configuration. In an environment with two uncoupled systems, a failure in one system causes a loss of the entire system except for any I/O devices that can be switched to the other system.

Improved Growth Options

The installation of tightly coupled multiprocessing in an environment with two uncoupled Model 168 systems to handle added workload or the addition of an application is an alternative, less expensive growth step than the installation of a third uncoupled system.

Assume the two uncoupled systems can handle the current workload but the installation of another data base application or more terminals, for example, would cause unacceptable performance during peak load periods. Installation of tightly coupled multiprocessing would enable the resources of both systems to be utilized more efficiently during peak load periods and probably could provide acceptable performance. If the workload continued to grow and additional resources were required, a third system (Model 145 or higher) could be installed and loosely coupled with the existing tightly coupled configuration to provide the desired performance.

Improved Throughput Possibilities

As a result of certain hardware and programming interference that occurs, the internal performance of a Model 168 tightly coupled multiprocessing configuration is in the area of 1.5 to 1.9 times that of a uniprocessor Model 168. However, because of the load leveling that occurs in a tightly coupled configuration, the throughput achieved for a given job stream can be greater than that achieved when the same job stream is split and processed by two uncoupled systems with total resources equal to those of the two coupled systems.

The reason the throughput potential is greater for the tightly coupled configuration is that load leveling enables most CPU and I/O time that is unoverlapped when two job streams execute in two uncoupled systems to be overlapped when the same two job streams execute in a tightly coupled configuration, as follows.

In a single system environment, unoverlapped I/O time occurs when there are no tasks ready to execute until some I/O completes and unoverlapped CPU time occurs when no I/O is operating and none can be started until certain CPU processing completes. When two job streams execute in two uncoupled processors, unoverlapped CPU and I/O time occur in both systems and the chances that unoverlapped CPU time or I/O time occurs simultaneously in the two systems are small.

When these two job streams execute in a tightly coupled multiprocessing system, because of the way work is dispatched (in effect, each CPU can process the other CPU's job stream), most unoverlapped CPU time that occurs in one system in the uncoupled environment is overlapped with unoverlapped I/O time that occurs in the other system in an uncoupled environment and vice versa. Studies and installation experience have shown that when the two job streams involved are relatively unbalanced as far as CPU utilization is concerned (80 percent for one and 50 percent for the other, as an example, instead of both 80 percent), the potential for increased throughput in a tightly coupled configuration, as a result of load leveling, is greater.

Greater throughput potential for a tightly coupled multiprocessing configuration also results from the other advantages such a configuration offers, such as the (1) availability of more processor storage for paging, (2) elimination of Shared DASD support, (3) elimination of lost productivity as a result of quiesce and restart operations, (4) availability of more of the total system resources for production processing when a special uniprocessor mode system is required, and (5) reduction in job reruns, data set rebuilding, and re-IPLing as a result of the improved availability features that reduce the number of system terminations.

The performance of an individual application in a tightly coupled multiprocessing environment versus that achieved in a uniprocessor system is related to the amount of multitasking within the application and how well the processing is balanced among tasks. Best performance is achieved when the work required to process a given transaction is distributed fairly evenly across the tasks that are to process the transaction. Good performance can be achieved when no task performs more than 50 percent of processing for a transaction.

ADVANTAGES OF LOOSELY COUPLED MULTIPROCESSING CONFIGURATIONS

Like tightly coupled multiprocessing configurations, loosely coupled multiprocessing configurations offer advantages over single system and uncoupled multiple system configurations, such as higher availability (via better recovery techniques and serviceability) and operational efficiency and flexibility (via a single input stream, a single operator interface, pooled I/O resources, automatic workload balancing, and better handling of peak load job conditions).

In addition, however, loosely coupled multiprocessing configurations offer certain advantages over tightly coupled multiprocessing configurations, as follows:

- Larger growth potential and growth without disruption. Many more systems can be coupled in a loosely coupled than in a tightly coupled configuration and additional systems can be added with relatively little disruption to operations. Because System/370 tightly coupled multiprocessing is limited to a maximum of two systems, the addition of a third system can be accomplished only by going to a loosely coupled configuration.

- Loosely coupled systems can be asymmetric. Different System/370 models with unlike hardware characteristics (CPU speed, number of channels, and processor storage size) can be loosely coupled. Although processor storage symmetry is not required for Model 168 and Model 158 Model 3 tightly coupled multiprocessing configurations, model symmetry (two Model 168 or Model 158 multiprocessor systems) is required. For a Model 158 Model 1, processor storage size symmetry is also required unless an RPQ that permits asymmetric processor storage configurations is installed.

- A larger range of models can be loosely coupled. JES3 supports Models 145, 155 II, 158, 165 II, 168, 158 multiprocessing, 168 multiprocessing, and 168 Attached Processor systems (running under OS/VS2 MVS) in a loosely coupled configuration. The configuration can range from a minimum of one Model 145 with 1024K of processor storage to a maximum of eight Model 168 tightly coupled multiprocessing systems that are loosely coupled.

A tightly coupled multiprocessing configuration has one significant advantage over a loosely coupled configuration. While either system can handle peaks in the number of jobs processed, only a tightly coupled system can handle peaks that occur within a single job (such as an online data base application), since the CPU, processor storage, and I/O devices of both systems can be used for the job during peak requirements. The unique advantages of both types of multiprocessing configurations can be obtained when required by combining the two within an installation.

In addition to the general benefits of tightly and loosely coupled multiprocessing, Model 168 tightly and loosely coupled multiprocessing configurations offer large-system users the following specific advantages:

- Lower-cost entry into a tightly coupled multiprocessing environment than was previously available with System/360

- Compatible growth for Model 65, 155, 155 II, 158, 165 II, and 168 uniprocessor systems

- Compatible growth for multiple-system installations with System/360 and/or System/370 models

- A growth path for Model 165 II and purchased Model 165 uniprocessor systems

- All the new and enhanced hardware and programming systems reliability, availability, and serviceability features provided in uniprocessor Model 168 systems, which are of even greater benefit in a multiprocessing environment

- All the new hardware and new I/O devices offered by the Model 168 (more processor storage and channels, faster I/O devices, more subchannels, significantly higher capacity online direct access storage, etc.) that are required for online applications

- Enhancements to Model 65 multiprocessing architecture and programming systems support

- Support of all the hardware and programming systems features provided for Model 168 uniprocessor systems operating in EC and DAT modes, including support of integrated emulation during multiprocessor mode operations

60:10 MODEL 168 MULTIPROCESSING ARCHITECTURE

UNIPROCESSOR AND MULTIPROCESSOR HARDWARE DIFFERENCES

The following identifies the major areas of architectural and hardware implementation differences between Model 168 uniprocessor and tightly coupled multiprocessor systems (that is, facilities implemented in multiprocessor but not uniprocessor systems):

- 3068 Multisystem Communication Unit - a new required unit that contains the hardware required for communication between two Model 168 multiprocessor systems and the configuration control panel
- Prefixing - a method of assigning unique areas of processor storage to addresses 0 to 4095 for each CPU
- CPU addressing and STORE CPU ADDRESS instruction - required to specifically identify each CPU
- Time-of-day clock - synchronization of the two physical clocks to provide one logical clock for the multiprocessing configuration
- Interprocessor programmed communication (SIGNAL PROCESSOR instruction) - required to enable a CPU to request services of the other CPU and to alert it to conditions to which it must respond during multiprocessor mode operations. For example, this capability is used during the initialization of multiprocessor mode operations, for reconfiguring hardware components, and in recovery procedures that occur after a CPU failure.
- Interprocessor hardware communication - required to alert a CPU to conditions in the other CPU and to synchronize certain operations in both CPUs during multiprocessor mode operations
- Channel reconfiguration hardware - used to expand the recovery provided by the OS/VS2 multiprocessing control program after a CPU failure occurs during multiprocessor mode operations

The instructions included in the multiprocessing feature, which are valid in uniprocessor as well as multiprocessor mode, are the following:

SET PREFIX
STORE PREFIX
STORE CPU ADDRESS
SIGNAL PROCESSOR

The model-dependent extended CPU logout area contains additional information in multiprocessor Model 168 models, such as the configuration in effect and the value in the prefix register at the time of the interruption.

3068 MULTISYSTEM COMMUNICATION UNIT

The 3068 Multisystem Communication Unit is required in every Model 168 multiprocessing system. The configuration control panel is mounted on the 3068 unit. The switches and controls on this panel are shown in Figure 60.10.1. The configuration control panel is used to establish the desired mode of system operation and to physically configure the hardware components (storage elements and I/O devices) of the operational system.

The operating mode (multiprocessor or uniprocessor), allocation of storage segments to CPUs, storage segment addressing, interleave mode (four-way or serial), and time-of-day clock and system oscillator control settings are indicated by the settings of the system mode, storage allocation, floating storage, interleave control, and oscillator control switches, respectively. These settings are not made effective until some time after one or both (depending on the system mode change being made) enter configuration pushbuttons are pressed. The configuration control panel contains two enter configuration pushbuttons, one for each CPU. The I/O allocation switches operate independently from the enter configuration pushbuttons.

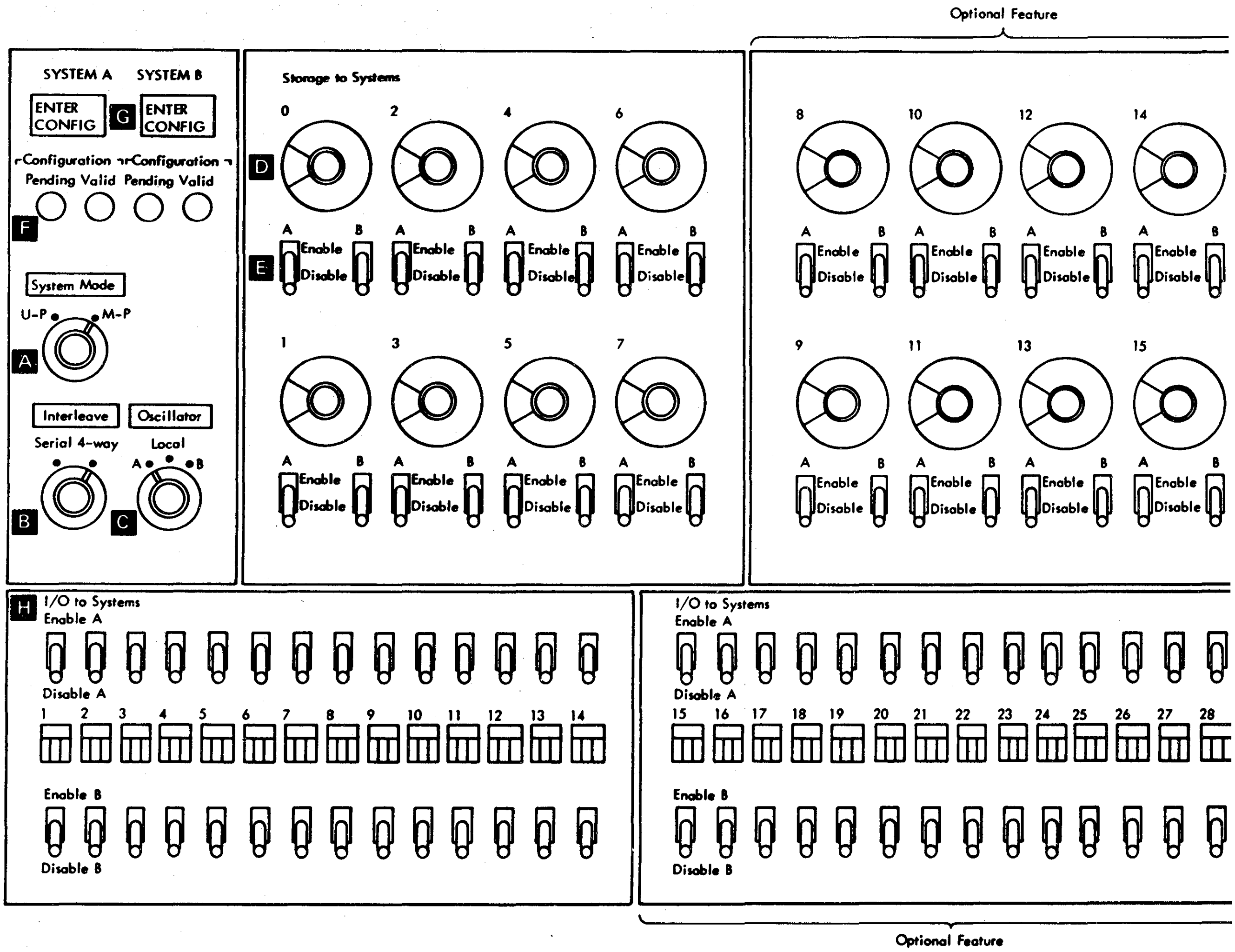


Figure 60.10.1. Configuration control panel for a Model 168 multiprocessing system

The configuration control panel is used to perform the following:

- Establish, via the system mode switch, multiprocessor or uniprocessor mode of operation for both CPUs. That is, both CPUs are set to multiprocessor mode or both are set to uniprocessor mode. Unlike a Model 65 multiprocessing configuration, no other combination is required.

  When the mode switch is set to the multiprocessor position, interprocessor communication hardware is enabled (SIGNAL PROCESSOR interface, high-speed buffer intercommunication, time-of-day clock oscillator, time-of-day clock security switch, time-of-day clock synchronization check, malfunction alert signal, and the broadcast of reset functions). These facilities are discussed in the remainder of this subsection.

  When uniprocessor mode is established, the interprocessor signals are deactivated and each CPU can operate independently from the other with that portion of available processor storage and the I/O devices enabled to it via the configuration control panel.

- Establish, via the storage allocation switches on the storage to systems portion of the configuration control panel, which storage elements are part of the physical configuration and the CPUs to which the storage segments are enabled. There is a pair of storage allocation switches, one for CPU A and one for CPU B, associated with each floating address rotary switch. Each storage allocation switch enables or disables the operation of its associated CPU with a storage element and the storage address range assigned to the storage segment via its floating storage address switch. A storage element is configured out of the physical processor storage configuration by disabling it from both CPUs.

  When multiprocessor mode is in effect, each storage address range and associated storage element that is enabled for one CPU must also be enabled for the other CPU. This requirement permits both CPUs to access all the processor storage currently enabled in the physical configuration. When uniprocessor mode is in effect, no storage address range and element can be enabled for access by both CPUs; however, storage address ranges and elements can be allocated such that one CPU has access to more processor storage than the other (storage allocation need not be symmetrical).

  Note that the storage allocation switches on the configuration control panel on the 3068 take precedence over the storage configuration panel on the two 3066 system consoles in the multiprocessing configuration. Only when all the storage allocation switches for a given CPU are in the disable position, the CPU is in uniprocessor mode, and none of its processor storage is enabled to the other CPU is the configuration panel on its associated 3066 console effective for processor storage configuration.

- Assign, via the floating storage address rotary switches, an address range to each enabled storage element. Each processor storage element in the multiprocessing configuration is assigned a unique element number. The eight even element numbers 0 to 14 are used for the processor storage of CPU A while the eight odd element numbers 1 to 15 are used for the processor storage in CPU B, as shown in Figure 60.10.2.

  Each of the 16 storage element numbers has an associated floating storage address switch that is used to assign a one-megabyte (1024K-byte) range of storage addresses to its corresponding storage element, also shown in Figure 60.10.2. The element numbers 8 to 15 and their associated floating storage address switches are present on the configuration control panel only for Model 168 multiprocessing configurations in which at least one CPU has more than four megabytes of processor storage installed.

  Any one of the 16 one-megabyte storage address ranges, which are listed on the floating storage address switches as 0-1M, 1M-2M, through 15M-16M, can be assigned to a storage element as long as the same address range is not assigned to more than one enabled storage element when multiprocessor mode is in effect or to two storage elements in the same system when uniprocessor mode is in effect. The storage address range 0-1M must be assigned to one enabled element when multiprocessor mode is in effect and to one enabled element in each system when uniprocessor mode is in effect; otherwise a successful IPL cannot be performed.

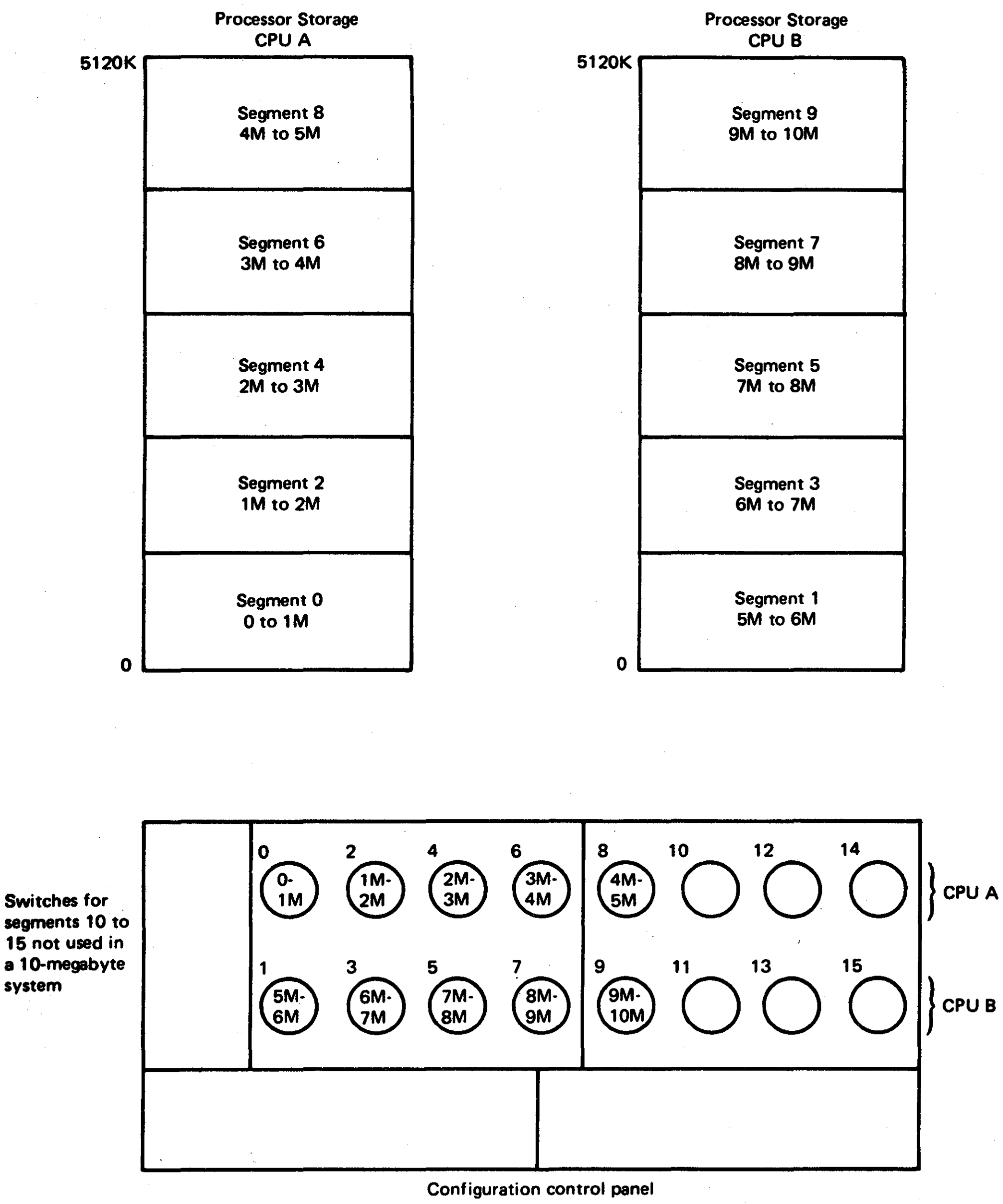


Figure 60.10.2. Storage elements and floating storage addressing

The storage ranges assigned to a given system for uniprocessor mode operations or to shared storage for multiprocessor mode operations need not be contiguous when OS/VS2 MVS (or TSS/370) is used as the operating system. In addition, address ranges higher than the total amount of processor storage in the system can be assigned.

For example, addresses 0 to 3M and 5M to 6M could be assigned to a multiprocessor mode configuration with four megabytes of processor storage enabled. The addresses between 3M and 5M are then marked unavailable by the OS/VS2 multiprocessing control program. When an operating system other than OS/VS2 MVS or TSS/370 is used for uniprocessor mode operations, processor storage in each system must be assigned contiguous addresses.

- Enable or disable, via the I/O allocation switches, access by each CPU to up to 14 or, optionally, 28 switched control units and/or switched direct access device strings. Each switching feature installed that is to be controlled by the I/O allocation switches must have installed the optional Remote Switch Attachment feature (which is a no-charge feature except for the 3803 control unit for 3400-series tape units) to connect the control unit directly to the configuration control panel. The devices listed previously that have a programmable two-channel switch also have a Remote Switch Attachment feature available.

  A pair of I/O allocation switches, one per CPU, is assigned to each control unit connected to the configuration control panel. When four-channel switching is installed on a control unit, two pairs of I/O allocation switches are assigned. As with storage segments, each control unit can be enabled for access by one or both CPUs or disabled for access by both CPUs. A square space is provided below each pair of I/O allocation switches to contain the two-digit hexadecimal address of the associated control unit.

  When a CPU issues an I/O instruction to an I/O device connected to a control unit that is disabled from that CPU, a "not operational" indication results. When the status of an I/O allocation switch is changed, the new status becomes effective when the control unit/channel interface becomes inactive.

- Establish, via the interleave control switch, four-way interleaved or serial mode of operation for all enabled processor storage in the multiprocessing configuration. Note that floating storage addressing is functional for both serial and interleaved mode.

- Establish, via the oscillator control switch, the system and time-of-day clock oscillators to be used by each CPU. In a Model 168 multiprocessing configuration, the system oscillator for each CPU is contained in its associated half of the 3068 unit and the system oscillator in each CPU is disabled. This is done so that a CPU need not be operational in order for its processor storage to be accessed by the other CPU. Each system still has its own time-of-day clock oscillator in its CPU.

  When the oscillator control switch is set to the A position, both time-of-day clocks operate from the time-of-day clock oscillator in CPU A and both CPUs use the system oscillator in the physical half of the 3068 associated with CPU A. When the switch is set to the B position, both CPUs use the two oscillators for CPU B. When the local position is selected, each system operates using its own system and time-of-day clock oscillators.

  The local position is valid only for uniprocessor mode and when processor storage is not cross-configured. However, the local position is not required for uniprocessor mode. Thus, if a switch from multiprocessor to uniprocessor mode is made, there is no need to change the oscillator control switch to local. Both systems continue to use the system and time-of-day clock oscillators for the CPU indicated by the switch.

  The design of the two oscillators is such that the oscillator control switch can be set to another position during a planned reconfiguration during system operation without disruption to the system or loss of synchronization of the two time-of-day clocks. Note also that if power fails or is removed from the 3068 half whose system oscillator is currently being used, the other powered system oscillator will automatically be made the effective oscillator by hardware logic. Since this action overrides the system oscillator specified via the configuration control panel, the valid configuration indication light (discussed below) goes off and remains off until the oscillator control switch is changed to select the oscillator being used.

The enter configuration pushbutton for a CPU operates with the half of the 3068 unit associated with that CPU. However, when multiprocessor mode is in effect, either enter configuration pushbutton can be pressed to change the configuration of one or both systems, including a change from multiprocessor to uniprocessor mode for both systems. That is, pressing either button causes the configuration specified via the configuration control panel to be entered in both systems if it is valid.

When both CPUs are in uniprocessor mode, only the enter configuration pushbutton for a CPU must be pressed to change the configuration for that CPU, as long as uniprocessor mode is to remain in effect. To change from uniprocessor to multiprocessor mode, either enter configuration button can be pressed.

Note that when processor storage is cross-configured or a change that affects both CPUs (such as switching the oscillator control switch from one CPU to the other) is made when uniprocessor mode is in effect, pressing the enter configuration pushbutton for either CPU causes the configuration specified via the configuration control panel to be entered in both CPUs. If neither of these two conditions exists, pressing the enter configuration pushbutton for a CPU in uniprocessor mode causes a configuration to be entered only for that CPU.

There is a pending indicator and a valid indicator for each CPU located below its configuration pushbutton on the configuration control panel. When a valid configuration is indicated in the switches, one or both valid indicators will be lit, as appropriate. When multiprocessing mode is set, both valid indicators must be on before the configuration can be established. When uniprocessor mode is set, the configuration will be set in a CPU if its valid indicator is on, regardless of the setting of the valid indicator for the other CPU. If all power is off in a system, its corresponding valid indicator will be off.

The following are all the conditions that cause one or both of the valid indicators to be turned off. Any one or more of these conditions prevents the specified configuration from being established after one or both configuration pushbuttons are pressed:

- More than one floating storage address switch is set to the same address range when multiprocessor mode is set or for one CPU with uniprocessor mode set (double addressing).

- The system mode switch indicates uniprocessor mode and a storage element is enabled to both CPUs (shared storage in uniprocessor mode).

- The system mode switch indicates multiprocessor mode and a storage element is enabled to one CPU but not to the other (partial sharing).

- One or more storage elements in CPU A are enabled to CPU B (cross-configured) and the half of the 3068 unit associated with CPU A is powered down. The reverse is also invalid (cross-configured powered down 3068 half).

- An element is enabled that is not powered up or installed. For example, in a configuration with a total of four megabytes installed (say two megabytes in each system), only the storage allocation switches for elements 0, 1, 2, and 3 can be set to the enable position (unavailable storage).

- The system mode switch is set to multiprocessor mode and the oscillator control switch is set to local (split oscillator).

- Processor storage is cross-configured and the oscillator control switch is set to local (cross-configured split oscillator).

- The oscillator control switch is set to a CPU that has its associated half of the 3068 unit powered down (unpowered system oscillator).

- Either CPU or its half of the 3068 unit is powered down (or not installed) and multiprocessor mode is set (partial multiprocessing system).

Some possible valid system configurations are the following:

- A full multiprocessing configuration, that is, both systems set to multiprocessor mode with all available processor storage enabled to both systems. This is the normal configuration.

- A full uniprocessor configuration, that is, each system set to uniprocessor mode with the processor storage of each CPU enabled only to that CPU (storage is nonshared and not cross-configured)

- Each system set to uniprocessor mode with processor storage of only one of the systems available. The available processor storage is divided between the two systems (each element enabled to only one system). A minimum of two megabytes of processor storage must be available in one system for this configuration.

- Each system set to uniprocessor mode with one megabyte of storage allocated to one system only (a maintenance or test system) and all the remaining available storage allocated only to the other system.

- Each system set to multiprocessor mode with the processor storage of only one system enabled to both systems. A minimum of one megabyte of processor storage must be available in either system for this configuration.

- One system set to uniprocessor mode with all available storage from both systems enabled to it. The other system is not part of the production processing configuration, and diagnostics that do not require processor storage can be executed on it if desired.

Note that when multiprocessor mode is in effect, a malfunction in a CPU may cause the CPU to be logically removed from the operational configuration by the OS/VS2 MVS multiprocessing control program. Both systems are still in multiprocessor mode and can remain in this mode until such time as something is to be executed in the malfunctioning CPU. At this time, uniprocessor mode must be established for both systems.

If an invalid configuration is specified when the enter configuration pushbutton(s) is pressed, the valid indicator(s) stays off until a valid configuration is placed in the switches and the configuration pending light(s) is not turned on. After the invalid switch settings are corrected, the enter reconfiguration pushbutton(s) must be pressed again in order to generate an enter configuration signal.

When an enter configuration pushbutton is pressed, an enter configuration signal is generated when a valid configuration is indicated but the specified configuration does not become effective until the next time one or both (depending on the mode in effect) CPUs enter the wait (enabled or disabled) or stopped state. The configuration currently in effect is maintained in two sets of registers, one in each physical half of the 3068 MCU.

When multiprocessor mode is in effect, the specified configuration does not become effective until the next time both CPUs are in the wait or stopped state simultaneously. When uniprocessor mode is in effect and processor storage is not cross-configured, a change affecting one system becomes effective as soon as that system only enters the wait or stopped state. When both CPUs are in uniprocessor mode and processor storage is cross-configured or a change is made that affects both CPUs (such as a change in the oscillator control switch setting), the specified configuration is not made effective until both CPUs simultaneously enter the wait or stopped state.

During the time an enter configuration request is pending, one or both pending indicators, as well as one or both valid indicators, are lit as appropriate. If any of the configuration switches controlled by the enter configuration pushbutton are changed while a valid configuration is pending, the configuration that is entered, if any, is unpredictable.

Note that after a system power-on sequence is completed, an enter configuration signal is generated to establish the configuration currently specified via the switches controlled by the enter configuration pushbutton if it is valid. If an invalid configuration is specified, one or both valid indicators remain unlit and the configuration is not entered as would occur if the enter configuration pushbutton had been pressed. However, the enter configuration signal is not generated when a unit that can be individually powered on and off (CPU, processor storage, etc.) is powered on. Therefore, the system configuration must be reestablished as appropriate any time a specific unit has been powered on.

Implementation of configuration pushbuttons, which are not present on the configuration control panel of a Model 65 multiprocessing system, prevents inadvertent alteration of the operating mode and storage segment allocation during system operation.

The implementation of the configuration control panel for the Model 168 permits a new hardware configuration to be entered during system operation without the possibility of introducing a hardware error, assuming the correct procedure is followed. Hence, system operations need not be quiesced prior to any alteration of the physical configuration using the configuration control panel, as is required prior to altering any switch on the configuration control panel in a Model 65 multiprocessing system.

Whenever an enter configuration pushbutton is pressed, all the entries in one or both (as appropriate) high-speed buffers and translation lookaside buffers are invalidated. TLB invalidation requires 5120 nanoseconds (80 nanoseconds per odd/even pair of entries).

The settings indicated by the configuration control panel cannot be obtained as a result of the execution of one instruction. (In a Model 65 multiprocessing system, execution of the DIAGNOSE instruction is necessary to obtain the settings of certain switches on the configuration control panel.) The OS/VS2 multiprocessing control program determines the storage addresses, channel paths, and I/O devices online during IPL by procedures similar to those used by uniprocessor control programs (TEST I/O instructions, invalid storage address indications, etc.). This technique for determining the online system configuration is System/370 model independent.

Power Control

Power control for Model 168 multiprocessing systems is designed to allow certain components within one Model 168 system to be powered on and off separately from other components of the given system without preventing operation of the other Model 168 system.

Model 168 multiprocessor system hardware and programming systems support are designed to enable a properly isolated system component to be powered down without the necessity of first quiescing system operations. This capability enables repair operations to be performed on a malfunctioning component while normal system operations continue, using the other system and functional components of the system with the malfunctioning component. When the malfunctioning component is repaired, it can be powered on and returned to the functional configuration, again without quiescing system operations.

Each of the system components that can be separately powered on and off has its own power control switch that can be set to the local or remote position. The switch must be set to local in order to perform separate power on and power off operations.

The following components within a given Model 168 system in a multiprocessing configuration can be powered on and off separately from the other components in the same system:

- The CPU frames of the 3168 Processing Unit together with the 3066 System Console

- All processor storage in the 3168 Processing Unit

- The channel control function in the 3168 Processing Unit

- The Integrated Storage Controls feature in the 3168 Processing Unit

- Each individual standalone channel for the system

- The physical half of the 3068 Multisystem Communication Unit associated with the system

- Individual control units and I/O devices attached to the channels of the system

When the CPU frames, processor storage, or half the 3068 is in the process of being powered up or down separately, the other system can experience machine check or extraneous interruptions unless certain configuration criteria are met even if the other system is in the stopped or wait state (since the channels can still be operating). Therefore, when a component of a system is to be separately powered off, the following configuration rules should be observed:

- To power off the CPU frames without or together with its dedicated channels or the channel control function in the 3168, uniprocessor mode must be in effect, no processor storage should be enabled to the CPU involved, all the I/O allocation switches for the CPU should be set to the disable position, and the oscillator control switch should be set to the other CPU. The processor storage in the powered down CPU can be utilized by the other CPU by enabling it only to the other CPU (cross-configuring).

  Note that the channel reconfiguration hardware in the powered up CPU cannot be enabled to access the channels and I/O devices attached to the powered down CPU. In order for channel reconfiguration hardware to be activated in one CPU (3168 Processing Unit) to control the channels of the other CPU, the other 3168 Processing Unit must be powered on (except for processor storage and the ISC feature).

- To power off processor storage in a CPU, none of the processor storage can be enabled to the other CPU. That is, all of the storage allocation switches for that processor storage (all even-numbered or all odd-numbered switches) must be in the disable position. The CPU containing the powered down processor storage can operate in multiprocessor mode, sharing the processor storage of the other CPU, or in uniprocessor mode with processor storage from the other CPU cross-configured to it. Processor storage would normally be powered down separately only to perform a maintenance function.

- To power off half of the 3068 unit, uniprocessor mode must be in effect, no processor storage can be cross-configured (physical processor storage in one CPU enabled only to the other CPU), the CPU associated with the 3068 half to be powered down must be in the stopped state, all the I/O allocation switches for the associated CPU should be set to the disable position, and the oscillator control switch must be set to the other CPU. Note that a CPU cannot operate when its associated 3068 half is powered down. A 3068 half would normally be powered down separately only to perform a maintenance function.

- To power off a channel, the interface disable switch for the channel must be on, all I/O allocation switches assigned to the devices attached to the channel should be in the disable position, and the CPU to which it is attached must pass through the wait or stopped state before the power off sequence begins.

- To power off the channel control function in the 3168, uniprocessor mode must be in effect, all the I/O allocation switches for the affected CPU must be in the disable position, the oscillator control switch must be set to the other CPU, and the affected CPU must be in the stopped state. Note that a CPU with its channel control function powered down cannot operate.

- To power off the Integrated Storage Controls feature, power must be off in the I/O devices attached to the ISC. In addition, if the ISC has any channel-switching features installed (but no string-switching and remote switch attachment features), the devices attached to the ISC must be varied offline from both CPUs and their I/O allocation switches must be set to the disable position.

The same configuration rules should be observed when powering on an individual component, although hardware interlocks prevent a nonfunctional configuration from being entered. Note also that for proper system operation, a component (CPU, processor storage, channel, or I/O device) should be logically varied offline to the operating system before it is powered off. A component must be varied online after it is again powered on if it is again to be part of the operational system.

Note that if a 3068 physical half or 3168 processing unit is to be powered down for the replacement of a part that is water-cooled (such as a regulator in the 3168) there is a procedure that allows the water cooling to be removed only from the part to be replaced instead of from the entire system containing the 3168 or 3068 half.

When the power control switch for a unit is set to the remote position, it is powered on and off under control of the power on and power off pushbuttons on the operator control panel of the two 3066 system consoles. Assuming all units in a multiprocessing configuration have their power control switch in the remote position, when the power on pushbutton for one system is pressed, power is turned on in the following units of the configuration:

- 3168 Processing Unit associated with the 3066 console used
- 3066 System Console whose power on button was pressed
- 3068 Multisystem Communication Unit (both halves)
- 3067 Power and Coolant Distribution Units for both systems
- All standalone channels attached to the system associated with 3066 console used
- All nonshared and shared units attached to the system associated with the console used (that is, I/O control units and I/O devices with their own integrated control unit). A shared unit is one that is attached to both systems via a channel or string switch.
- Processor storage in the system whose 3066 console was not used

When the power on button on the other 3066 console is then pressed, power is turned on in that 3066, the remaining portions of the 3168 processor associated with the console used, as well as this processor's standalone channels and nonshared I/O devices. Assuming all units in the multiprocessing configuration have their power control switch in the remote position, when the power off pushbutton on one system is pressed, power is turned off in the following units:

- All components of the 3168 Processing Unit associated with the 3066 console used except processor storage
- 3066 System Console whose power off button was pressed
- All standalone channels attached to the system associated with the 3066 console used
- All nonshared units attached to the system associated with the 3066 console used

When the power off button on the other 3066 console is pressed, the remaining powered on units are powered off. When the emergency power off switch in either system is pulled, power is removed from all units in the entire multiprocessing configuration (both systems).

The optional Power Warning feature can be installed in one or both of the CPUs in a Model 168 multiprocessing system. If the feature is installed on only one CPU, an undervoltage condition will generate a power warning machine check condition only in that CPU during multiprocessor mode operations. If all processor storage is to be dumped after the interruption is taken, power must be on in both systems and both systems must be partially protected with uninterruptible power supplies. That is, uninterruptible power supplies must be provided for both 3168 processors, all channels in both systems, the 3068 MCU, and those control units and devices that are to receive the storage dump.

If the Power Warning feature is installed on both CPUs in the multiprocessing configuration, each CPU is capable of recognizing an undervoltage condition. A cable connecting the two 3066 consoles in the configuration can be installed that makes the power warning interruption available to both CPUs when it occurs in either system during multiprocessor mode operations. The power warning interruption is then processed by whichever CPU is first enabled to receive such interruptions. Both systems must be partially protected with uninterruptible power supplies (as described previously) in order for all processor storage to be dumped.

When the Power Warning feature is installed on only one CPU, during uniprocessor mode operations a power warning interruption is generated only in the system containing the feature when an undervoltage condition occurs and only the processor storage enabled to that CPU can be dumped to disk. The Power Warning feature must be installed on both systems in order for a power warning condition to be recognized and handled in both systems during uniprocessor mode operations. Uninterruptible power supplies must be provided for a control unit and disk device in each system that is to receive processor storage dumps.

PREFIXING

Each CPU in a shared storage multiprocessing configuration must have a unique area of storage to be used for permanently assigned locations and logout areas, which in the Model 168 are contained in addresses 0 to 1938. Since there is only one set of storage locations with these addresses in shared processor storage, a means of assigning these addresses to two different storage areas, one for each CPU, is required. Prefixing, sometimes referred to as direct address relocation (not to be confused with dynamic address translation), is the technique used. Prefixing is applied to real storage addresses only, that is, to translated addresses when dynamic address translation is operative.

The implementation of prefixing in a Model 168 multiprocessing system permits the assignment of addresses 0 to 4095 to any storage area of 4096 bytes that begins at an address that is a multiple of 4096. (Addresses up to 4095 are prefixed so that the prefixed area for each CPU can contain certain control blocks that are required for each CPU, which in turn, simplifies control program coding). A 4K-byte storage area that is assigned to contain addresses 0 to 4095 for a given CPU is called a prefixed storage area (PSA).

Storage addresses are prefixed by means of a prefix value register. Each CPU has its own prefix value register in the processor storage control function (PSCF) and, for proper system operation, a unique 12-bit prefix value must be assigned for each CPU during multiprocessor mode operations. Prefix registers are initialized to zero during system reset.

The SET PREFIX and STORE PREFIX privileged instructions are provided to access the prefix register. SET PREFIX is partially retryable. STORE PREFIX is totally retryable. A given CPU can set or store only its own prefix register. A CPU cannot address the prefix register of the other CPU.

SET PREFIX is used to place a 12-bit prefix value in the prefix register in bit positions 8 to 19. Bits 0 to 7 and 20 to 31 of the prefix register are ignored. The translation lookaside buffer is purged when a SET PREFIX instruction is issued. STORE PREFIX can be used to place the prefix value contained in a prefix register in processor storage. Bits 0 to 7 and 20 to 31 are stored as zeros.

Prefixing operates as follows. When a CPU references a real storage address in the range of 0 to 4095 (the high-order 12 bits, 8 to 19, of the real 24-bit storage address are zeros), bits 8 to 19 of the prefix register for that CPU are added to bits 8 to 19 of the real address. The new address will then point to a location in the PSA of the CPU. This is called forward prefixing. When a CPU references a real storage address in the 4096-byte block that is pointed to by its prefix register (that is, an address in its own PSA), zeros are substituted for bits 8 to 19 of the real storage address so that an address range of 0 to 4095 results. This is called reverse prefixing. Reverse prefixing essentially provides the capability for either CPU to access locations 0 to 4095.

The prefix register contents can be displayed on the CRT screen on the 3066 console when the CE mode of operation is selected. The prefix register value can be altered manually using the system control panel on the 3066 console.

Prefixing is always active and is not subject to mode control. That is, prefixing always occurs in a Model 168 that has the multiprocessing feature installed, whether the CPU is in uniprocessor or multiprocessor mode. Prefixing is applied to all real storage references made by a CPU. It is implemented in hardware and does not affect instruction execution time. Channel hardware references to locations 0 to 4095 (for a channel address word, during channel status word storing, etc.) are also prefixed.

References by a channel to a channel command word (CCW), I/O data, indirect data address lists, and CPU extended logout area locations, either during system operation or IPL, are not prefixed, as they are in a Model 65 multiprocessing system. This approach permits channel programs (CCW lists) and I/O buffers that are contained in the PSA of one CPU to be executed by the other CPU without the necessity of moving the channel program and buffers to the PSA of the other CPU. Prefixing is also not performed during a store status operation.

The implementation of prefixing in a Model 168 multiprocessing system is different from that used in a Model 65 multiprocessing system. The PSAs in a Model 65 configuration are defined to be 0 to 4095 and the highest addressed 4K bytes online, and these locations cannot be varied. In addition, a PSA is assigned to a CPU by the operator via the Model 65 configuration control panel rather than by programming. The Model 168 implementation has the advantages of flexibility (PSAs can be placed anywhere) and of transparency to the operator, who need not be concerned with establishing the location of PSAs.

CPU ADDRESSING

In a multiprocessing configuration, each CPU has a unique four-bit address, which is stored during certain external interruptions to identify the CPU involved. The CPU address is also used in the SIGNAL PROCESSOR instruction, which is discussed under "Interprocessor Programmed Communication".

Addresses 0000 and 0001 are used, respectively, for the two CPUs, CPU A and CPU B, in a Model 168 multiprocessing system. These addresses are permanently assigned during system installation and are effective during both uniprocessor and multiprocessor operations. The STORE CPU ADDRESS privileged instruction is provided only in multiprocessor (and Attached Processor) systems to enable a program to obtain the address of the CPU in which it is executing.

An example of a situation in which this instruction is issued is after the IPL of a multiprocessing configuration. IPL from either CPU is possible. Therefore, STORE CPU ADDRESS is issued by the control program to determine the CPU in which it is operating so that the other CPU can be started via execution of a SIGNAL PROCESSOR instruction, which requires a CPU address.

STORE CPU ADDRESS should not be confused with STORE CPU ID, which is a System/370 instruction implemented in both uniprocessor and multiprocessor models. The STORE CPU ADDRESS instruction is totally retryable.

TIME-OF-DAY CLOCK

Because there are two time-of-day clocks in a multiprocessing configuration, one in each CPU, modifications to uniprocessor time-of-day clock hardware are required to ensure that both clocks contain the same time during multiprocessing operations with both CPUs online. Logically, there must be only one clock in the multiprocessing system. When the two CPUs operate as independent processors (in uniprocessor mode), these changes are disabled so that the two time-of-day clocks operate independently from one another.

When the two CPUs are operating in multiprocessor mode, a SET CLOCK instruction can be executed by either CPU, if the time-of-day clock security switch on either CPU is held in the enable set position. An enable set condition on one CPU is broadcast to the other CPU to accomplish this clock-setting function. When a time-of-day value is set in one clock, the OS/VS2 multiprocessing control program ensures that the identical time is contained in the other clock.

When multiprocessor mode is in effect, stepping pulses for both time-of-day clocks are provided by only one of the two time-of-day clock oscillators so that both clocks step synchronously. The time-of-day clock oscillator to be used is controlled by the setting of the oscillator control switch on the configuration control panel, as discussed previously.

The values of the two clocks are checked for synchronization by a combination of OS/VS2 multiprocessing support and hardware. OS/VS2 compares bits 0 to 31 of two time-of-day clocks to determine an out-of-synchronization condition in the high-order portion of the clocks. The synchronization of bits 32 to 51 in each clock is checked by hardware by the use of a one-second synchronization pulse that is broadcast by each CPU to the other CPU.

A time-of-day clock synchronization check external interruption condition is generated (normally simultaneously in both CPUs) when bits 32 to 51 of the two clocks are out of synchronization (differ by more than 960 nanoseconds). This interruption is maskable by the external mask in the current PSW and bit 19 in control register 0. The interruption condition can then be handled by whichever CPU enables i.self for synchronization checks first.

A synchronization check interruption condition continues to be presented until (1) bits 32 to 51 in the two clocks are equal or differ by less than 960 nanoseconds or (2) the latch indicating multiprocessing mode is in effect turns off. This interruption is enabled only during the times that the OS/VS2 control program is checking clock synchronization, such as after IPL, setting the time-of-day clock, and varying a CPU online.

So that unique time-of-day values are provided in the event that the CPUs simultaneously issue a STORE CLOCK instruction, the clock value stored contains a zero in bit 52 for CPU 0(A) and a one in bit 52 for CPU 1(B). If a STORE CLOCK instruction is issued for a time-of-day clock that is in the stopped state, bit 52 is not stored.

Modification of the hardware implementation of the interval timer for operation in a multiprocessing environment is nc. required. Only one interval timer is used. Unlike the time-of-day clock, the interval timer is located in processor storage (in a PSA) and, therefore, the timer can be addressed by both CPUs no matter which one of the two interval timers is used by the multiprocessing control program.

INTERPROCESSOR PROGRAMMED COMMUNICATION

The SIGNAL PROCESSOR (SIGP) privileged instruction is provided in Model 168 multiprocessor systems to support program-initiated communication between the two CPUs. Its operands indicate the address of the CPU being signaled, the function to be performed by the addressed CPU (via a one-byte order code), and a register in the signaling CPU in which status information from the addressed CPU can be placed, if necessary. A CPU can signal itself by placing its own CPU address in the SIGP instruction when uniprocessor or multiprocessor mode is in effect.

SIGNAL PROCESSOR is used in Model 168 multiprocessing systems rather than WRITE DIRECT, the instruction used for communication between Model 65 multiprocessor systems, as the former provides a more flexible method of interprocessor communication. The SIGNAL PROCESSOR instruction is not retryable.

When a SIGP instruction is executed, the addressed CPU receives the signal via the hardware interface between the two processors. The addressed CPU must be operating in multiprocessor mode in order to receive a SIGP instruction. The addressed CPU decodes the order to be performed, performs the operation if possible, and sends a response back to the signaling CPU to indicate the action taken. The response consists of a condition code setting and, in some cases, a set of status bits.

Status bits are presented in response to a SIGP instruction when the addressed CPU cannot perform the indicated function (because another external call is pending, the addressed CPU is in the stopped or check stopped state, the operator is performing an alter or display operation, for example).

The orders that can be specified in the SIGP instruction are the following:

Sense. The addressed CPU is requested to respond with an indication of its status (operating, stopped, not operational, not ready, external call pending, reseting, for example). A sense could be issued by a CPU at IPL, for example, to determine whether the other CPU is online.

External Call. An external call type of external interruption condition is generated in the addressed CPU, and is taken if the addressed CPU is enabled for these interruptions (via bit 18 in control register 0 and current PSW bit 7). If the addressed CPU is disabled for this interruption, the external call interruption remains pending. Only one such interruption for a given CPU can be pending at a time.

External call is issued by a CPU to request that a service be performed by the other CPU. The act of one CPU in a multiprocessing configuration sending a request to the other CPU via programmed communication is called "shoulder tapping".

The external call and emergency signal orders cause the CPU address of the CPU that issued the SIGP instruction to be stored in processor storage locations 132 and 133 when the external interruption occurs in the addressed CPU.

Emergency Signal. An emergency signal external interruption condition is generated in the addressed CPU and is taken if the addressed CPU is enabled for these interruptions (via bit 17 in control register 0 and current PSW bit 7). The interruption remains pending if such interruptions are disabled. Only one emergency signal interruption can be pending at a time.

Emergency signal, like external call, is issued by one CPU when a request is to be communicated to the other CPU. Emergency signal and external call functions are used to differentiate between two categories of requests. Emergency signal is used in OS/VS2 multiprocessing support primarily to signal a CPU failure via programming and to coordinate purging of the TLBs.

Start. The addressed CPU performs the same function as when its start key is pressed. The CPU on which IPL is performed issues a start to the other CPU, for example, during initialization of the configuration for multiprocessing operations.

Stop. The addressed CPU performs the same function as when its stop key is pressed. A CPU issues a stop to the other CPU, for example, when the addressed CPU has been varied offline by the operator.

Restart. The addressed CPU performs the same function as when its restart key is pressed. The restart occurs without a reset and the prefix register is not reset.

Initial Program Reset. The addressed CPU, its dedicated channels, and attached dedicated and shared I/O devices are reset (as per a system reset without the system clear pushbutton pressed). The control registers are initialized in the addressed CPU, and its PSW register, prefix register, CPU timer, and clock comparator are set to zero. The reset signal is not propagated to the other CPU.

Program Reset. The addressed CPU, its dedicated channels, and attached dedicated and shared I/O devices are reset (as per a system reset without the system clear pushbutton pressed). The reset signal is not propagated to the other CPU. This order can be used during recovery operations to reset a hung I/O condition.

Stop and Store Status. The addressed CPU enters the stopped state and the store status function is invoked. This order can be used to enable the functional CPU to attempt to obtain from the failing CPU the CPU timer, clock comparator, PSW, prefix, general register, and floating point register values, which are needed for a successful recovery.

Initial Microprogram Load. The addressed CPU, its channels, and its I/O devices are reset, after which system microcode is loaded into writable control storage from the disk cartridge device in the console unit.

Initial CPU Reset. An initial program reset, except for channel resetting, is performed in the addressed CPU. Pending channel interruptions are not reset. This function can be performed only via the SIGP instruction.

CPU Reset. A program reset, except for channel resetting, is performed in the addressed CPU. Pending channel interruptions are not reset. This function can be performed only via the SIGP instruction.

INTERPROCESSOR HARDWARE COMMUNICATION

Hardware-initiated communication between two CPUs in multiprocessor mode occurs as follows:

Malfunction Alert. Whenever a CPU loses power or a machine check error causes the CPU to enter the check stopped state, a malfunction alert indication is sent to the other CPU. A malfunction alert external interruption condition is generated in the receiving CPU if multiprocessor mode is in effect and the interruption occurs if the CPU is enabled for this condition (via bit 6 in control register 0 and current PSW bit 7). The address of the malfunctioning CPU is stored in processor storage locations 132 and 133.

Time-of-Day Clock Facilities. These functions have already been discussed. Briefly, hardware communication resulting from use of the time-of-day clock consists of (1) broadcasting a time-of-day clock security switch enable set condition in one CPU to the other CPU, (2) broadcasting the time-of-day clock oscillator pulse from one CPU (as selected via the configuration control panel) to the other CPU, (3) broadcasting a synchronization pulse between the two CPUs to check clock synchronization in the low-order bit positions (32 to 51), and (4) broadcasting a time-of-day clock synchronization check indication if a synchronization error occurs.

Buffer Intercommunication. The high-speed buffer storage controls in each CPU must communicate with each other to ensure that all real storage references by both CPUs result in access to the most current copy of the addressed data. In addition, changes to the storage protect keys must be provided to both CPUs. Therefore, there is a hardware interface between the two buffer controls by which each buffer control broadcasts to the other the real storage addresses of the data changed and the storage protect keys set.

In a two-CPU multiprocessing configuration, buffer contents are managed as follows. When a CPU makes a request for data that is not contained in its high-speed buffer, the data is fetched from real storage, placed in that CPU's buffer, and sent to the CPU, just as in a uniprocessor environment. The high-speed buffer in the other CPU is not affected. When a CPU, say CPU A, stores data in real storage, its buffer is updated as well if a copy of the data is currently being maintained in the buffer (as in a uniprocessor environment). In addition, however, the real storage address of the data stored is broadcast to CPU B.

CPU B then determines whether the contents of that storage location are currently being maintained in its buffer, and if they are, the data in the buffer in CPU B is marked invalid. This procedure ensures that the next fetch request for the data by CPU B will cause a real storage fetch and access to the most current version of the data.

Similarly, when a channel in either CPU stores data in real storage, the address of the data is broadcast to the other CPU so that each CPU can inspect its own buffer to determine whether the data is currently being maintained there. If so, the data is invalidated.

Both CPUs can fetch data from their respective buffers simultaneously. During uniprocessor and multiprocessor operations, a CPU can fetch eight bytes from its buffer in 160 nanoseconds (two CPU cycles). This is called a local request. During multiprocessor operations, however, if a local request for a buffer fetch occurs simultaneously with a remote request for marking buffer information invalid, the local request is held for 80 nanoseconds (one cycle) while a search for the data is made and another 80 nanoseconds if invalidation must be performed. Hence 240 or 320 nanoseconds (3 or 4 cycles) are required in this case.

Whenever a CPU issues a SET STORAGE KEY instruction, its TLB is inspected and any entries that have the same real address as the address specified in the SSK instruction are invalidated, as in a uniprocessor environment. In addition, the real address is sent to the other CPU so that appropriate entries in its TLB can be invalidated.

Interlock Mode for CS and CDS Instructions. When multiprocessor mode is in effect, an interlock mechanism for the COMPARE AND SWAP (CS) and COMPARE DOUBLE AND SWAP (CDS) instructions is activated. When a CPU issues either of these instructions, the logical storage involved is made to appear busy to the other CPU so that the other CPU cannot store into the location indicated in the CS or CDS instruction between the fetch and store operation the instruction requires. That is, the CPU that issued the CS or CDS instruction enters interlock mode.

If one CPU, say A, is in interlock mode and the other CPU (B) also enters interlock mode, CPU B is also prevented from performing any fetch operations until CPU A leaves interlock mode (completes its fetch and store operation).

Manual Controls. All manual controls on the system control panel on the 3066 function identically in uniprocessor and multiprocessor mode except for the time-of-day clock security switch, system reset pushbutton, load pushbutton, and system clear pushbutton, as discussed previously.

CHANNEL RECONFIGURATION HARDWARE

Channel reconfiguration hardware is contained in all multiprocessor models of the Model 168. It is a programmable facility that enables either CPU to control the operation of the channels normally dedicated to the other CPU. This hardware is designed to improve the programmed recovery capability provided by OS/VS2 multiprocessing support when a CPU fails and must be removed from the operational system.

Channel reconfiguration is accomplished by the issuing of a DIAGNOSE instruction in the CPU that is to activate the channel reconfiguration hardware. The maintenance control word (MCW) specified in the DIAGNOSE instruction has two bits that control the channel reconfiguration hardware. One bit controls the enabling and disabling of the channel reconfiguration hardware in the activating CPU. The other bit activates and deactivates the channel reconfiguration hardware for one specific channel in the other CPU.

Only one channel can be reconfigured to the activating CPU at a time. The address of the channel to be reconfigured is also specified in the MCW. Multiprocessor mode must be in effect in order for channel reconfiguration to be activated via a DIAGNOSE instruction.

When one CPU issues a DIAGNOSE instruction to activate channel reconfiguration for a channel in the other CPU, the channel control interface for the specified channel is diverted to channel 6 in the activating CPU. Only the major CPU-to-channel and channel-to-CPU control lines are diverted to the activating CPU. (See line from channel signal conversion to execution unit in Figure 20.15.3.) The data flow lines are not affected and therefore channel data rates are not changed when channel reconfiguration is activated for a channel.

When channel reconfiguration is activated, the activating CPU can issue I/O instructions and receive I/O interruptions from the reconfigured channel in the other CPU. Channel 6 must be used in I/O instructions to the configured channel and the channel identification of 6 is stored with any logout data for the reconfigured channel. A reconfigured channel uses the prefix value of the activating CPU for CSW storing, CAW storing, and normal channel logouts.

Channel reconfiguration for a channel is reset by issuing a DIAGNOSE instruction with the enable bit on and the activate bit off. Channel reconfiguration is also reset when a switch from multiprocessor to uniprocessor mode is made and when any type of reset, including power on reset, occurs.

During the time a channel is reconfigured to the activating CPU, channel 6 in the activating CPU can be involved in an I/O operation that was started before channel reconfiguration was activated. However, the activating CPU cannot receive any I/O interruptions or issue any I/O instructions to its own channel 6 until after channel reconfiguration is reset.

By activating and deactivating the channel reconfiguration hardware for one remote channel at a time, the activating CPU can control the I/O operations on all the channels in the other CPU as well on its own channel 6 and other channels. Note that the activating CPU can receive an I/O interruption from a given remote channel only during the time the remote channel is reconfigured to the activating CPU and the activating CPU is enabled for I/O interruptions from channel 6.

For most efficient system operation after a CPU failure occurs that causes channel reconfiguration hardware to be activated, all paths to symmetric I/O devices from the malfunctioning CPU should be varied off-line by the operator as soon as possible.

Use of channel reconfiguration hardware does not affect the I/O devices that can be attached to the Model 168 channels in a tightly coupled multiprocessing configuration. However, proper operation of time-dependent I/O devices cannot be assured during the periods of time channel reconfiguration hardware is active.

Certain components and portions of the CPU whose channels are to be reconfigured cannot be powered off while the channel reconfiguration hardware in the other CPU is active, since certain portions of this CPU's hardware are required by the channel reconfiguration hardware. Specifically, the three CPU frames (01, 03, and 04) in the 3168 Processing Unit, 3066 System Console, 3068 Multisystem Communication Unit, 3067 Power and Coolant Distribution Unit, and channel convert logic in the 3168 Processing Unit must be powered on. (That is, power must remain on in the units through which the data flow lines pass in order for channel reconfiguration hardware to operate.) The Integrated Storage Controls in the 3168 Processing Unit must also be powered on if devices attached to it are to operate under control of the channel reconfiguration hardware.

Channel reconfiguration can be performed by both CPUs in a multiprocessing configuration but should be activated by only one CPU at a time. This restriction must be controlled by programming, since there is no hardware check that prevents both CPUs from attempting to activate the facility simultaneously.

For correct logical operation of channel reconfiguration programming support, the CPU whose channels are to be reconfigured should be in the stopped state. Therefore, diagnostic programs cannot be performed in this CPU while channel reconfiguration hardware is activated by the other CPU. Note also that a CPU cannot activate channel reconfiguration hardware while the Dynamic Support System (DSS) is active and DSS cannot take control of the system while channel reconfiguration is active.

The support of channel reconfiguration hardware provided by the OS/VS2 multiprocessing control program offers the following recovery advantages:

- Error conditions are posted and I/O error recovery facilities are invoked only for those I/O operations that actually had an error on the failed CPU instead of for all I/O operations in progress at the time of the CPU failure.

- Recovery of I/O operations in progress on asymmetric devices attached to the failed CPU are handled in most instances (where time-dependencies are not involved) without the necessity of halting I/O operations to the device and requesting operator intervention to manually switch the device to the operative CPU.

- There is no need for the multiprocessing configuration to enter the wait state for operator action when a CPU fails in a configuration in which the tightly coupled multiprocessing configuration is sharing with another system direct access storage that can be updated.

60:15 RECORDING AND DIAGNOSTIC PROGRAMS

The same diagnostic programs (ST370, HDM, etc.), microdiagnostics consisting of approximately 25 programs (fault locating tests), and recording programs (RMS, SEREP, EREP, OLT, LOA, etc.) are provided for multiprocessor as for uniprocessor configurations and are updated as appropriate for multiprocessing hardware. Additional microdiagnostics are provided to test multiprocessing hardware.

When a malfunction occurs in a multiprocessor CPU, the appropriate nonresident microdiagnostics provided for uniprocessor models should be executed first. A minimum system consisting of the malfunctioning CPU, 1024K of processor storage, one channel and tape unit for loading the tests, and the 3066 System Console are required to execute these microdiagnostics and uniprocessor mode must be in effect. These components can be logically and physically removed from the multiprocessing configuration without quiescing system operations, as discussed previously. The other system can continue to perform production processing while microdiagnostics are being executed in the malfunctioning CPU.

If these microdiagnostics do not locate the malfunction, the MP Model Half Duplex Functional Test should be executed next. This requires the malfunctioning CPU plus all its installed processor storage. This test can be executed concurrently with production processing in the other system provided the other CPU has enough processor storage to operate.

The following microdiagnostics, which require both systems for their execution, are to be executed in the sequence listed to test tightly coupled operation of the multiprocessing configuration:

- MP Model Full Duplex Functional Test

- MP Model Hardware Functional Test

- MP Model Full Duplex Functional and System Test

- Channel Prefixing and Channel Reconfiguration Test

In order to replace a malfunctioning card in the 3068, the oscillator control switch must be set to the operational system or to local, and processor storage must not be cross-configured. Card replacement in a CPU requires that the malfunctioning system (that portion on which diagnostics were run) be powered down.

60:20 PLANNING CONSIDERATIONS

Successful installation and operation of a shared storage multiprocessing installation requires consideration of some factors that need not be considered in a uniprocessor environment or that are of less importance. Additional planning considerations are discussed in this subsection.

PLANNING FOR MAXIMUM SYSTEM AVAILABILITY

System availability is directly affected by the reliability of the components of the system (both hardware and programming) and by the serviceability of the system components. The reliability of a component is defined in terms of its frequency of outage (solid failures in a given time period) while serviceability is measured in terms of diagnostic facilities available and the duration of time required for repair.

The four components of a system that affect its net productive time are the hardware configuration, the environment in which the system operates, the operating personnel, and the operating system (control and processing programs). The following discusses the reliability and serviceability features of these components as they relate to a Model 168 shared storage multiprocessing configuration and the steps that can be taken by system designers to maximize the availability of such a system.

Hardware Configuration

The reliability of the hardware components of a Model 168 multiprocessing configuration (like that of a uniprocessor model 168 system) is enhanced by the use of an inherently more reliable technology than that used in System/360 and by the implementation of extensive hardware retry, programmed retry, automatic hardware deletion, programmed hardware removal, and programmed recovery procedures that prevent system outages because of intermittent or solid failures in many cases.

Hardware serviceability is improved by extensive error recording, of intermittent as well as solid errors, and by the availability of inline diagnostics, online diagnostics, and enhanced fault-locating microdiagnostics. In general, more diagnostic data and automated diagnostic routines (such as the Logout Analysis Program for the Model 168) are provided for System/370 than for System/360.

While system design cannot improve the inherent reliability of any hardware component, a shared storage multiprocessing system can be designed such that the failure of one or more hardware components has minimal impact on the critical subsystem. At the least, critical hardware components should be duplexed. Critical hardware components are those that make up the minimal critical subsystem--that portion of the total hardware configuration that is necessary for the performance of productive work.

For maximum availability, CPU features, channels, and I/O devices should be symmetrically configured on the two CPUs and channel-switching or string-switching features should be installed on control units, where available, to provide access to I/O devices from both CPUs. Control unit redundancy (cross-channel switching for tape units, 2844 Auxiliary Storage Control for the 2314, string switching for 3330-series, 3340, and 3350 disk storage, etc.) should be used for critical I/O devices to eliminate the loss of access to critical I/O devices because of a malfunctioning control unit.

System designers can plan to use standard and optional serviceability features that are provided, such as OLTEP. When system usage is being planned, adequate time should be allocated for preventive maintenance procedures. In a shared storage multiprocessing environment, maintenance can be performed on one CPU while the other CPU continues processing (assuming that the system doing productive work consists of the minimal critical subsystem).

Operating Environment

The hardware components of the system configuration have been designed to operate within certain environmental constraints of temperature, humidity, cleanliness, etc., as detailed in System/370 Installation Manual - Physical Planning (GC22-7004). Any unit or system installed in an environment that violates any of these constraints can be expected to experience increased error occurrence and unavailability. Hence system design should ensure that the required physical environment is provided and maintained to avoid outages caused by improper physical surroundings.

Operating Personnel

Knowledgeable operators are highly desirable in any data processing installation. However, in a high-availability environment, having well-trained operators is of even more significance because operator errors or lack of proper responses to the changing operational environment can severely impact system availability.

Although the operator has a single interface to both systems in a Model 168 multiprocessing configuration and less action is required for operation of this configuration, as compared to the operational requirements of two independent Model 168 uniprocessors, successful operation of a multiprocessing configuration requires a little more knowledge than is required to run one Model 168 uniprocessor configuration.

The major new operational procedures that the operator must learn are those associated with hardware reconfiguration, that is, use of the configuration control panel and the configurability commands and procedures associated with OS/VS2 multiprocessing support.

An installation-designed operator checklist form for the configuration unit is a highly desirable operator aid. These checklist forms, marked with the most frequently used configurations, should become a standard part of job setup instructions for the operator.

Operating System

The reliability of the control and processing programs that are included in the operating system affect the availability of the system. As discussed previously, higher reliability and availability than that provided by previous operating systems were major design objectives of OS/VS2 MVS. Processing program development procedures for the installation should include thorough processing program testing and use of available debugging and problem determination aids as part of the standard application development procedures. These procedures are even more important in the development of routines that interface with the control program.

Despite extensive program testing, however, it is not always possible to test every condition that can arise during the execution of a program. Hence, processing program errors will occur that can cause termination of system operations, particularly during the initial use of any program. Therefore, the use of restart (such as advanced checkpoint restart and warm start) procedures, should be preplanned so that system operation can be restarted as quickly as possible after an abnormal termination.

SECTION 65: DIFFERENCES BETWEEN THE MODEL 3 AND THE MODEL 1

Model 3 of the Model 168 differs from the Model 1 primarily in its faster internal performance and the improved serviceability and availability made possible by the service processor unit, which is standard in the Model 3.

A tightly coupled Model 168 multiprocessing configuration can include any combination of Model 3 and Model 1 systems. The same standard features are provided for both models except for the amount of high-speed buffer storage. The same optional features are available for the Model 3 (uniprocessor and multiprocessor models) as for the Model 1 except for the 16K Buffer Expansion feature, which is not available for the Model 3.

A Model 1 CPU (3168-1 Processing unit) can be field-converted to a Model 3 CPU (3168-3 Processing Unit). The standalone 3066 Model 2 System Console for the Model 1 is also used with the Model 3. It must have the field-installable 3168-3 attachment feature in order to be used with a Model 3. The 3067 Model 3 Power and Coolant Distribution Unit must be used in a Model 3 system configuration. A 3067 Model 2 can be field converted to a 3067 Model 3. A motor generator set is required for the Model 3 as for the Model 1. The same motor generator can be used with a Model 3 as with a Model 1.

The Model 3 has the same processor storage capacity as the Model 1 (from one megabyte to eight megabytes in one-megabyte increments). Processor storage models for the Model 3 are U31 to U38 for uniprocessor systems and M31 to M38 for multiprocessor systems.

The same standalone channels (2860, 2870, and 2880) and I/O devices attach to the Model 3 as to the Model 1. The 3165/3168 attachment is required on the standalone channels attached to a Model 3.

The model-dependent fixed storage locations are the same in the Model 3 as in the Model 1 (see Figure 20.10.3) except for minor differences in a few fields in the CPU extended logout area. For example, the buffer size installed bits for 8K and 16K are no longer used.

Both models are supported by the same IBM-supplied programming systems, that is, OS MFT and MVT Releases 21.6, 21.7, and 21.8, OS/VS1, OS/VS2 Releases 1 and up, and VM/370. OS MFT and MVT Releases 21.7 and 21.8, OS/VS1 Releases 1 and up, OS/VS2 Releases 1 and up, and VM/370 Releases 2 and up are modified to process the model-dependent logout area data for the Model 3 that differs slightly from that of the Model 1. Although the EREP program in Release 21.6 of OS MFT and MVT will not be modified, it can still be used on a Model 168 Model 3.

A program can determine whether it is operating on a Model 1 or a Model 3 by issuing the STORE CPU ID instruction. The version field byte (bits 0 to 7 in the doubleword stored) indicate the model of the 3168 processor being used.

65:05 PERFORMANCE ENHANCEMENTS

The internal performance of the Model 3 Model 168 CPU is generally in the range of 5 to 13 percent faster than that of the Model 1 CPU (using a 16K buffer) when the same hardware configurations, programs, and programming systems that do not use 2K pages are used. The increase in

Model 3 internal performance will be less for users of VS1 since it supports a 2K page size.

The faster internal performance of the Model 3 is the result of the following differences between the Model 3 and the Model 1:

- 32K of high-speed monolithic buffer storage is standard for the Model 3. The 32K capacity is not utilized when a page size of 2K is being used. The buffer operates at a capacity of 16K in this situation (see buffer discussion below). Buffer fetch times, the way in which the buffer is used, and the buffer assignment algorithm are the same in both models. The 32K buffer and processor storage contain 128 columns, as shown in Figure 65.05.1, instead of 64 columns as in the Model 1. A column in the Model 3 buffer contains eight 32-byte blocks, as does a column in the buffer in the Model 1.

- The execution time of each of the following instructions is improved: SUPERVISOR CALL (SVC), MONITOR CALL (MC), STORE THEN OR SYSTEM MASK (STOSM), STORE THEN AND SYSTEM MASK (STNSM), INSERT STORAGE KEY (ISK), INSERT PSW KEY (IPK), SET PSW KEY FROM ADDRESS (SPKA), LOAD PSW (LPSW), SET SYSTEM MASK (SSM), STORE CLOCK (STCK), and SET PROGRAM MASK (SPM). In addition, under certain conditions execution of the following instructions is faster in the Model 3: OR CHARACTERS (OC), AND CHARACTERS (NC), EXCLUSIVE OR CHARACTERS (XC), TEST AND SET (TS), COMPARE LOGICAL CHARACTERS UNDER MASK (CLM), INSERT CHARACTERS UNDER MASK (ICM), STORE CHARACTERS UNDER MASK (STCM), COMPARE AND SWAP (CS), and COMPARE DOUBLE AND SWAP (CDS). These instructions are more heavily used by the virtual storage programming systems.

- Improved execution time for all levels of interruption

Improvements in the execution time of the instructions listed above and all interruptions are made possible by the increase in the size of writable control storage in the Model 3. The Model 3 has 1024K instead of 512K words of writable control storage. A denser technology is used for the implementation of writable control storage in the Model 3 so that less space is required for 1024K words in the Model 3 than for 512K words in the Model 1.

The 32K high-speed buffer in the Model 3 is also implemented in a denser technology than is used for the high-speed buffers in the Model 1 and it requires less space than a 16K buffer. The 32K buffer operates at its 32K capacity when the Model 3 CPU is operating with dynamic address translation mode disabled or with dynamic address translation mode and a 4K page size enabled. When dynamic address translation and a 2K page size are enabled, the 32K buffer operates at a 16K capacity just like the 16K buffer in the Model 1.

The reason for using a 16K capacity when a 2K page size is enabled is the following. Bits 20 to 26 of the referenced processor (real) storage address are required to determine the column address (0 to 127) in the buffer address array for a 32K buffer size. For a 16K buffer size, processor storage address bits 21 to 26 are required to determine the column address (0 to 63).

When a 4K page size is used, bits 20 to 31 of the referenced virtual storage address are the same as bits 20 to 31 of the corresponding real storage address and do not need to be translated. However, when a 2K page size is used, bit 20 must be translated as only bits 21 to 31 in the virtual and corresponding real storage addresses are equal.

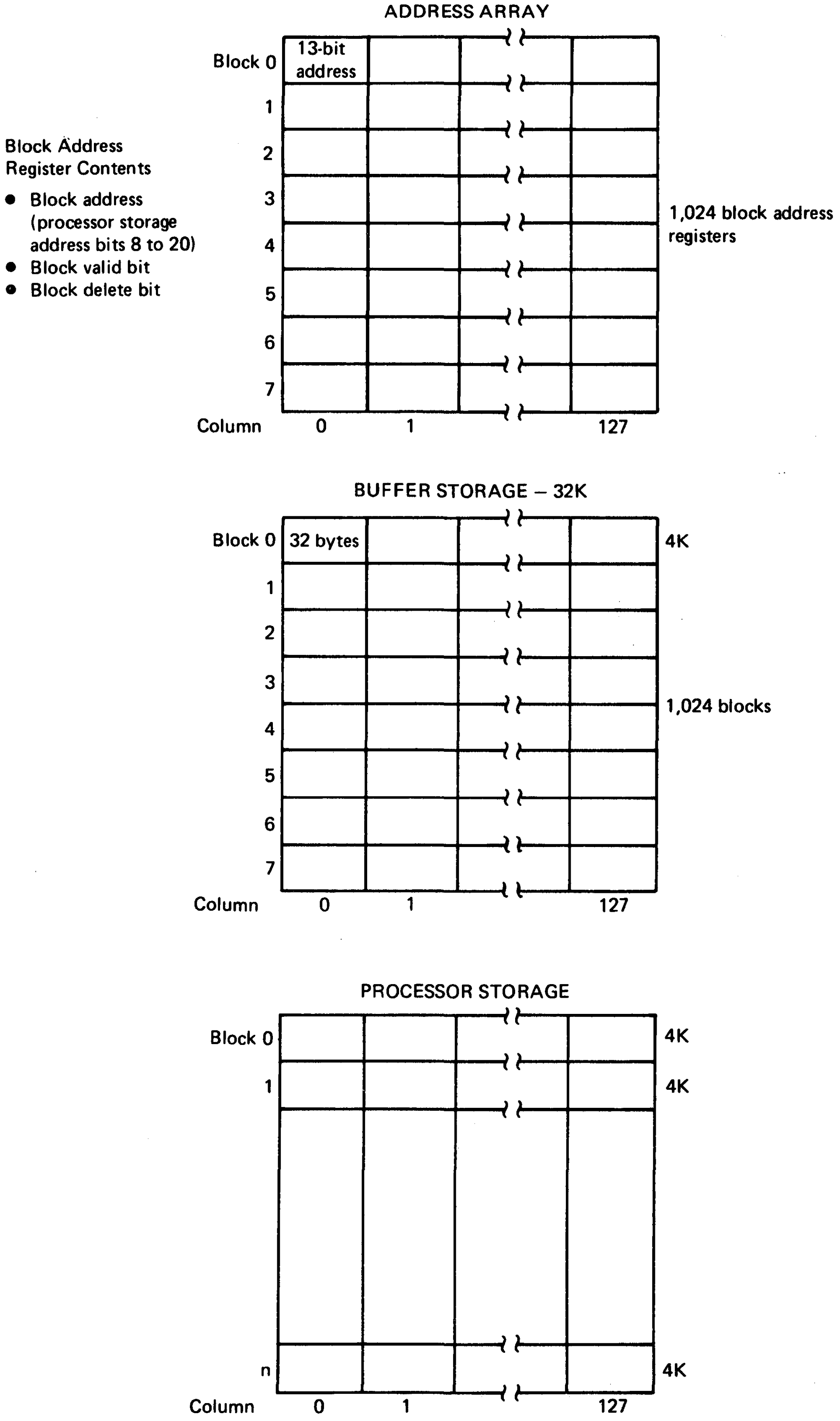


Figure 65.05.1. High-speed buffer and processor storage organization in the Model 3

Therefore, if the 32K capacity were to be used for a 2K page size, bit 20 would not be available for buffer address array addressing until after address translation had been performed. By using a 16K capacity for a 2K page size, bits 21 to 26 are available for accessing the buffer address array before address translation is performed.

Whenever the buffer in the Model 3 is reset, it is set to operate at its 32K capacity. The buffer is reset when one of the following occurs:

- IPL, program reset, power on reset, or system clear
- CPU reset caused by any condition except pushing the computer reset pushbutton
- All console loads except load microdiagnostics
- Enter reconfiguration pushbutton is pressed
- SIGNAL PROCESSOR (multiprocessing) instruction is issued that specifies one of the orders that is defined to cause a buffer reset (IPL, program reset, IMPL, initial CPU reset, or CPU reset)
- Load control instruction is issued to change the page size in effect. When page size is changed from 4K to 2K, buffer capacity is reduced to 16K. When page size is changed from 2K to 4K, buffer capacity is increased to 32K.

65:10 THE SERVICE PROCESSOR

FUNCTIONS AND GENERAL OPERATION

The service processor is standard in the Model 168 Model 3. It is a hardware unit that is contained in the Model 168 CPU (3168-3 Processor Unit), but that is functionally separate from the CPU. Its function is to provide greater maintenance capabilities for the Model 3 than are provided by the optional 2955 Remote Analysis Unit for the Model 1 (which is not available for the Model 3). The service processor supports an interface to RETAIN/370 that offers the same functions as the 2955 interface to RETAIN/370 in the Model 1 as well as enhancements that improve the remote problem analysis capability for the Model 3.

The service processor provides the capability of continuously monitoring selected logic points in the Model 168 CPU, capturing and storing status data when an intermittent or solid hardware error occurs (or at other specified times) for later use by a customer engineer, producing a printout of the stored data for use by a local customer engineer, and transmitting the stored data over a communication line for remote analysis by a customer engineer specialist in the Large System Support Group in Poughkeepsie.

The capabilities of the service processor are designed to make more timely information available for both on-site and remote customer engineer analysis so that fault location, particularly for intermittent errors, which are frequently difficult to duplicate, can be accomplished more quickly.

The components of the service processor are the processor unit, trace unit, two battery-powered counters, internal disk file, CE panel, printer control for an optional 3213 Printer, corporate standard interface, and modem, as shown in Figure 65.10.1.

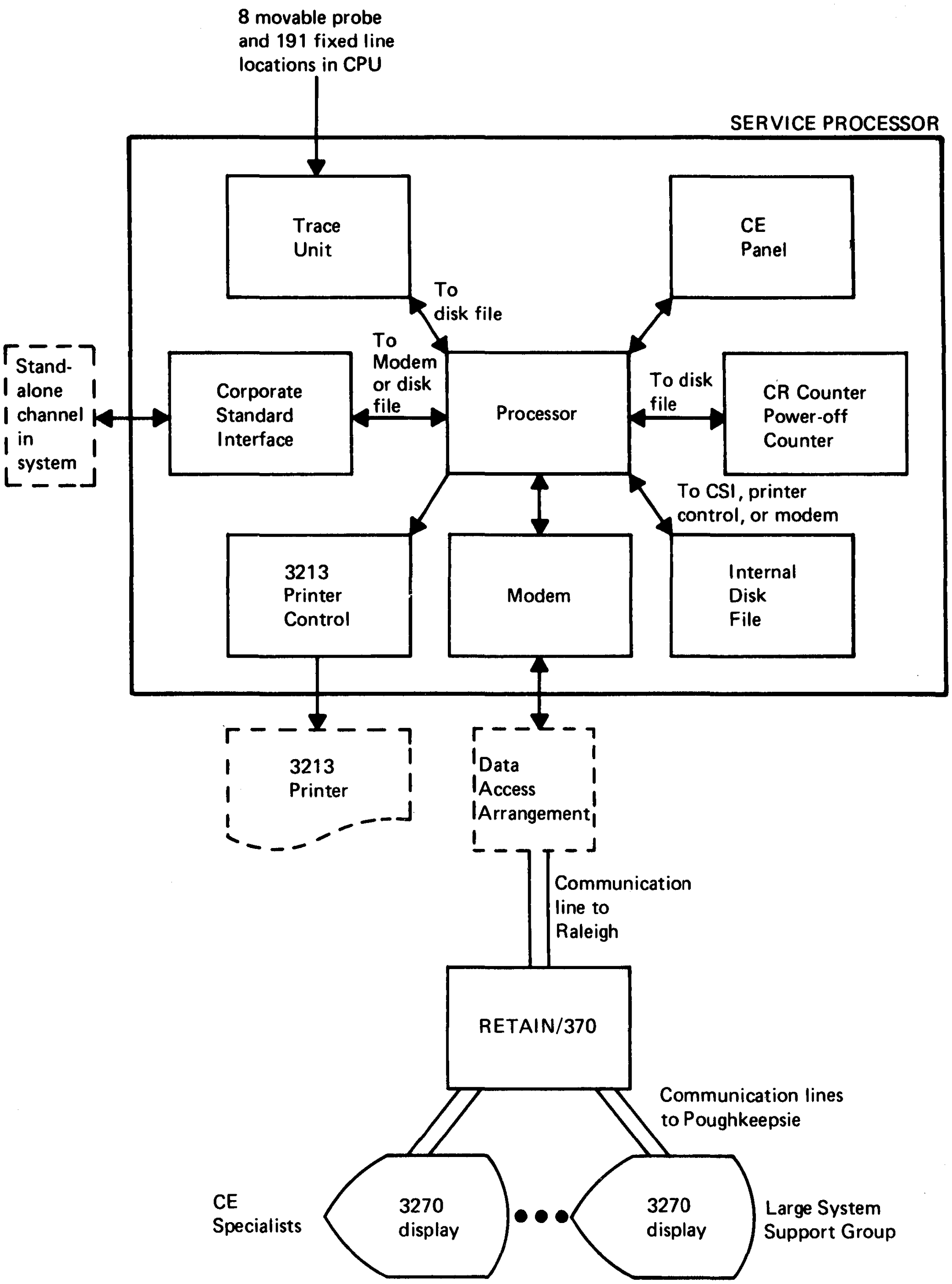

8 movable probe and 191 fixed line locations in CPU
SERVICE PROCESSOR
Trace Unit
CE Panel
To disk file
To Modem or disk file
Stand-alone channel in system
Corporate Standard Interface
Processor
To disk file
CR Counter Power-off Counter
To CSI, printer control, or modem
3213 Printer Control
Modem
Internal Disk File
3213 Printer
Data Access Arrangement
Communication line to Raleigh
RETAIN/370
Communication lines to Poughkeepsie
CE Specialists
3270 display
3270 display
Large System Support Group

Figure 65.10.1. Components of the service processor in a Model 3

The service processor operates under the control of the stored-program-controlled processor unit. The service processor normally operates simultaneously with the operation of the Model 168 CPU to capture status data and record it on the internal disk file component, usually when a hardware failure occurs. The customer engineer controls the status data collected by setting switches on the CE panel component. When operating in this manner (recording mode), the service processor never steals any machine cycles from the Model 168 CPU.

Using the CE panel on the service processor, the customer engineer can also operate the service processor independently from the Model 168 CPU. In this mode, the customer engineer can print the status data stored on the internal disk file on the 3213 printer or transmit the status data to a remote location for analysis. The entire service processor except for the trace unit is powered independently from the Model 168 CPU. Therefore, the service processor can perform functions, other than status data collection, under control of the CE panel in the service processor even when the Model 168 CPU is not operating or is powered down (for a maintenance operation, for example).

The way in which the service processor is powered on and off is determined by the setting of a switch on the CE panel of the service processor. If the switch is set to remote, the service processor is automatically powered on or off when the Model 168 CPU is powered on or off, respectively. When the switch is set to local, the power on and power off pushbuttons on the CE panel are used to power the service processor on and off.

When power is turned on in the service processor, an IMPL of diagnostic routines is automatically initiated. Pressing the SVP IMPL pushbutton on the CE panel also causes these diagnostics to be loaded. Once loaded, the IMPL diagnostics exercise the service processor to determine whether it is functioning correctly. If so, the service processor automatically goes into the recording mode of operation. In this mode, the trace unit of the service processor collects data from certain locations in the Model 168 CPU during its operation.

Operation of the service processor in the Model 168 is always controlled by the local customer engineer. If a customer engineer in the Large System Support Group in Poughkeepsie wishes additional history data or wishes to have a function performed on the Model 168, he communicates the request to the local customer engineer via telephone and the local customer performs the function.

One communication line is used for both the transmission of data to Poughkeepsie via RETAIN/370 (using data mode) and voice communication between the local customer engineer and a specialist in Poughkeepsie (using voice mode). When either customer engineer wishes voice communication with the other, he sounds an alarm and then goes from data to voice mode.

If an error occurs in the service processor during its operation, a switch on the CE panel is inspected by the processor unit to determine the action to be taken. If the error switch is set to the stop position, operation of the service processor terminates. If the switch is not set to the stop position, an IMPL is initiated. The diagnostic routines then determine whether the service processor can continue operating based on the type of error condition that exists.

PROCESSOR UNIT

The processor unit contains an arithmetic/logic unit, read-only control storage, data registers, and a main storage. It operates under the control of a program whose instructions are similar in format and mnemonics to System/370 instructions.

The basic functions of the processor unit are to (1) take data from the trace unit buffers and write it to the internal disk file and (2) read data from the internal disk file and transfer it to the printer control, corporate standard interface (CSI), or modem component of the service processor. The processor unit can also transfer data from the corporate standard interface to the modem and transfer data to the internal disk file from the corporate standard interface. The processor unit can perform only one of its functions at a time.

The control program routines for the processor unit are contained on the internal disk file in the service processor. Some of these routines are always resident in main storage of the processor unit during its operation. The resident routines are loaded into the processor unit after a successful execution of the IMPL diagnostics. Other control program routines are brought into the processor unit only when they are required to service a request.

The basic control routine is a polling loop. This routine constantly interrogates each of the other components of the service processor (trace unit, corporate standard interface, etc.) on a rotating basis to determine whether the component has an outstanding request. When a request is recognized, polling stops and the appropriate control routine is loaded into the processor unit to service the request. Polling continues as soon as the request has been processed.

TRACE UNIT

The trace unit receives certain status data from the Model 168 CPU while the latter is operating, stores the data in trace buffers, and when a predetermined event occurs presents the trace buffer data to the processor unit for storing on the internal disk file.

The trace unit obtains the following data:

- Information from 191 fixed lines to points in the Model 168 CPU. Every machine cycle (80ns), the data from these 191 fixed points is placed in a trace buffer. The buffer has a maximum capacity of 32 machine cycles of data. A wraparound technique is used to store data in the trace buffer so that the buffer always contains information regarding the last 32 machine cycles.

- Information from eight movable probe points in the Model 168 CPU that the customer engineer can establish. Every ten nanoseconds, information from these probe points is placed in a second trace buffer. The capacity of this buffer is 256 ten-nanosecond cycles. A wraparound technique is also used to store data in this buffer.

- Up to 224 doublewords of logout data from the Model 168 CPU. This is the data logged in the logout area in lowest addressed processor storage when a CPU logout occurs.

Information from the 191 fixed points and eight probe points continues to be stored in the trace buffers in wraparound fashion until a predetermined event occurs. When the event is recognized, recording stops temporarily and the fixed point and probe data in the trace buffers is sent to the processor unit. The appropriate control program routine then formats the data, time stamps it, and writes it to the internal disk file. Trace unit recording resumes as soon as the trace data has been transferred to the processor.

The event that is to cause existing trace data to be written is indicated via the CE panel and can be one of the following: a machine check interruption (for any enabled soft or hard machine check condition), main storage address compare, instruction counter address compare, control storage address compare, hang-detect, SIGNAL PROCESSOR instruction from the other CPU in a tightly coupled multiprocessing configuration, or a logic line input that can be wired to any point in the CPU or a fix card.

When the predetermined event occurs, an interval of approximately 655 microseconds is established. If a CPU logout occurs before the interval expires, it is assumed to be associated with the event that caused this recording to take place. The CPU logout data in the trace buffer is formatted, time-stamped, and recorded on the internal disk file along with the fixed point and probe data. If a CPU logout does not occur within the interval, logout data is not written to the internal disk file.

The CPU logout data is divided into three areas for the purpose of recording: status area (corresponding to the fixed logout area from processor storage locations 0 to 184), local store area (corresponding to the fixed logout area between locations 216 and 511), and the CPU area (corresponding to the model-dependent CPU extended logout area beginning at the location indicated in control register 15). The CPU area is further subdivided into subareas so that the printing of CPU logout data can be done on a selective basis by area and by subareas within the CPU area recordings.

COUNTERS

The CR (continuously running) counter is a battery-powered counter that is always running to maintain the time of day. When power is on in the Model 168 CPU, the CR counter runs from this power. When CPU power is off, the CR counter runs from the power supplied by its battery. The time in the CR counter is synchronized with the time in the time-of-day clock in the Model 168 CPU whenever a SET CLOCK instruction is issued that sets the time-of-day clock. The CR counter is used to time stamp trace data records that are written to the internal disk file.

The power-off counter is also a battery-powered counter. It runs only when power is off in the Model 168 CPU. This counter starts to run when CPU power is turned off and stops running when CPU power is turned on. It can be used to keep account of how long CPU power is turned off.

INTERNAL DISK FILE

The internal disk file used in the service processor is the same file that is used to load microprograms in Models 158 and 168. The disk file is used to hold the control programs required by the processor unit and for storage of trace data records.

The disk file can contain a maximum of 16 trace data records. When this maximum is reached, the action taken depends on the mode set via the CE panel. If wrap mode is in effect, each successive trace record replaces the oldest existing trace record so that the file contains only the last 16 trace records. If wrap mode is not in effect, the existing trace records are not overwritten and tracing operations terminate.

The local customer engineer can clear the internal disk file using a toggle switch on the CE panel of the service processor. All existing or selected event class records can be cleared. Clearing consists of zeroing the header records that are associated with the existing data records. The data records themselves are not zeroed.

CE PANEL

The CE panel in the service processor enables the customer engineer to (1) establish operating conditions for the trace unit, as discussed previously, (2) transfer data records from the internal disk file to the modem or 3213 printer control component of the service processor, (3) clear the internal disk file, and (4) execute microdiagnostics to test the service processor for correct operation. When the trace unit presents data to the processor component, the data is always written to the internal disk file. The data cannot be transferred directly to the modem or printer control component.

The customer engineer controls the transfer of status records from the internal disk file using a set of disk data control toggle switches or an OLT (online test) routine. The customer engineer uses a set of diagnostic selection switches to control the service processor diagnostic routine to be executed. When the customer engineer wishes to transfer status records or execute a specific diagnostic routine, he turns on the appropriate toggle switch and pushes the execute button on the CE panel. The function indicated by the toggle switch is then performed.

Separate destination toggle switches are provided for trace records. Trace records can be directed to the printer control component for printing on the 3213 Printer or to the modem component for transmission to RETAIN/370 in Raleigh.

A set of printer control toggle switches is also used when trace data records are to be transferred to the printer control component. These toggle switches are used to select the types of trace data records that are to be printed: fixed line status data, movable probe status data, header, or CPU logout records. Status area and local store area and/or CPU area records can be selected for printing when the CPU logout toggle switch is turned on. When CPU area records are selected, two microfiche selection switches are also used to indicate the subarea of the CPU area whose records are to be printed.

In order to transfer trace records from the internal disk file to the modem component, the modem must be enabled. Activation of the modem can be accomplished only by inserting the CE key (same CE key as is used for the Model 168 CPU) in the activate TP slot in the CE panel in the service processor. Once the modem is activated, the TP active/key reset pushbutton lights up and the CE key can be removed. The modem is deactivated by pressing the TP active/key reset pushbutton.

The local customer engineer can perform <u>one</u> of the following operations involving the service processor at the same time normal system operations are taking place in the Model 168 system:

- Print trace data from the internal disk file on the 3213 Printer. This is controlled via the CE panel.

- Send trace data from the internal disk file to RETAIN/370 in Raleigh. This is controlled via the CE panel.

- Print trace data from the internal disk file on a printer attached to the channel to which the corporate standard interface component of the service processor is connected. This is accomplished by executing an OLT under OLTEP control.

PRINTER CONTROL

The printer control component is provided to enable a 3213 Printer to be attached directly to the service processor. Attachment of this printer is optional. Using the CE panel, the customer engineer can cause all or selected trace data records from the internal disk file to be printed on the 3213. Operation of the 3213 printer is controlled entirely by the CE panel and is independent from operation of the Model 168 CPU.

CORPORATE STANDARD INTERFACE

The corporate standard interface (CSI) is provided to connect the processor unit of the service processor to any one System/370 or System/360 channel. Normally it would be connected to a 2870 Multiplexer Channel in the Model 168 configuration. A switch on the CE panel is used to enable or disable the connection between the CSI and the processor unit.

This interface can be used by the customer engineer, for example, to print trace data records on a local system printer that is faster than the 3213, such as a 3211 or 1403. The processor unit in the service processor can also receive input from the CSI. Using this capability, a program running in the Model 168 CPU (such as an OLT) can send data to the modem component for transmission to a remote location or read data from the internal disk file.

An OLT that runs under OLTEP or OLTSEP is provided that reads trace records (status and CPU logout) from the internal disk file via the CSI, formats the data, and writes it to an output device (usually a printer). Another OLT is provided that reads only the CPU logout trace records from the internal disk file (via the CSI). This OLT then invokes the Logout Analysis Program to operate on the CPU logout data and print the results on a local printer via the CSI.

MODEM

The modem component provides the means of connecting the service processor to RETAIN/370 in Raleigh via a data access arrangement and communication line for the purpose of remote problem analysis. The modem has two modes of operation, the remote program mode and the teleprocessing link mode.

The remote program mode enables the service processor to perform the same functions for the Model 3 as can be performed for the Model 1 using the 2955 remote analysis unit. The same OLTs that are used with the 2955 can be used with the modem in remote program mode. That is, an OLT running under OLTEP concurrent with normal system operations (or under OLTSEP in a standalone environment) can send SYS1.LOGREC data to the RETAIN/370 system in Raleigh, after which the modem connection can be disabled.

When the modem is operating in teleprocessing link mode, the local customer engineer can transmit trace records contained on the internal disk file to the RETAIN/370 system using the CE panel on the service processor, as discussed previously. This data is then transmitted to a specialist in the Large System Support Group in Poughkeepsie or another technical support group connected to RETAIN/370. The specialist interfaces with RETAIN/370 using a 3270 display station.

If the specialist requires any additional information or history data from the Model 168, he requests it from the local customer engineer via telephone. Similarly, once the specialist has analyzed the problem, he communicates the information to the local customer engineer via telephone.

The Model 3 offers the following improvements in the remote analysis capability when compared with that provided for a Model 1:

- Additional and more timely status data is made available to the specialist, as provided by the trace unit of the service processor.

- The specialist in Poughkeepsie uses a 3270 display unit to communicate with the RETAIN/370 system in Raleigh. This enables the specialist to see much more history data displayed concurrently than does the display device used in a Model 1 environment.

- The analysis and data reduction capabilities of the programs that operate on the history data sent to RETAIN/370 have been enhanced and enable the specialist to be more selective in his requests for data.

The features listed above are designed to enable a customer engineer specialist in the Large System Support Center to diagnose failures in a Model 168 more frequently without the need to go to the installation itself. In addition, status data about intermittent errors can be analyzed by the specialist concurrently with normal system operations.

As in a Model 1 environment, a 2955 OLT operating under OLTSEP can control the operation of the HDM Diagnostic Program in the Model 168 Model 3 CPU. A service processor maintenance program is also provided that operates under the control of the HDM Diagnostic Program. This program runs diagnostics that test the operation of the lines between the Model 168 CPU and the service processor.

ADVANTAGES

The advantages of the service processor in the Model 3 Model 168 are the following:

- More detailed information about intermittent and recoverable errors is provided than for the Model 1 and on a realtime basis. The customer engineer can obtain this information and perform problem analysis concurrently with normal system operations.

- The need to try to re-create intermittent failures for analysis by the customer engineer is reduced.

- More detailed information about solid errors is provided than for the Model 1. This data can be analyzed by the local customer engineer or sent via a communication line to the Large Systems Support Group for analysis by customer engineers with more expertise.

- Customer engineer operation of the service processor is controlled by a separate CE panel in the unit instead of by the operator console so that problem analysis operations can be performed concurrently with normal system operations.

- The service processor is physically independent of the Model 168 CPU so that no processing time is taken from the Model 168 CPU and an error in the service processor does not impact the Model 168 CPU. Similarly, the service processor can operate when the Model 168 CPU is down for maintenance.

The detailed, timely status information provided about errors, remote analysis capability, and concurrent problem analysis capabilities provided by the service processor should result in a reduction in the number of times normal system operation is interrupted for intermittent error analysis and the amount of time the system is not operational for the purpose of locating the cause of solid failures. Since more data about failures is provided by the service processor, faster error analysis should occur even if the remote analysis capability is not utilized.

SECTION 67: ATTACHED PROCESSOR SYSTEM

67:05 SYSTEM DESCRIPTION

The Model 168 Attached Processor (AP) System offers uniprocessor Model 168 users increased internal performance. The AP System is a Model 168 system configuration in which an additional instruction execution capability is provided. An AP System contains two tightly coupled processor units (a Model 168 Model A3 CPU and a 3062 Attached Processing Unit) that share all processor storage in the Model A3 CPU and execute under the control of a single multiprocessing operating system.

The 3062 Attached Processing Unit (APU) is an instruction processor that is capable of executing all but two of the same instructions as the Model A3 CPU. Hence, a Model 168 AP System can execute two instruction streams simultaneously, as does a Model 168 tightly coupled multiprocessing configuration. The AP System is supported by OS/VS2 MVS, as of Release 3.7, and TSS/370.

The AP System is a growth system for Model 168 uniprocessor users who require additional internal performance. The internal performance improvement realized when a uniprocessor Model 168 system is upgraded to a Model 168 AP System is dependent on the amount of multitasking that can be achieved. The Model 168 AP System is capable of providing internal performance 1.5 to 1.8 times that of a Model 168 Model 3 uniprocessor system, depending on the amount of multitasking in the job streams involved.

The Model 168 AP System also offers better price performance than two Model 168 uniprocessor systems and provides the operator with a single system image. That is, the operator has one operational interface to the entire AP configuration and one job scheduling interface. The AP System also eliminates the need to use Shared DASD support, which is required in a configuration with two uncoupled systems in order to share a data base. Shared DASD support reduces throughput because of the interference it causes.

The components of the AP System are shown in Figure 67.05.1. It contains the following units:

- A Model 168 Model A3 CPU (3168-A3 Processing Unit). This unit is a Model 3 CPU with tightly coupled multiprocessing hardware functions and the hardware modifications required to tightly couple it with the 3062 APU. A Model 3 CPU can be field-converted to a Model A3 CPU. Once this change is made, a Model A3 CPU cannot operate until the 3062 APU is installed.

  All standard and optional features available for the Model 3 CPU (3168-3 Processing Unit) except multiprocessing are also available for the Model A3 CPU. The same four-way, doubleword interleaved processor storage (from 1024K minimum to 8192K maximum in 1024K-byte increments) is available for CPU Models A3 and 3 and is designated as Models A31 to A38 for the Model A3.

- A standalone 3067 Model 3 Power and Coolant Distribution Unit for the Model A3 CPU. This is the same unit as is used with a Model 3 CPU with an AP System feature installed.

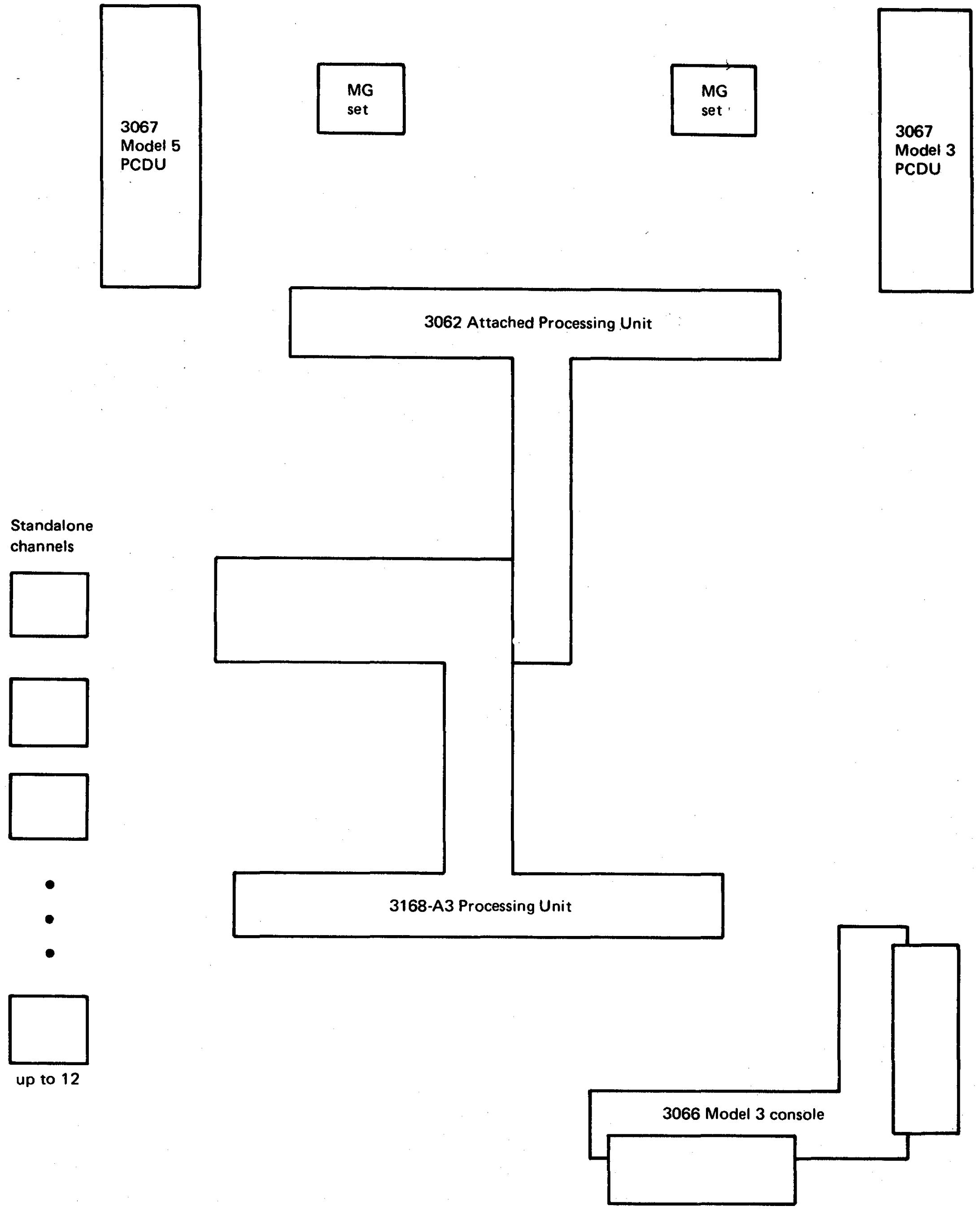


Figure 67.05.1. Components of the Model 168 Attached Processor System

- Standalone 2860, 2870, and 2880 channels (up to twelve maximum in the same combinations as supported for the Model 3 CPU) attached only to the Model A3 CPU. The same I/O devices can be attached to the channels of a Model A3 CPU as to those of a Model 3 CPU. No channels can be attached to the 3062 APU. The maximum I/O configuration is the same for the AP System as for a uniprocessor Model 1 or 3 configuration and the aggregate data rate is approximately the same (see Section 20:20).

- A standalone 3066 Model 3 System Console to control both instruction processor units. This unit is a 3066 Model 2 console with an additional panel containing controls for the 3062 APU. A 3066 Model 2 can be field-converted to a 3066 Model 3.

- A motor generator set for the Model A3 CPU. The same motor generator set that is used with a Model 3 CPU can be used with a Model A3 CPU.

- A 3062 Model 1 Attached Processing Unit. This unit contains an instruction execution unit similar in capability to that in the Model A3 CPU. Details about this unit are discussed in Section 67:10.

- A 3067 Model 5 Power and Coolant Distribution Unit for the 3062 APU, which has lesser power and cooling requirements than the Model A3 CPU

- A motor generator set for the 3062 APU

Note tnat RPQs are available that enable a Model 168 Model 1 to be field-converted to a Model 168 AP System. When the 3168-1 Performance RPQ is installed on a 3168-1 Processing Unit, the 3168-1 is functionally equivalent to a 3168-3 Processing Unit except for the absence of the service processor unit. The 3168-1 AP Attachment RPQ (which requires the 3168-1 Performance RPQ as a prerequisite) and the 3062 APU can then be installed to complete the conversion to a Model 168 AP System.

Like a Model 3 CPU, the Model A3 CPU and 3062 APU each have an 80-nanosecond cycle time, local storage with an 80-nanosecond cycle time, and read-only and writable control storage with an 80-nanosecond cycle time. (Writable control storage in both the Model A3 CPU and 3062 APU is loaded from the disk file in the 3066 console.) The Model A3 CPU or 3062 APU can fetch eight bytes of data from its own high-speed buffer in 160 nanoseconds, as can the Model 3 CPU.

Since processor storage is four-way interleaved in the Model A3 CPU, the total processor storage present in this unit is divided into four logical storages, each of which can be accessed simultaneously. Nonsimultaneous requests from the two processors (Model A3 CPU and 3062 APU) to a nonbusy logical storage are handled on a first-come, first-served basis. When both processors simultaneously request access to the same logical storage, a priority scheme is used to determine which processor is given access.

First, the highest priority contender within each processor is established. This determination is made concurrently in the two processors. Within the Model A3 CPU, priority is resolved among requests from channel buffers and among requests from three CPU registers. These two priorities are resolved at the same time. Then the highest priority channel buffer is given priority over the highest priority CPU register in the Model A3 CPU. In the 3062 APU, priority is resolved among the three CPU registers.

After the highest priority request in each processor is determined, priority is resolved between the two requests according to the rule that each processor has the capability of selecting a given logical storage on an alternate (every other) storage (80-nanosecond) cycle basis. That is, one processor can access a given logical storage every odd storage cycle while the other can access the same given logical storage every even storage cycle.

Note that the 3168 CPU is permitted to have two or more successive accesses to the same logical storage even though the 3062 APU may have a request outstanding for the same logical storage. Thus, for example, if the CPU is given access to processor storage and accesses four consecutive logical storages beginning with, say, logical storage 1, and has another request outstanding for logical storage 1, the CPU is given access to logical storage 1 even though the APU may also have a request outstanding for logical storage 1 (since the current cycle is still the alternate one during which the CPU can select logical storage 1).

The APU is permitted to have two or more successive requests to the same logical storage, as described for the CPU, unless the highest priority request in the CPU is a channel request for the same logical storage as the APU is to access. In this case, the channel is given priority over the APU for access to the logical storage involved. If the channel then accesses four consecutive logical storages, beginning with logical storage 0, for example, and the CPU has a request outstanding for logical storage 0, the CPU is given access to logical storage 0 even though the APU may have a request for logical storage 0 outstanding, since the current cycle is still the alternate one during which the CPU can select logical storage 0.

Processor storage access time for the AP System (that is, the time from PSCF selection to availability of the data in the PSCF) is 480 nanoseconds. A fetch of eight bytes from processor storage by either the Model 3 CPU or 3062 APU (the time from request acceptance to availability of the data in a processor register) is 720 or 800 nanoseconds assuming the logical storage required is not busy. If the fetching processor has priority for selecting storage, 720 nanoseconds are required; otherwise, 800 nanoseconds are required. These two times increase if the logical storage is busy when required.

The power on and power off pushbuttons on the operator control panel of the 3066 console cause power to be turned on in both the Model A3 CPU and 3062 APU if the local/remote switch on one of the frames of the 3062 is in the remote position. When this switch is in the local position, the 3062 APU is powered on and off separately from the Model A3 CPU, using the power on and off pushbuttons on the 3062 frame. Power is removed from all components of the AP System (Model A3 CPU, 3062 APU, channels, etc.) when the emergency power off switch on the 3066 is pulled.

A power on causes writable control storage in both the Model A3 CPU and the 3062 APU to be loaded. An IMPL is performed in the 3062 APU first, and if it is successful, an IMPL is then performed in the Model A3 CPU.

Both the Model A3 CPU and the 3062 APU must be powered on in order for the AP System to operate. If a hardware failure in the 3062 APU prevents its operation, the AP System can still function with only the Model A3 CPU operative, as long as power remains on in the 3062 APU. However, interruptions from the 3062 APU may interfere with normal system operations, depending on the type of malfunction in the 3062 APU. Diagnostics cannot be executed on the malfunctioning 3062 APU until normal system operations involving the Model A3 CPU are terminated.

When a hardware failure prevents the Model A3 CPU from operating, the entire AP System cannot function unless the no-charge APS Maintenance Switch RPQ is installed. When this RPQ is installed and the 3168 processor has an uncorrectable hardware failure, the maintenance switch on the panel added to the 3066 console for AP Systems can be set to the APU position and a system reset performed to enable I/O instructions to be executed by the 3062 APU. A re-IPL can then be performed to continue system operation in a degraded mode until such time as the system can be made available for maintenance.

Note that certain types of hardware failures in the 3168 processor can prevent successful system operation using only the APU (in which case maintenance should be performed as soon as possible), programs that require features not available for the APU cannot be run, and the system activity meter does not function correctly with respect to I/O operations. The APS Maintenance Switch RPQ cannot be installed on an AP System that has a 3850 Mass Storage System attached.

Installations with a Model 168 AP System installed that require the additional channels, additional processor storage, and/or availability advantages provided by a Model 168 tightly coupled multiprocessing configuration (see Section 60:05) can field-convert an AP System to a tightly coupled multiprocessing system. This involves removing the 3062 APU (which cannot be converted to a 3168 Processing Unit), adding a 3068 Multisystem Communication Unit, adding another 3168 Processing Unit (either Model 1 or 3) with the multiprocessing feature, and replacing portions of the multiprocessing function hardware in the existing Model A3 CPU with Model 168 multiprocessing feature hardware.

Note that two Model 168 AP Systems cannot be coupled to form a tightly coupled Model 168 multiprocessing configuration. However, AP Systems can be included in the loosely coupled multiprocessing configurations supported by JES2 Multi Access Spool and JES3 support in OS/VS2 MVS.

Operating systems other than OS/VS2 MVS and TSS/370 that support the Model 168 can operate in the Model A3 CPU but do not support the 3062 APU (that is, a tightly coupled multiprocessing environment). The performance achieved when these operating systems are used on a Model A3 CPU will be somewhat less than when they execute on a Model 3 CPU.

67:10 THE 3062 ATTACHED PROCESSING UNIT

As shown in Figure 67.10.1, the 3062 APU contains an instruction execution unit, dynamic address translation hardware, a translation lookaside buffer, a 32K-byte high-speed buffer, and a processor storage control function. These components operate in the same way in the 3062 APU as in the Model A3 and 3 CPU's. Like a Model 1, 3, or A3 CPU, a 3062 APU is water-cooled.

The 3062 APU differs from the Model A3 CPU in that the 3062 does not contain any processor storage and its processor storage control function does not contain any channel buffers and channel control hardware. The processor storage access priority control function in the 3062 APU and Model A3 CPU communicate with each other to support the sharing of all processor storage available in the Model A3 CPU. The configuration panel on the 3066 Model 3 System Console is used to configure processor storage in a Model A3 CPU in the same way as for a Model 168 uniprocessor Model 3 system.

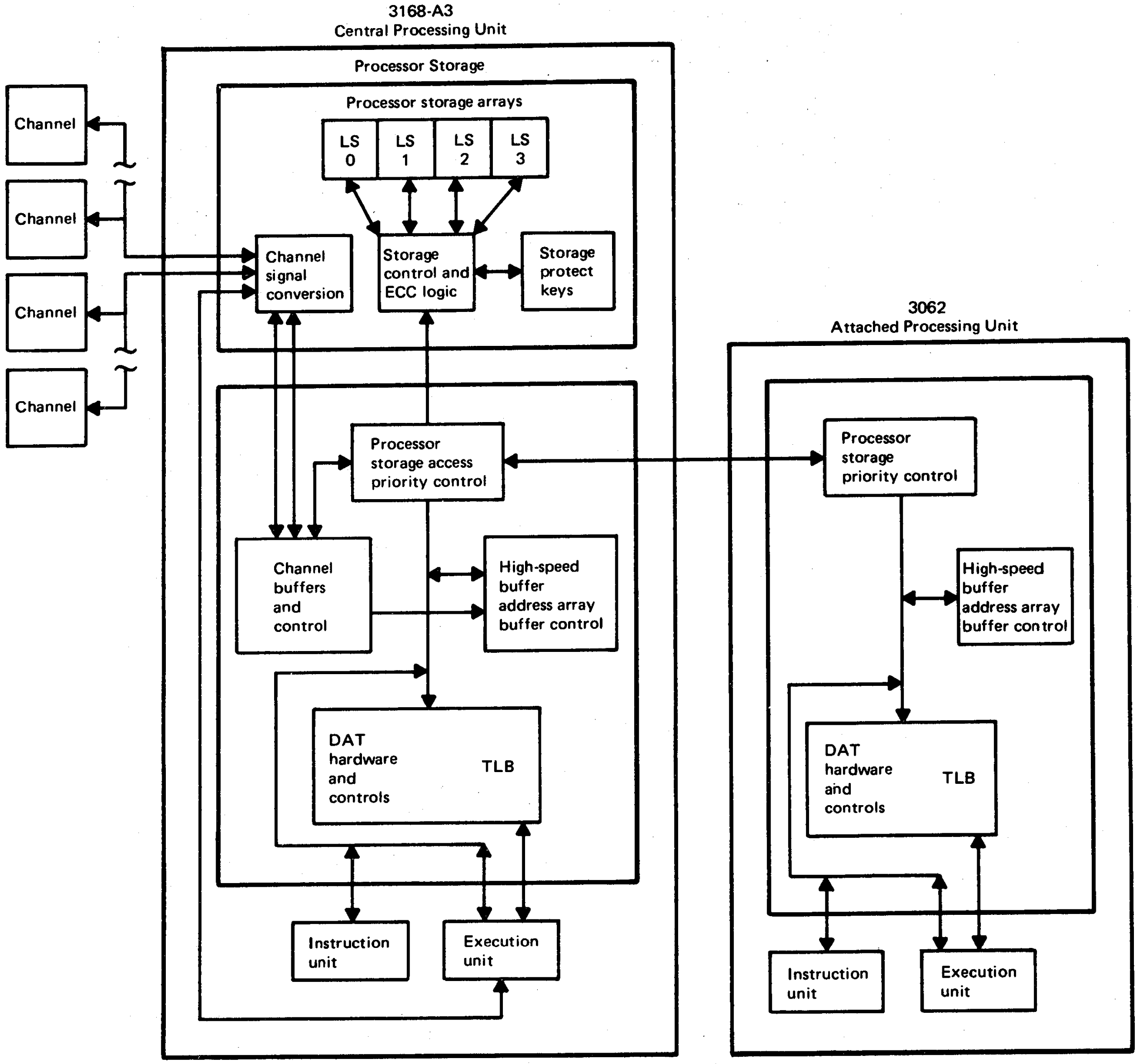

3168-A3
Central Processing Unit
Processor Storage
Processor storage arrays
LS 0
LS 1
LS 2
LS 3
Channel
Channel
Channel
Channel
Channel signal conversion
Storage control and ECC logic
Storage protect keys
Processor storage access priority control
Channel buffers and control
High-speed buffer address array buffer control
DAT hardware and controls
TLB
Instruction unit
Execution unit
3062
Attached Processing Unit
Processor storage priority control
High-speed buffer address array buffer control
DAT hardware and controls
TLB
Instruction unit
Execution unit

Figure 67.10.1. Organization of the Model A3 CPU and 3062 APU

The 3062 APU contains multiprocessing functions and the following standard features, which are identical to the same features for Models 3 and A3 of the Model 168:

- The standard and new System/370 instructions listed for the Model 168 in Section 20:30
- BC and EC mode of operation
- Dynamic address translation
- Reference and change recording
- Instruction retry
- Interval timer (3.3 ms resolution)

- Time-of-day clock
- Clock comparator and CPU timer
- Monitoring feature
- Program event recording
- Program interruption for SSM instruction
- Expanded machine check interruption
- ECC on processor storage
- Byte-oriented operands
- Store and fetch protection
- High-speed buffer storage - 32K bytes
- Writable and read-only control storage
- Store status function

The only optional feature available for the 3062 APU is High-Speed Multiply for fixed- and floating-point multiply instructions. This feature (not recommended for field installation) can be installed only on the 3062 APU, only on the Model A3 CPU, or on both the 3062 APU and the Model A3 CPU.

As an instruction processor, the 3062 APU is functionally equivalent to the Model A3 CPU except for the following:

- The Direct Control feature is not supported for the 3062 APU. This feature is present only in the Model A3 CPU.
- The 7000-series compatibility features cannot be installed on the 3062 APU.
- The Power Warning feature cannot be installed on the 3062 APU.
- Input/output operations are normally handled only by the Model A3 CPU. While I/O instructions are implemented in the 3062 APU, they are utilized only in special situations (such as for the execution of diagnostic routines).

67:15 UNIPROCESSOR AND ATTACHED PROCESSOR SYSTEM HARDWARE DIFFERENCES

The Model 168 Model A3 CPU and 3062 APU differ from a Model 3 uniprocessor CPU in that they contain multiprocessing functions similar to those contained in multiprocessor models of the Model 168. This is the only difference between Models A3 and 3 of the Model 168 CPU (3168 Processing Unit). Differences between the 3062 APU and the Model A3 CPU, previously discussed, are also differences between the 3062 APU and the Model 3 CPU.

Except for channel reconfiguration hardware, the interprocessor communication functions provided in the Model 168 Model A3 CPU and 3062 APU are the same as those provided by the multiprocessing feature in Model 168 tightly coupled multiprocessing configurations. See Table 67.15.1 at the end of this subsection for a comparison of the hardware features of the Model 3 CPU (uniprocessor and multiprocessor models), Model A3 CPU, and 3062 APU.

The following are implemented in the Model A3 CPU and 3062 APU but not in the uniprocessor Model 3 CPU:

- Prefixing, which is a method of assigning unique areas of processor storage, one for the Model A3 CPU and one for the 3062 APU, the addresses 0 to 4095. Associated with the prefixing facility are the SET PREFIX and STORE PREFIX instructions. Prefixing operates in a Model 168 AP System in the same way as in a Model 168 tightly coupled multiprocessing configuration (see discussion in Section 60:10 under "Prefixing").

- CPU addressing, which is required to specifically identify each processing unit (Model A3 CPU and 3062 APU) in the AP System configuration. The Model A3 CPU has a CPU address of 0001 while the 3062 APU has a CPU address of 0000. The STORE CPU ADDRESS (STAP) instruction can be executed in both processing units and returns a version code of X'81' to identify an AP System configuration. A CPU address is stored during certain external interruptions to identify the CPU involved and is used in the SIGNAL PROCESSOR instruction.

- Time-of-day clock synchronization, which is required to provide one logical clock for the AP System. Hardware changes involving the time-of-day clock are the same for a Model 168 AP System as for a Model 168 tightly coupled multiprocessing configuration, as discussed under "Time-of-Day Clock" in Section 60:10, except for one item. The oscillator in the 3062 APU is used for both time-of-day clocks in the AP System. The pulses from the oscillator in the 3062 APU are broadcast to the Model A3 CPU. When the STORE CLOCK instruction is issued, bit 52 of the stored value is set to 0 if the 3062 issued the instruction or 1 if the Model A3 CPU issued the instruction.

- Interprocessor programmed communication, which is required to enable one processing unit (Model 168 A3 CPU or 3062 APU) to request services of the other processing unit and to alert it to conditions to which it must respond. The SIGNAL PROCESSOR instruction is provided for this communication and operates in the same way in a Model 168 AP System as in a Model 168 tightly coupled multiprocessing configuration (see discussion under "Interprocessor Programmed Communication" in Section 60:10). Buffer timings for the AP System are the same as for the tightly coupled multiprocessing configuration as discussed in Section 60:10.

- Interprocessor hardware communication, which is required to alert one processing unit to conditions in the other processing unit and to synchronize certain operations in both processing units. The hardware facilities provided for hardware communication between a Model A3 CPU and 3062 APU are the same as those provided for the two processing units in a Model 168 tightly coupled multiprocessing configuration (see discussion under "Interprocessor Hardware Communication" in Section 60:10).

- Logout of additional data in the CPU extended logout area that is unique to a Model 168 AP System.

- Channel control reconfiguration, which is provided to enable diagnostic programs contained on I/O devices to be loaded and executed in the 3062 APU. Channel control reconfiguration is accomplished using the system control panel of the 3066 System Console. When the maintenance switch is placed in the APU position and the system reset button is pressed, the channel control interface is switched from the Model A3 CPU to the 3062 APU. Channel access to processor storage is still controlled by the processor storage control function in the Model A3 CPU, which must remain operational in order for the 3062 APU to access the channels.

When channel control reconfiguration is in effect, the 3062 APU can issue I/O instructions and receive I/O interruptions. When channel control reconfiguration is not in effect, a channel-not-operational indication (condition code 3) is returned when the 3062 APU issues an I/O instruction and no I/O interruptions are presented to the 3062 APU. Note that when the APS Maintenance Switch RPQ is installed, channel control reconfiguration can be made effective to attempt to continue system operations using only the APU after a hardware failure in the 3168 processor makes the CPU unusable.

Table 67.15.1. Comparison of hardware features in the Model 3 CPU (uniprocessor and multiprocessor models), Model A3 CPU, and 3062 APU

Feature	Model 3 Uniprocessor CPU (3168-3)	Model 3 Multiprocessor CPU (3168-3)	Model A3 CPU (3168-A3)	APU (3062)
BC and EC mode of operation	Std	Std	Std	Std
Byte-oriented operands	Std	Std	Std	Std
Channels	Up to 12	Up to 12	Up to 12	No
Channel control reconfiguration hardware	No	No	No	Std
Channel dual I/O bus	Std	Std	Std	No
Channel indirect data addressing	Opt	Opt	Opt	No
Channel reconfiguration hardware	No	Std	No	No
Channel retry data	Std	Std	Std	No
Clock comparator and CPU timer	Std	Std	Std	Std
Direct control	Std	Std	Std	No
Dynamic address translation	Std	Std	Std	Std
ECC on processor storage	Std	Std	Std	No
Expanded machine check interruptions	Std	Std	Std	Std
High-speed buffer	Std-32K	Std-32K	Std-32K	Std-32K
Instruction retry	Std	Std	Std	Std
Interprocessor hardware communication	No	Std	Std	Std
Interprocessor programmed communication (SIGP instruction)	No	Std	Std	Std
Interval timer	Std	Std	Std	Std
Monitoring feature	Std	Std	Std	Std
Prefixing	No	Std	Std	Std
Processor storage	1 to 8 megabytes	1 to 8 megabytes	1 to 8 megabytes	No
Program event recording	Std	Std	Std	Std
Reference and change recording	Std	Std	Std	Std
Service processor	Std	Std	Std	No
Trace Unit	Std	Std	Std	Std
SSM instruction interruption	Std	Std	Std	Std
Store and fetch protection	Std	Std	Std	Std
Store status function	Std	Std	Std	Std
Time-of-day clock	Std	Std	Std	Std

Table 67.15.1. (continued)

Feature	Model 3 Uniprocessor CPU (3168-3)	Model 3 Multiprocessor CPU (3168-3)	Model A3 CPU (3168-A3)	APU (3062)
Translation lookaside buffer	Std	Std	Std	Std
Writable and read-only control storage	Std	Std	Std	Std
Channel-to-Channel Adapter	Opt	Opt	Opt	No
Extended Unit Control Words on 2880 channels	Opt	Opt	Opt	No
High-Speed Multiply	Opt	Opt	Opt	Opt
Integrated Storage Controls	Opt	Opt	Opt	No
Power Warning	Opt	Opt	Opt	No
7070/7074 Compatibility	Opt	Opt	Opt	No
7080 Compatibility	Opt	Opt	Opt	No
709/7090/7094/7094 II Compatibility	Opt	Opt	Opt	No
Virtual Machine Assist RPQ	Opt	Opt	No	No

67:20 3066 MODEL 3 SYSTEM CONSOLE

The 3066 Model 3 System Console provides console functions for both the Model A3 CPU and 3062 APU. The 3066 Model 3 performs the same functions as the 3066 Model 2 for the Model A3 and has additional capabilities that are required to support an AP System configuration. The system control panel on the 3066 Model 3 has an additional panel in the lower right-hand corner that provides the following additional operator control switches.

- CPU/APU Select switch. Most controls and indicators are shared by the Model A3 CPU and 3062 APU. The setting of the CPU/APU Select switch determines whether the function is performed for the Model A3 CPU or 3062 APU. The following operate under the control of this switch:

  CRT display
  Microfiche display
  Manual indicator
  Wait indicator
  Data keyboard
  Manual Entry Select switch
  Data Select switch
  Advance Address pushbutton
  Display pushbutton
  Store pushbutton
  Store Status pushbutton
  Computer Reset pushbutton
  System Activity Meter Switch on additional
  console panel (discussed below)
  Check Reset pushbutton
  Set IC pushbutton
  Restart pushbutton
  Trans Adr and Displ Main pushbutton
  Logout pushbutton
  Diagnose Restart pushbutton
  Load I-bfr pushbutton
  Start Ripple pushbutton

CS Transfer pushbutton
Data Bus Scoping
Interrupt pushbutton
Set PSW pushbutton
Start pushbutton
Stop pushbutton

- Maintenance switch. This switch has a CPU, NORM, and APU position and controls whether diagnostics are loaded in the Model A3 CPU only, both the Model A3 CPU and 3062 APU, or 3062 APU only, respectively, when the Load MD pushbutton is pressed. In addition, this switch controls the channel control reconfiguration capability discussed previously.

- System Activity Meter switch. This switch selects the processing units whose activity is to be displayed on the system activity meter on the system control panel. With this switch in the normal position, the system activity meter displays the activity of the processing unit specified in the CPU/APU Select switch. With this switch in the "CPU or APU" position, the combined activity of either processing unit is displayed (logical ORing of activity). With this switch in the "CPU+APU" position, the system activity meter displays the overlapped activity of both processing units (logical ANDing of activity).

- Resident Diagnostic switch. This switch is used in conjunction with the Load MD pushbutton.

All manual controls on the system control panel on the 3066 function in the same way in an AP system as in a uniprocessor system except for the system reset pushbutton, load pushbutton, system clear pushbutton, and time-of-day clock security switch. When one of these is used, it affects both the Model A3 CPU and the 3062 APU. Specifically, when a system reset (with or without the system clear pushbutton pressed) or load (IPL with or without the system clear pushbutton pressed) is performed by pressing the system reset or load key, the same function is also automatically sent (broadcast) to the other system. The setting of the system clear pushbutton is propagated as well. System reset causes the prefix register in each CPU to be reset to zero.

When the time-of-day clock security switch is held in the enable set position, this setting is sent to both the Model A3 CPU and 3062 APU so that both clocks can be set.

67:25 RECORDING AND DIAGNOSTIC ROUTINES

The same diagnostic programs (ST370, HDM, etc.), microdiagnostics consisting of approximately 25 programs (fault-locating tests), and recording programs (RMS, SEREP, EREP, OLT, LOA, etc.) are provided for a Model 168 AP System as for uniprocessor Model 168 systems and are updated as appropriate for AP System multiprocessing function hardware. Additional microdiagnostics are provided to test the multiprocessing function hardware.

When a malfunction occurs in the Model A3 CPU or 3062 APU, the appropriate uniprocessor microdiagnostics should be executed first on the failing instruction processor. If these diagnostics do not locate the fault, the following microdiagnostics, which test operations in both the Model A3 CPU and 3062 APU and are different from the tightly coupled multiprocessing feature microdiagnostics, should be executed:

- AP Functional Test

- AP Hardware Test

- AP System Test

The 3062 APU contains a trace unit and trace buffers identical in function and operation to those in the Model A3 CPU (as discussed in Section 65:10). The service processor in the Model A3 CPU accepts the status data (191 permanently monitored points, eight probe inputs, and logout data) from the trace buffers in the 3062 APU just as for the Model A3 CPU.

SECTION 70: COMPARISON TABLES

These tables have been included for quick reference. The first compares hardware features of the System/360 Model 65 and System/370 Models 158, (Models 1 and 3), 165, 165 II, and 168 (Models 1 and 3). The second compares OS MFT, MVT, VS1, VS2 SVS (Release 1.7), and VS2 MVS (as of Release 3.7) support of the Model 168 (Models 1 and 3).

70:05: COMPARISON TABLE OF HARDWARE FEATURES OF THE SYSTEM/360 MODEL 65 AND SYSTEM/370 MODELS 158 (MODELS 1 AND 3), 165, 165 II, AND 168 (MODELS 1 AND 3)

Hardware Feature	System/360 Model 65	System/370 Model 158 (Models 1 and 3)	System/370 Model 165	System/370 Model 165 II	System/370 Model 168 (Models 1 and 3)
I. CPU					
A. BC mode of system operation	Comparable to BC mode	Standard	Standard	Standard	Standard
B. EC mode of system operation	Not implemented	Standard	Not implemented	Standard	Standard
C. Instruction set					
1. Standard set (binary arithmetic)	Standard	Standard	Standard	Standard	Standard
2. Decimal arithmetic	Standard	Standard	Standard	Standard	Standard
3. Floating-point arithmetic	Standard	Standard	Standard	Standard	Standard
4. Extended precision floating-point	Not available	Optional (no-charge)	Standard	Standard	Standard
5. New instructions a. COMPARE LOGICAL CHARACTERS UNDER MASK COMPARE LOGICAL LONG HALT DEVICE INSERT CHARACTERS UNDER MASK LOAD CONTROL MONITOR CALL MOVE LONG SET CLOCK SHIFT AND ROUND DECIMAL START I/O FAST RELEASE STORE CHANNEL ID STORE CHARACTERS UNDER MASK STORE CLOCK STORE CONTROL STORE CPU ID	Not available	Standard	Standard (except for MONITOR CALL)	Standard	Standard
b. CLEAR I/O COMPARE AND SWAP COMPARE DOUBLE AND SWAP INSERT PSW KEY LOAD REAL ADDRESS PURGE TLB RESET REFERENCE BIT SET CLOCK COMPARATOR SET CPU TIMER	Not available	Standard	Not available	Standard	Standard

Hardware Feature	System/360 Model 65	System/370 Model 158 (Models 1 and 3)	System/370 Model 165	System/370 Model 165 II	System/370 Model 168 (Models 1 and 3)
STORE CLOCK COMPARATOR STORE CPU TIMER SET PSW KEY FROM ADDRESS STORE THEN AND SYSTEM MASK STORE THEN OR SYSTEM MASK					
D. Overlap of instruction fetching and preparation with instruction execution	Instruction unit normally prepares one instruction at a time. Imprecise interruptions occur only for storage violations.	Instruction prefetching is performed. One prefetched instruction is decoded, no operand prefetching is performed. Imprecise interruptions cannot occur. A 64-word buffer is provided in the Model 1. A 128-word buffer is provided in the Model 3.	Instruction unit can process several instructions while execution unit executes one instruction. Imprecise interruptions can occur.	Same as Model 168	Same as Model 165 except that instruction and execution unit implementation is enhanced and imprecise interruptions cannot occur.
E. High-speed multiply	Not available	Not available	Optional	Optional	Optional
F. CPU cycle time	200 nanoseconds, 8-byte data path	115 nanseconds, 4-byte parallel data path	80 nanoseconds, 8-byte data path	Same as Model 165	Same as Model 165
G. Dynamic address translation	Not available	Standard	Not available	Standard	Standard
H. Interval timer	Standard (16.6 ms resolution)	Standard (3.33 ms resolution)	Standard (3.33 ms resolution)	Standard (3.33 ms resolution)	Standard (3.33 ms resolution)
I. Time-of-day clock	Not available	Standard	Standard	Standard	Standard
J. CPU timer and clock comparator	Not available	Standard	Not available	Standard	Standard
K. Monitoring feature	Not available	Standard	Not available	Standard	Standard
L. Program event recording	Not available	Standard	Not available	Standard	Standard
M. Direct control	Optional	Optional	Standard	Standard	Standard
N. Interruption for SSM instruction	Not implemented	Standard	Not implemented	Standard	Standard

Hardware Feature	System/360 Model 65	System/370 Model 158 (Models 1 and 3)	System/370 Model 165	System/370 Model 165 II	System/370 Model 168 (Models 1 and 3)
O. Compatibility features (all are optional and mutually exclusive except where noted otherwise)	1. 7070/7074 2. 7080 (for both 705 and 7080) 3. 709/7040/7044/ 7090/7094/7094II	1. 1401/40/60, 1410/7010 2. OS/DOS 3. 7070/7074 (features are no-charge and not mutually exclusive)	1. 7070/7074 2. 7080 (for both 705 and 7080) 3. 709/7090/7094 7094II (does not include 704, 7040, 7044)	Same as Model 165	Same as Model 165
P. Control logic	Microprogram in ROS	Microprogram in reloadable control storage	Microprogram in capacitor ROS and monolithic WCS	Same as Model 168	Microprogram in monolithic ROS and monolithic WCS
Q. Instruction retry by hardware	No	Yes	Yes	Yes	Yes
R. Machine check interruption	One level of machine check provided for all machine errors and one machine check mask.	Occurs after corrected and uncorrected errors. There are seven types of machine check and many are individually maskable.	Occurs after corrected and uncorrected errors. There are four types of machine check and many are individually maskable.	Same as Model 168	Same as Model 165, except five types of machine check are implemented and more data is logged by a Model 168.
S. Fixed storage area size in lower storage (including logout area for machine and channel errors)	328 bytes including CPU and channel logouts	1184 bytes reducible to 512 if extended logout area of 672 bytes is moved	1504 bytes reducible to 512 if the extended logout area of 992 bytes is moved	Same as Model 168	1928 bytes reducible to 512 if the extended logout area of 1416 bytes is moved
T. Multiprocessor systems	1. Multisystem optional feature permits interconnection of two Model 65s. Main storage is shared (512K or more). Direct control is required. main processor 2. The support or in an ASP configuration can be a Model 65. Two or three systems are connected via a Channel-to-Channel Adapter.	Same as Model 168	1. A multisystem feature is not available. 2. A Model 165 can be a support or a main processor in an ASP configuration	Same as Model 165	1. The 3068 Multisystem Communication Unit is used to connect two Model 168 systems (any combination of Models 1 and 3) together for a tightly coupled (shared storage) multiprocessing configuration. 2. JES3 (Job Entry Subsystem 3) of OS/VS2 Release 3 supports the Model 168 in a loosely coupled multiprocessing configuration.

Hardware Feature	System/360 Model 65	System/370 Model 158 (Models 1 and 3)	System/370 Model 165	System/370 Model 165 II	System/370 Model 168 (Models 1 and 3)
					3. A Model 168 can be a main or support processor in an ASP configuration.
U. Attached Processor System	Not available	Not available	Not available	Not available	Supported (RPQs required for Model 1)
V. Power warning	Not available	Optional	Not available	Not available	Optional
W. Virtual machine assist	Not available	Optional	Not available	RPQ feature	RPQ feature
X. Remote analysis unit	Not available	Yes - service processor is standard.	Yes - 2955 Remote Analysis Unit is optional.	Same as a Model 165	Yes - 2955 Remote Analysis Unit is optional in the Model 1, service processor is standard in the Model 3.
Y. Integrated Storage Controls	Not available	Optional for attachment of 3330-series, 3340/3344, and/or 3350 disk storage, or the 3850 Mass Storage System	Not available	Not available	Same as Model 158
II. STORAGE					
A. Processor (main) storage sizes	256K 512K 768K 1024K	- 512K - 1024K 1536K 2048K 3072K 4096K	- 512K - 1024K 1536K 2048K 3072K	- - - 1024K - 2048K 3072K	- - - 1024K - 2048K 3072K 4096K 5120K 6144K 7168K 8192K
B. Type of processor storage	Ferrite cores	Monolithic technology	Ferrite cores	Ferrite cores	Monolithic technology
•C. Processor storage interleaving	Two-way interleaving of sequential accesses other than by the channels is provided.	None	Storage is 4-way doubleword interleaved for CPU and channel requests.	Same as Model 165	Storage is 4-way doubleword interleaved.

Hardware Feature	System/360 Model 65	System/370 Model 158 (Models 1 and 3)	System/370 Model 165	System/370 Model 165 II	System/370 Model 168 (Models 1 and 3)
D. High-speed buffer storage	No	8K is standard in the Model 1. 16K is standard in the Model 3.	8K is standard, 8K more can be added. 80-nanosecond cycle.	Same as Model 165	Model 1 is same as Model 165. 32K is standard in the Model 3. Buffer row deletion is implemented in both models.
E. Processor storage validity checking	Parity checking by byte. No hardware error correction is provided.	ECC checking on a doubleword. Single-bit errors are corrected by hardware.	Same as Model 158	Same as Model 158	Same as Model 158
F. Byte-oriented operands	No	Standard	Standard	Standard	Standard
G. Store and fetch protection	Standard	Standard	Standard	Standard	Standard
H. Shared processor storage	Optional (Model 65 system shares 2361 Core Storage with a Model 50, 65, or 75)	Not available	Not available	Not available	Not available
I. 2361 Core Storage	Optional. Up to 8 million bytes can be attached.	Cannot be attached	Cannot be attached	Cannot be attached	Cannot be attached
III. CHANNELS					
A. Total number per system	Up to 7	Byte multiplexer channel 0 and block multiplexer channels 1 and 2 are standard. Block multiplexer channels 3 to 5 are optional. Channel 4 can be a second byte multiplexer channel instead of a block multiplexer in systems with 768K or more and channel 3 installed. Selector mode standard for all block multiplexers. Channel rate is 1.5 MB/sec.	1. Up to 7 standard 2. Up to 12 with Extended Channels optional feature	Same as Model 165	Same as Model 165

Hardware Feature	System/360 Model 65	System/370 Model 158 (Models 1 and 3)	System/370 Model 165	System/370 Model 165 II	System/370 Model 168 (Models 1 and 3)
B. 2870 Multiplexer Channel	One or two can be attached (192 subchannels)	Does not attach (Byte multiplexer channel 0 or 4 on a Model 1 can have 256 nonshared subchannels or eight shared subchannels and 120 nonshared subchannels. Byte multiplexer channel 0 or 4 on a Model 3 can have 256 nonshared without subchannel sharing installed or 256 nonshared less 16 or 32 for each control unit position wired for 16 or 32 shared subchannels.	Same as Model 65	Same as Model 65	Same as Model 65
C. 2860 Selector Channel (1.3 MB/sec.)	A maximum of 6 can be attached	Does not attach	Same as Model 65	Same as Model 65	Same as Model 65
D. 2880 Block Multiplexer Channel (1.5 MB/sec). Two-Byte Interface feature permits a 3.0 MB/sec. data rate	Cannot be attached	Does not attach	A maximum of 6 can be attached without the Extended Channels feature, a maximum of 11 with this feature.	Same as Model 165	Same as Model 165
1. Maximum number of subchannels		For all Model 1 block multiplexer channels, 16 shared and 480 nonshared. For all Model 3 block multiplexer channels when the second byte multiplexer channel is not installed: a) 736 nonshared with no subchannel sharing installed b) With subchannel sharing installed, 40 shared and 736 nonshared less 1	1 shared 56 nonshared	Same as Model 165	1 shared and 56 nonshared without extended unit control words feature, 1 shared and up to 256 nonshared with feature

Hardware Feature	System/360 Model 65	System/370 Model 158 (Models 1 and 3)	System/370 Model 165	System/370 Model 165 II	System/370 Model 168 (Models 1 and 3)
		for each shared subchannel For the Model 3 with the second byte multiplexer installed: a) 480 nonshared with no subchannel sharing installed b) With subchannel sharing installed, 32 shared and 480 nonshared less 1 for each shared subchannel			
E. Channel dual I/O bus	No	No	No	No	Yes
F. Maximum aggregate data rate for channels	In excess of 4 MB/sec for one 2870 and six 2860s	6.75 MB/sec for five block multiplexer channels	In excess of 9 MB/sec with twelve channels	Same as Model 165	17MB/sec
G. Channel retry data provided after channel error	Yes	Yes	Yes	Yes	Yes
H. Channel-to-Channel Adapter	Optional on 2860	Optional	Optional on 2860	Optional on 2860	Optional on 2860
I. Extended Unit Control Words on the 2880	-	-	Optional	Optional	Optional
J. Channel indirect data addressing	Not available	Standard	Not available	Optional (required by the virtual storage programming systems)	Optional (required by the virtual storage programming systems)

Hardware Feature	System/360 Model 65	System/370 Model 158 (Models 1 and 3)	System/370 Model 165	System/370 Model 165 II	System/370 Model 168 (Models 1 and 3)
IV. OPERATOR CCNSOLE DEVICES	1. 1052 Printer-Keyboard (optional) 2. Second 1052 Printer-Keyboard is optiorial. 3. A 2250 Display Unit and a remote 2150 Console are optional. 4. Other devices can be used as primary and secondary consoles.	1. Display ccnsole with keybcard and light pen is standard. Hardcopy can be provided optionally via a 3213 Printer for display mode. Display console can also cperate in printer keyboard mode instead of display mcde. 2. 2150 Console with 1052-7 Printer-Keyboard 3. Additional cônsoles, such as display units, are optional. (Store status function is provided.) 4. 3056 Remote System Console in addition to the standard display ccnsole is optional.	1. Standalone 3066 Model 1 system Ccnsole is required. It includes: a. A CRT-keyboard combination for operator/system communication b. An indicator viewer c. A microfiche document viewer d. A processor storage configuration plugbcard e. A system activity meter f. A device for loading WCS and microdiagnostics The store status function is not prcvided. 2. Cptionally, other devices can be used as secondary ccnscles as listed for the Model 65.	1. Standalone 3066 Model 1 System Console is required. The store status function is supported.	1. Standalcne 3066 Model 2 System Console provides same features as 3066 Model 1 and store status function. Cther conscles can be attached as for Model 165.
V. I/O DEVICES					
A. 3505 Card Reader 3525 Card Punch	No	Yes	Yes	Yes	Yes
B. 3211 Printer	Yes	Yes	Yes	Yes	Yes
C. 3803/3420 Magnetic Tape Subsystem (Models 3,5,7 and 4,6,8)	Yes, except Model 8	Yes	Yes	Yes	Yes

Hardware Feature	System/360 Model 65	System/370 Model 158 (Models 1 and 3)	System/370 Model 165	System/370 Model 165 II	System/370 Model 168 (Models 1 and 3)
D. Direct access devices (2311, 2314, 2303, 2301, and 2321)	All attach	All except 2301 drum	Same as Model 65	Same as Model 65	Same as Model 65
E. 3330-series disk storage	No	Yes (all models)	Yes (Models 1 and 2 only)	Yes (all models)	Yes (all models)
1. 3830 Storage Control Model 1	-	Yes	Yes	Yes	Yes
2. 3830 Storage Control Model 2	-	Yes	Yes	Yes	Yes
3. Integrated Storage Controls feature	-	Yes	No	No	Yes
F. 2305 facility Models 1 and 2	No	2305 Model 2 only	Yes cn 2880	Yes on 2880	Yes on 2880
G. 3340 Direct Access Storage Facility	No	Yes (attachment via 3830 Model 2 and Integrated Storage Controls)	No	Yes (attachment via 3830 Model 2)	Yes (attachment via 3830 Model 2 and Integrated Storage Controls)
H. 3344 Direct Access Storage	No	Yes (attachment via 3830 Model 2 and Integrated Storage Controls)	No	Yes (attachment via 3830 Model 2)	Yes (attachment via 3830 Model 2 and Integrated Storage Controls)
I. 3350 Direct Access Storage	No	Yes (attachment via 3830 Model 2 and Integrated Storage Controls)	No	Yes (attachment via 3830 Model 2)	Yes (attachment via 3830 Model 2 and Integrated Storage Controls)
J. 3410/3411 Magnetic Tape Subsystem	No	Yes	No	No	No
K. 3540 Diskette I/O Unit	No	Yes	No	Yes	Yes
L. 3600 Finance Communication System	No	Yes	No	Yes	Yes
M. 3650 Retail Store System	No	Yes	No	Yes	Yes
N. 3660 Supermarket System	No	Yes	No	Yes	Yes
O. 3704, 3705-I, and 3705-II Communications Controllers	Yes, emulation mode only	Yes, emulation mode and network control program modes	Same as Model 158	Same as Model 158	Same as Model 158
P. 3740 Data Entry System	Yes	Yes	Yes	Yes	Yes

Hardware Feature	System/360 Model 65	System/370 Model 158 (Models 1 and 3)	System/370 Model 165	System/370 Model 165 II	System/370 Model 168 (Models 1 and 3)
Q. 3767 Data Communication Terminal	Yes	Yes	No	Yes	Yes
R. 3770 Data Communication System	Yes	Yes	No	Yes	Yes
S. 3780 Data Communications Terminal	Yes	Yes	Yes	Yes	Yes
T. 3790 Communication System	No	Yes	No	Yes	Yes
U. 3800 Printing Subsystem	No	Yes	No	Yes	Yes
V. 3850 Mass Storage System	Nc	Yes via 3830 Model 3 and Integrated Storage Controls	No	Yes via 3830 Model 3	Yes via 3830 Mcdel 3 and Integrated Storage Controls
W. 3881 Optical Mark Reader	Nc	Yes	No	No	No
X. 3886 Optical Character Reader	Nc	Yes	No	Yes	Yes
Y. 3890 Document Processor	No	Yes	No	Yes	Yes

70:10 OS AND OS/VS SUPPORT OF THE MODEL 168 (MODELS 1 AND 3)

Hardware Feature	OS MFT and MVT	OS/VS1	OS/VS2 SVS	OS/VS2 MVS
I. CPU				
A. Mode of system operation	BC mode only. Up to 15 problem program partitions or regions.	EC and DAT modes only. One virtual storage of up to 16 million bytes is supported. Up to 52 partitions of which 15 can be problem program.	EC and DAT modes only. One virtual storage of 16 million bytes is supported. Up to 63 problem program regicns of which up to 42 can be TSO foreground regions.	EC and DAT modes only. Multiple virtual storages are supported. Each user has cne 16-million-byte virtual storage for user prcgrams, system programs, shared data, and shared program areas. The maximum number of concurrent users is limited only by the availability of system resources (external page and real storage).
B. Instruction set				
1. Standard set (binary arithmetic)	All languages	All languages	All languages	All languages
2. Decimal arithmetic	All languages except FORTRAN	All languages except FCRTRAN	All languages except FORTRAN	All languages except FORTRAN
3. Floating-point arithmetic	All languages except RPG	All languages except RPG	All languages except RPG	All languages except RPG
4. Extended precision floating-point	Assemblers F and H, PL/I Optimizing Compiler, PL/I Checkout Compiler, FORTRAN H, FORTRAN H-Extended	Same as MFT and MVT	Same as MFT and MVT	Same as MFT and MVT
5. New instructions a. COMPARE LOGICAL CHARACTERS UNDER MASK COMPARE LOGICAL LCNG INSERT CHARACTERS UNDER MASK LOAD CONTROL MCNITOR CALL MOVE LONG SET CLOCK SHIFT AND ROUND DECIMAL START I/O FAST RELEASE STORE CHANNEL ID STORE CHARACTERS UNDER MASK STORE CLOCK STORE CONTROL STORE CPU ID	Mnemonics in Assemblers F and H. Option to generate certain instructions in ANS Full COBOL (CLCL, MVCL, ICM, SRP).	Same as MFT and MVT	Same as MFT and MVT	Same as MFT and MVT

Hardware Feature	OS MFT and MVT	OS/VS1	OS/VS2 SVS	OS/VS2 MVS
b. LOAD REAL ADDRESS PURGE TLB RESET REFERENCE BIT SET CLOCK COMPARATOR SET CPU TIMER STORE CLOCK COMPARATOR STORE CPU TIMER STORE THEN AND SYSTEM MASK STORE THEN OR SYSTEM MASK	Supported by Assembler F, as of OS Release 21.6. Not supported by Assembler H.	All are supported by the System Assembler	Same as OS/VS1	Same as OS/VS1
c. CLEAR I/O COMPARE AND SWAP COMPARE DOUBLE AND SWAP INSERT PSW KEY SET PSW KEY FROM ADDRESS	Not supported	Supported	Supported	Supported
C. Interval timer	Supported for timing facilities, except for time of day	Supported for all timing facilities (except time of day) unless the extended timer option is included in the VS1 control program	Not supported	Not supported
D. Time-of-day clock	Supported for time of day	Same as MFT and MVT	Same as MFT and MVT	Same as MFT and MVT
E. Clock comparator and CPU timer	Not supported	Supported for job step and interval timing when extended timer option is included in the VS1 control program	Supported for timing facilities except for time of day	Supported for timing facilities except time of day
F. Expanded machine check interruptions	Supported by MCH	Same as MFT and MVT	Same as MFT and MVT	Same as MFT and MVT
G. Monitoring feature	Supported by GTF and an Assembler mnemonic	Same as MFT and MVT	Same as MFT and MVT	Same as MFT and MVT
H. Program event recording	Not supported	Supported by Dynamic Support System	Supported by Dynamic Support System	Supported by Dynamic Support System
I. Interruption for SSM instruction	Not supported	Supported	Supported	Supported

Hardware Feature	OS MFT and MVT	OS/VS1	OS/VS2 SVS	OS/VS2 MVS
J. Compatibility features	All are supported	All are supported	All are supported	All are supported
K. Power warning	Supported by MVT as of Release 21.6	Supported	Supported	Supported
L. Multiprocessing				
1. Tightly coupled	Not supported	Not supported	Not supported	Two multiprocessor models, connected via the 3068 Multisystem Communication Unit, that share real storage are supported.
2. Loosely coupled	Supported under MVT by ASP	Not supported	Yes by ASP Version 3.1	JES3, an upward compatible extension of ASP Version 3, supports one to eight System/370 systems connected via Channel-to-Channel Adapters. JES2 with the multi-access spool capability enables from two to eight systems operating with OS/VS2 Release 3 to share input and output work queues on shared DASD.
M. Attached Processor System	Not supported	Not supported	Not supported	Supported
II. STORAGE				
A. Real storage sizes (1024K to 8196K)	All are supported	All are supported	All are supported	All are supported
B. Byte-oriented operands	Programmers can use the byte alignment hardware facility in Assembler programs	Same as MFT and MVT	Same as MFT and MVT	Same as MFT and MVT
C. Store and fetch protection	Store protect only is supported	Store and fetch protection are supported	Store and fetch protection are supported for all regions	Store and fetch protection are supported for all virtual storages
III. CHANNELS				
A. Byte multiplexer channels	One or two are supported	One or two are supported	One or two are supported	One or two are supported
B. Block multiplexer and selector channels	Supported	Supported	Supported	Supported
C. Channel retry performed	Yes	Yes	Yes	Yes
D. Channel indirect data addressing	Not supported	Supported	Supported	Supported

Hardware Feature	OS MFT and MVT	OS/VS1	CS/VS2 SVS	OS/VS2 MVS
IV. CONSOLES				
A. 3066 Console	Supported. MCS and DIDOCS required.	Same as MFT and MVT	Same as MFT and MVT	Same as MFT and MVT
B. Alternate and additional consoles supported	Yes	Yes	Yes	Yes
V. I/O DEVICES				
A. 3505 Card Reader and 3525 Card Punch	Supported	Supported	Supported	Supported
B. 3211 Printer	Supported	Supported	Supported	Supported
C. 3803/3420 Magnetic Tape Subsystem (Models 3, 5, 7 and 4, 6, 8)	Supported	Supported	Supported	Supported
D. 2314/2319 facilities	Supported for system residence, data sets, SYSIN devices, and SYSIN and SYSOUT data sets. Record Overflow and channel switching features are supported.	Supported for system residence, data sets, paging devices, JES spooling devices and data sets, and SYSIN devices. Record Overflow and channel switching features are supported.	Supported for system residence, data sets, paging devices, SYSIN and SYSOUT data sets, and SYSIN devices. Record Overflow and channel switching features are supported.	Same as OS/VS1
E. 3330-series Direct Access Storage	Supported as V.D. above. RPS, multiple requesting, sixteen-drive addressing, 32 Drive Expansion, Two-Channel Switch, Two-Channel Switch Additional, 3333 String Switch, and Record Overflow are supported. Only Models 1 and 2 are supported.	Same as V.D. above. RPS, multiple requesting, sixteen-drive addressing, 32 Drive Expansion, Two-Channel Switch, Two-Channel Switch, Additional, 3333 String Switch, and Record Overflow are supported. All models are supported.	Same as V.D. above. RPS, multiple requesting, sixteen-drive addressing, 32 Drive Expansion, Two-Channel Switch, Two-channel Switch Additional, 3333 String Switch, and Record Overflow are supported. All models are supported.	Same as OS/VS1
F. 3340 and 3344 Direct Access Storage	Not Supported	Support same as for 3330-series	Support same as for 3330-series (not yet available)	Support same as for 3330-series

Hardware Feature	OS MFT and MVT	OS/VS1	OS/VS2 SVS	OS/VS2 MVS
G. 3350 Direct Access Storage	Not supported	Supported in native and 3330 compatibility modes as for 3330-series	Supported in native and 3330 compatibility modes as for 3330-series (not yet available)	Supported in native and 3330 compatibility rodes as for 3330-series
H. 2305 Facility Models 1 and 2	Supported for system residence, data sets, and SYSIN/SYSOUT data sets. RPS and multiple requesting are supported.	Same as V.D. above except for SYSIN devices. RPS and multiple requesting are supported.	Same as V.D. above. RPS and multiple requesting are supported.	Same as OS/VS1
I. 3540 Diskette I/O Unit	Not supported	Supported as a SYSIN and SYSCUT device (not an input/output device)	Not supported	Same as OS/VS1
J. 3600 Finance Communication System	Not supported	Supported attached to a 3704/3705 in NCP/VS mode by VTAM and TCAM through VTAM	Not supported	Same as OS/VS1
K. 3650 Retail Store System	Not supported	Supported in binary synchronous control mode attached to a 3704/3705 in emulaticn mode by BTAM. Supported in synchronous data link control mode attached to a 3704/3705 in NCP/VS mode by VTAM and TCAM through VTAM	Not supported	Suppcrted in synchronous data link control rode attached to a 3704/3705 in NCP/VS mode by VTAM and TCAM through VTAM
L. 3660 Supermarket System	Not supported	Supported in binary synchronous mode attached to a 3704/3705 in emulation mode by BTAM. Supported in synchronous data link control mode attached to a 3704/3705 in NCP/VS mode by VTAM	Supported in binary synchronous mode attached to a 3704/3705 in emulation mode by BTAM	Suppcrted in synchrcnous data link control rode attached to a 3704/3705 in NCP/VS mcde by VTAM

Hardware Feature	OS MFT and MVT	OS/VS1	OS/VS2 SVS	OS/VS2 MVS
M. 3704 and 3705 Communications Controllers	Supported in emulation mode Supported in NCP mode by TCAM	Supported in emulation mode. Supported in NCP mode by TCAM. Supported in NCP/VS mode by TCAM and VTAM.	Supported in emulation mode. Supported in NCP mode by TCAM. Supported in NCP/VS mode by TCAM.	Same as OS/VS1
N. 3740 Data Entry System	Supported (BTAM, TCAM)	Supported (BTAM, TCAM, and TCAM through VTAM)	Supported (BTAM, TCAM)	Same as OS/VS1
O. 3767 Data Communication Terminal	Not supported	Supported (as a start/stop device) attached to a 3704/3705 in emulation mode by BTAM and TCAM. Supported attached to a 3704/3705 in NCP/VS mode by VTAM and TCAM through VTAM.	Supported (as a start/stop device) attached to a 3704/3705 in emulation mode by BTAM and TCAM	Same as OS/VS1
P. 3770 Data Communication System	Not supported	Supported for synchronous data link control (SDLC) operations attached to a 3704/3705 in NCP/VS mode by VTAM and TCAM through VTAM. Supported for binary synchronous communication (BSC) operations attached to a 2701 or 3704/3705 by 2770 support in BTAM, TCAM, and VTAM.	Supported for binary synchronous communication operations attached to a 2701 or 3704/3705 by 2770 support in BTAM and TCAM.	Same as OS/VS1
Q. 3780 Data Communications Terminal	Supported (BTAM, TCAM)	Supported (BTAM, TCAM, and TCAM via VTAM)	Supported (BTAM, TCAM)	Same as OS/VS1
R. 3790 Communication System	Not supported	Supported attached to a 3704/3705 in NCP/VS mode by VTAM	Not supported	Same as OS/VS1
S. 3800 Printing Subsystem	Not supported	Supported	Not supported	Supported
T. 3850 Mass Storage System	Not supported	Supported	Not supported	Supported
U. 3886 Optical Character Reader	Not supported	Supported	Not supported	Supported
V. 3890 Document Processor	Not supported	Supported	Supported	Supported

INDEX (Sections 01 to 70)

address space, definition 36
architecture design 9
ASCII/EBCDIC mode 9
Attached Processor System
  advantages 177
  channel control reconfiguration 184
  channels 178
  comparison of uniprocessor, multiprocessor, and AP system
    hardware features 185
  conversion to a tightly coupled multiprocessing configuration 180
  CPU addressing 184
  cycle times 179
  differences between uniprocessor and AP systems 183
  internal performance 177
  interprocessor hardware communication 184
  interprocessor programmed communication 184
  prefixing 183
  programming support 177, 181
  power 180
  processor storage
    access and fetch times 180
    request priority 179
    sizes 177
  recording and diagnostic programs 187
  RPQs 179, 180
  system components 177-178
  time-of-day clock 184
  3062 APU
    components 181
    differences from Model A3 CPU 181
    features 183
  3066 Model 3 Console 186

basic control mode
  compatibility with System/360 9
  programming systems support 200
buffer row deletion 22
buffer storage
  Model 1 28
  Model 3 166
  Model A3 181
change bit 66
channel control reconfiguration 184
channel dual I/O bus 32
channel indirect data addressing 69
channel masking changes for EC mode 15
channel program translation 70
channel reconfiguration hardware 160
channel retry 21
channels
  AP systems 178
  multiprocessor systems 131
  uniprocessor systems 32
CLEAR I/O instruction 20
clock comparator 20, 24
command retry 21
COMPARE AND SWAP instruction 20
COMPARE DOUBLE AND SWAP instruction 20
comparison table, Models 65, 158, 165, 165 II, and 168 hardware features 190

comparison table, OS MFT, MVT, VS1, and VS2 support of the Model 168 200
compatibility
  BC mode with System/360 9
  Model 165 with Model 168 10
  Model 3 with Model 1 of the Model 168 164
  Model A3 with Model 3 181
configuration control panel 144
  configuration pushbuttons
control registers 13
control storage
  read-only 13
  writable
    Model 1 13
    Model 3 166
CPU
  cooling 6
  cycle time 13
  extended logout area 24
cycle time
  control storage 13
  CPU 13
  local storage 13
  processor storage 26
CPU retry 21
CPU timer 21, 24

DAT hardware (see dynamic address translation)
dynamic address translation
  addresses translated 66, 67
  functions 43-45
  instruction nullification 68
  segment table origin address saving 64
  time to perform 61
  translation lookaside buffer 61
  translation process 61, 63
  translation tables 59

ECC checking 21
execution unit 13
extended control mode
  description 13-19
  programming systems support 10
extended unit control words feature 32
external page storage 54

features
  Model 1 34
  Model 3 34
  Model A3 185
  3062 Attached Processing Unit 185
fixed processor storage locations
  model-dependent 17
  model-independent 16
floating storage addressing 130, 145

imprecise interruptions 13
indirect data address list 69
indirect data address word 69
INSERT PSW KEY instruction 20
instruction nullification 68
instruction unit 13
instructions
  buffering 13
  changes to for EC mode 17
  list of standard 34

  multiprocessing 143
  new 19
  with improved execution speed 166
Integrated Storage Controls feature
  for 3330-series strings 97
  for 3340 strings 119
  Staging Adapter feature 100
  summary of features 124
interleaving 25
internal performance
  AP systems 177
  Model 1 uniprocessor systems 2
  Model 3 uniprocessor systems 2
  multiprocessor systems 140
interruptions
  external for multiprocessor models 157, 158
  machine check 22
  page translation exception 55
  segment translation exception 59
  SSM instruction 18
interval timer 11
I/O allocation switches 147
I/O configurations possible for the Model 168
  Attached Processor System 178
  multiprocessor systems 133, 134
  uniprocessor systems 33
I/O devices for the Model 168
  Attached Processor Systems 178
  multiprocessor systems 132
  uniprocessor systems 96
I/O operation retry 21

LOAD REAL ADDRESS instruction 61
local storage 13
logical storage elements 25
long-term fixing 56

machine check code 23
machine check interruptions 22
main storage (See processor storage)
Model 165 II 1, 12, 13, 19, 20, 21, 22, 28, 34, 36, 61, 190
Model 168 Model 1 and 3 differences 165
monitoring feature, description 19
monolithic technology for processor storage 8
motor generator set
  Attached Processor System 179
  Model 1 6
  Model 3 165
multiprocessing
  advantages
    loosely coupled 141
    tightly coupled 135
  asymmetric CPU features 129
  asymmetric processor storage 128
  buffer intercommunication 158
  channels
    number supported 131
    reconfiguration 160
  configuration control panel 144
    configuration pushbuttons 148-150
    floating storage address switches 145
    interleave control switch 147
    I/O allocation switches 147
    oscillator control switch 147
    pending indicators 148

storage allocation switches 145
system mode switch 144
valid indicators 148
configuration for Model 168 128
CPU addressing 155
CPU features 128
definition
loosely coupled 127
tightly coupled 127
differences between uniprocessor and multiprocessor systems 142
floating storage addressing 130, 145
instructions 143
interprocessor hardware communication 158
interprocessor programmed communication 157
I/O configurations possible 133-134
I/O devices supported 132
manual controls 160
multisystem activity meter 135
operator control panel 135
performance 140
planning considerations 163
power control 151
prefixing 154
programming support 126, 127
recording and diagnostic programs 162
SET PREFIX instruction 154
SIGNAL PROCESSOR instruction 156
storage
access and fetch times 131
configuration 145
request priority 130
sizes 130
STORE CPU ADDRESS 155
STORE PREFIX 154
time-of-day clock 156
valid system configurations 149
3066 console 135
3068 Multisystem Communication Unit 134, 143

nonpaged mode of program operation 57

optional features
Model 1 35
Model 3 35
Model 177
3062 Attached Processing Unit 183
OS MFT and MVT 1, 10, 200
OS/VS1 and OS/VS2 1, 10, 200

page 54, 58
page fault 55
page frame 54
page-in 54
page-out 54
page replacement algorithm 56
page table 54, 59, 62
page translation exception 55
paged mode of program operation 57
paging 54
paging device 54
performance in a virtual storage environment 71
factors affecting 75
increasing 82
relationship to virtual storage size 78

Power Warning feature
  AP systems 184
  multiprocessor systems 153
  uniprocessor systems 24
prefixing 154
processor storage
  AP systems 177, 180
  multiprocessor systems 131, 145
  technology 8
  uniprocessor systems 25
    cycle and access times 26
    reconfiguration 26
    sizes 25
program event recording
  comparison with monitoring feature 19
  description 18
programming systems support of the Model 168
  OS MFT and MVT 200
  OS/VS1 and OS/VS2 200
PSW
  BC mode format 14
  changes for EC mode 14
  EC mode format 14
PURGE TLB instruction 64

RAS features 21-25
read-only storage 13
real storage 40
reconfiguration, processor storage
  multiprocessor systems 145
  uniprocessor systems 26
reference bit 66
RESET REFERENCE BIT instruction 66

segment 53, 58
segment table 54, 59, 62
segment table origin address saving 64
segment translation exception 59
service processor in the Model 3
  advantages 175
  CE panel 173
  components 168
  corporate standard interface 174
  counters 172
  functions 168
  general operation 168
  internal disk file 172
  modem 174
  printer control 173
  processor unit 170
  RETAIN/370 interface 174
  trace unit 171
  2955 mode 174
SET CLOCK COMPARATOR instruction 20
SET CPU TIMER instruction 21
SET PREFIX instruction 154
SET PSW KEY FROM ADDRESS instruction 20
SET SYSTEM MASK instruction interruption 18
short-term fixing 56
SIGNAL PROCESSOR instruction 156
slot 54
standard features
  Model 1 34
  Model 3 34
  Model A3 177

3062 Attached Processing Unit 182
storage
  buffer 28
  control 13
  external page 54
  interleaving 25
  local 13
  processor (main) 25
  protect key expansion 15
  real 40
  reconfiguration 26
  ripples 27
  virtual (See virtual storage)
storage protect key 15, 66
STORE CLOCK COMPARATOR instruction 20
STORE CPU ADDRESS instruction 155
STORE CPU TIMER instruction 21
STORE PREFIX instruction 154
store status function 34
STORE THEN AND SYSTEM MASK instruction 19
STORE THEN OR SYSTEM MASK instruction 19
STO-stack 64
system console 6, 34, 60
system highlights 1-5
system space requirements 6
system technology 7-8

thrashing condition 75
time-of-day clock
  AP systems 184
  multiprocessor systems 155
translation lookaside buffer 61

virtual equals real mode 57
virtual machine assist RPQ 92
Virtual Machine Facility/370 1, 10, 45, 54, 56, 86-95
virtual machines
  advantages 93-95
  definition 86
  general operation 86-92
virtual storage
  advantages 46
  definition 40
  need for 36
  organization 58
  relationship between size and performance 78
  resources required to support 72
virtual storage address fields 60
virtual storage page 53

writable control storage
  Model 1 13
  Model 3 166

2955 Remote Analysis Unit 11, 25, 168, 174

3062 Attached Processing Unit 181

3066 System Console
  Model 2 6, 34
  Model 3 186
  multiprocessor systems 135

- 3067 Power and Coolant Distribution Unit
  - Model 2 6
  - Model 3 165
  - Model 5 179

- 3068 Multisystem Communication Unit 134, 143

- 3330-series disk storage
  - attachment via ISC 97
  - Model 11 drives 96

- 3340 direct access storage facility
  - advantages summary 122
  - alternate tracks 107, 110, 112
  - attachment via integrated storage controls 119
  - attachment via the 3830 Model 2 116
  - capacity 101, 103
  - channel switching features 117, 120
  - defect skipping 114
  - description of 3340 drives 101
  - error detection and correction 112
  - error logging 118
  - features table 123
  - fixed head feature 103, 110
  - intermixing 3340 and 3330-series drives on an attachment 121
  - multiple requesting 116
  - physical address bytes 115
  - read-only feature 105
  - rotational position sensing 116
  - sectors 114
  - seek verification 115
  - servo system 107
  - string configurations 101
  - string switching 117, 120
  - timing characteristics 114

- 3348 Data Module, for the 3340 Direct Access Storage Facility
  - advantages 103-105
  - capacity
    - Model 35 101, 108
    - Model 70 101, 109
    - Model 70F 101, 111
  - cylinder and read/write head layout
    - Model 35 107
    - Model 70 107
    - Model 70F 110
  - general description 101
  - initialization 115
  - loading and unloading 104
  - reliability 103
  - track formatting 112
  - track layout on the recording surface 105

  - 3830 Storage Control Model 2
    - features for 3360 facilities 116
    - summary of features 124

SECTION 90: OS/VIRTUAL STORAGE 1 FEATURES

If required, the OS/Virtual Storage 1 Features Supplement, GC20-1752, should be inserted here.

This page intentionally left blank

SECTION 100: OS/VIRTUAL STORAGE 2 SINGLE VIRTUAL STORAGE (SVS) FEATURES

If required, the OS/Virtual Storage 2 Single Virtual Storage (SVS) Features Supplement, GC20-1753, should be inserted here.

This page intentionally left blank

SECTION 110: VIRTUAL MACHINE FACILITY/370 FEATURES

If required, the Virtual Machine Facility/370 Features Supplement, GC20-1757, should be inserted here.

This page intentionally left blank

READER'S COMMENT FORM

A Guide to the IBM System/370 Model 168
for System/370 Model 165 Users

GC20-1755-3

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

Fold

Fold

Fold

Fold

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.
  FOLD ON TWO LINES, STAPLE AND MAIL.

GC20-1755-3

Your comments, please . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold **Fold**

First Class
Permit 40
Armonk
New York

Business Reply Mail

No postage stamp necessay if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
1133 Westchester Avenue
White Plains, New York 10604

Att: Technical Publications/Systems – Dept. 824

Fold **Fold**

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
360 Hamilton Avenue, White Plains, New York 10601
(International)

GC20-1755-3

A Guide to the IBM System/370 Model 168 for System/370 Model 165 Users Printed in U.S.A. GC20-1755-3

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
360 Hamilton Avenue, White Plains, New York 10601
(International)