CYBER 18-20 ARITHMETIC LOGIC

SUPPLEMENTARY REFERENCE MANUAL

CONTENTS

Block 1

ALU Registers

ALU Functional Areas

One of the major components of a central processing unit is the arithmetic logic unit, or ALU module. This is the part of the computer that performs the actual arithmetic and logic operations. This text introduces the ALU module in the CYBER 18-20 microprocessor; it describes the basic operation, the components, and the data flow of the ALU module.

Basic ALU Operation

The ALU module can perform two basic arithmetic operations, add and subtract; it can check sixteen-bit words for positive or negative signs, zero, and overflow; and it can make logical comparisons between two operands. The module that does these arithmetic and logic operations is found in slot M of the microprocessor chassis (see figure 1-1).

The ALU module operation consists of:

- Selecting two sixteen-bit words, one from the A-input selector and one from the B-input selector.
- Combining the words based on the function code portion of the microinstruction (ALU control).
- Delivering the result to main memory, the interrupt system, I/O, panel interface, or one of several destination registers; the output also may be shifted by the output selector (S3).

This process is shown in figure 1-2.

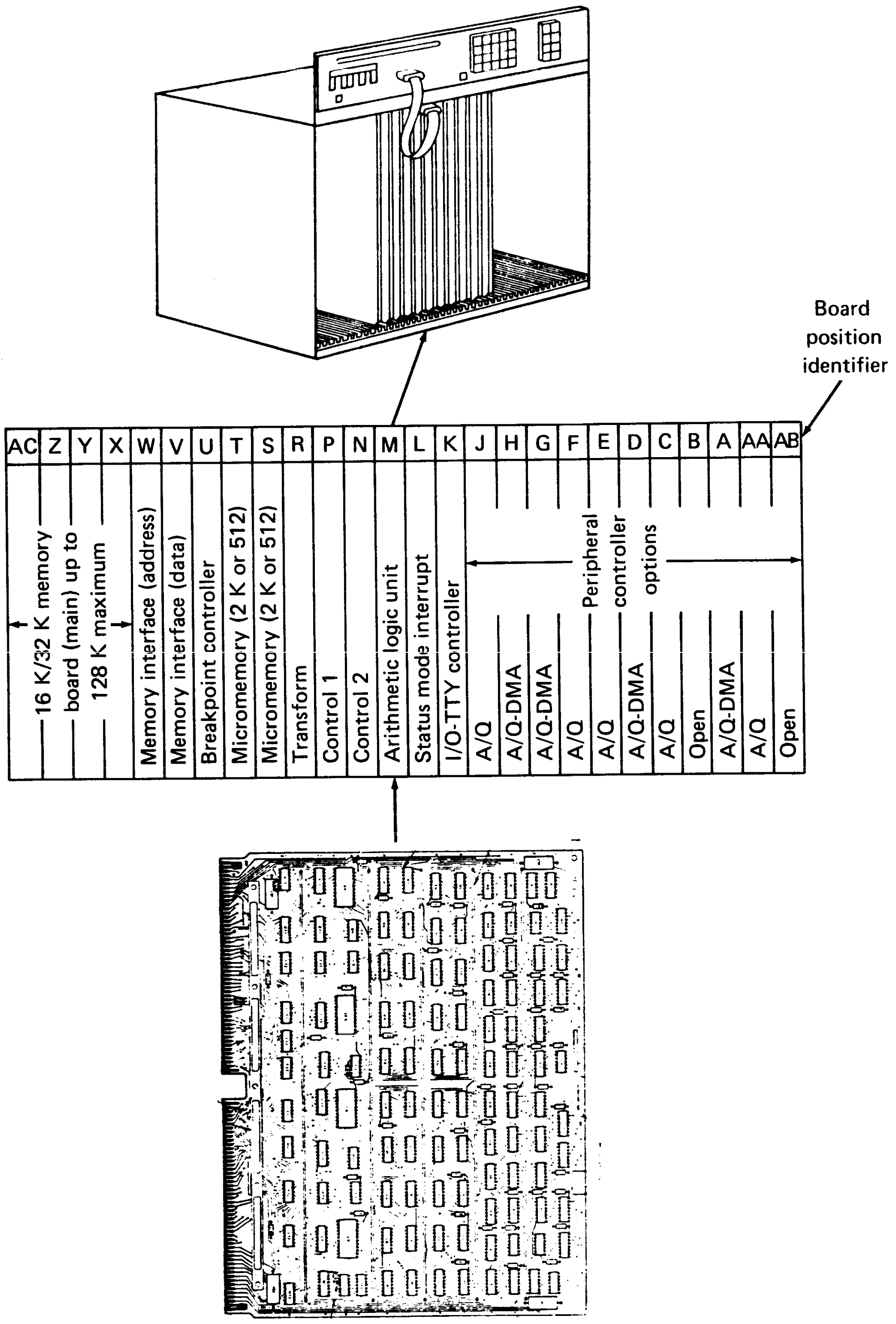


Figure 1-1. Location of ALU in MP Chassis

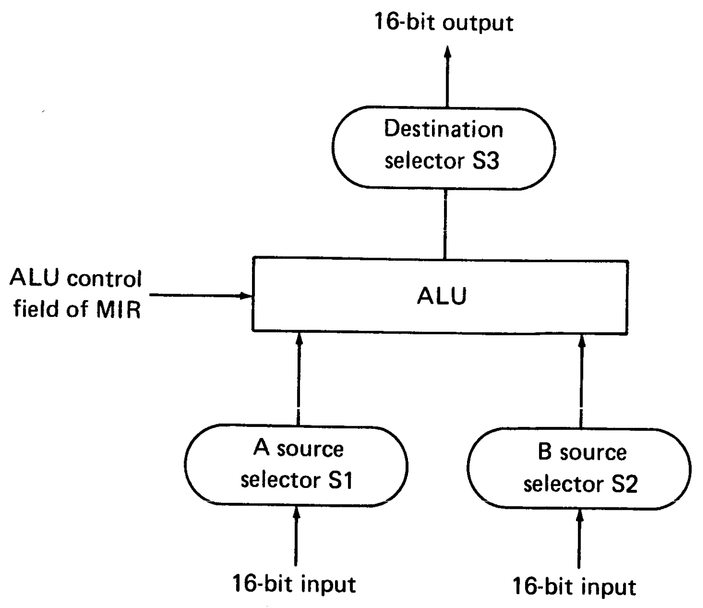


Figure 1-2. ALU Module Operation

Components of Logic Board*

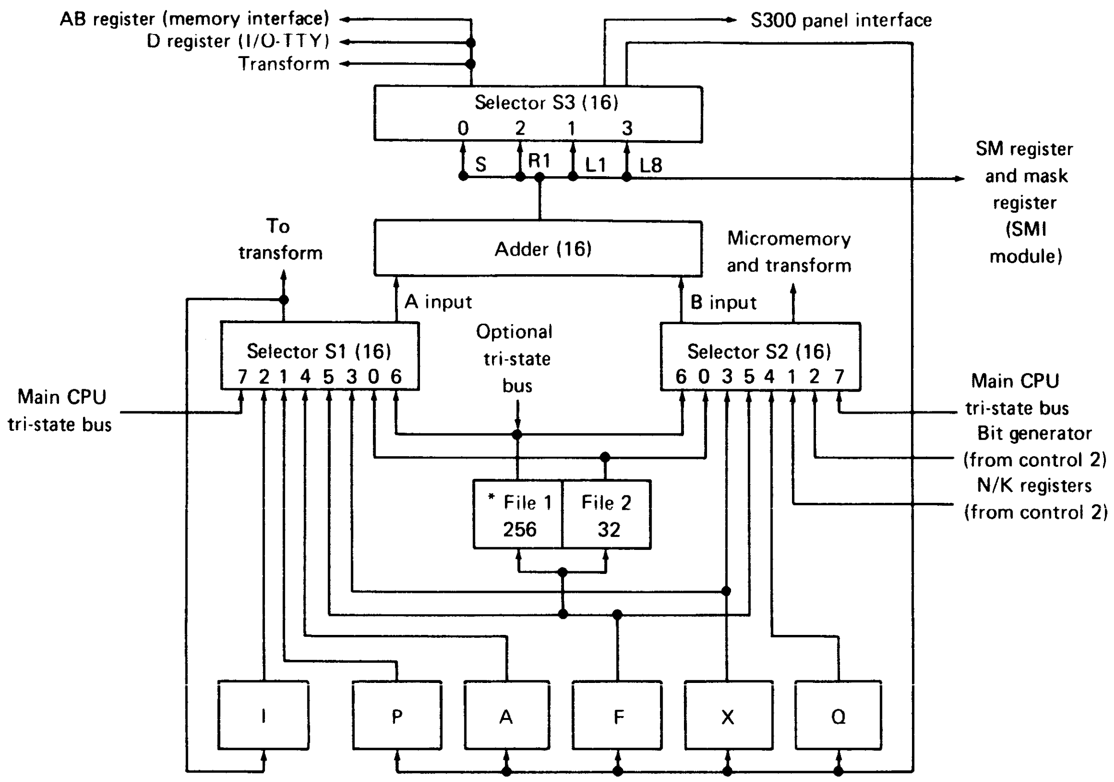
Six working registers and two files provide temporary storage of sixteen-bit words within the microprocessor as shown in figure 1-3.

Any of the registers or files can be selected as a sixteen-bit input to the ALU. With the exception of I, the registers can also receive the results of an ALU operation, if the microinstruction so specifies. ALU operation results can also be sent to external equipment or memory.

The following describes the functional areas of the ALU (as shown in figure 1-4):

- ***Selectors S1 and S2*** provide the controlled gating of sixteen-bit words to the ALU from various data sources within the processor.
- ***Selector S3*** provides controlled gating for the sixteen-bit output of the ALU to a number of possible destinations within the processor.
- ***ALU*** performs all arithmetic and logic operations.

* A distinction should be made between the ALU logic module and the ALU. The ALU module refers to all logic functions contained on the ALU logic board. The ALU refers to the four integrated circuit chips which perform the arithmetic and logic operations. It will be referred to as the adder in this text.



* File 1 is optional

Notes:

1. The numbers inside the selector blocks indicate the selector position.
2. The numbers in parentheses indicate the width of registers and selectors.

Figure 1-3. ALU Module Block Diagram

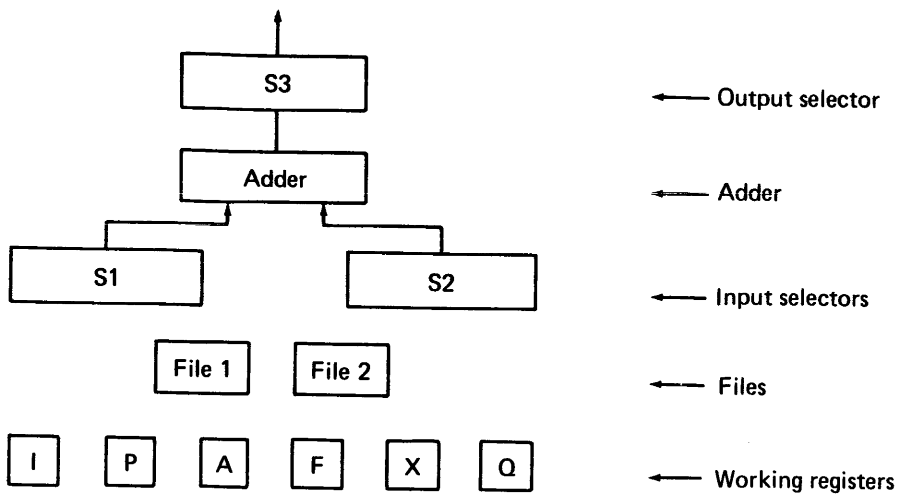


Figure 1-4. ALU Functional Areas

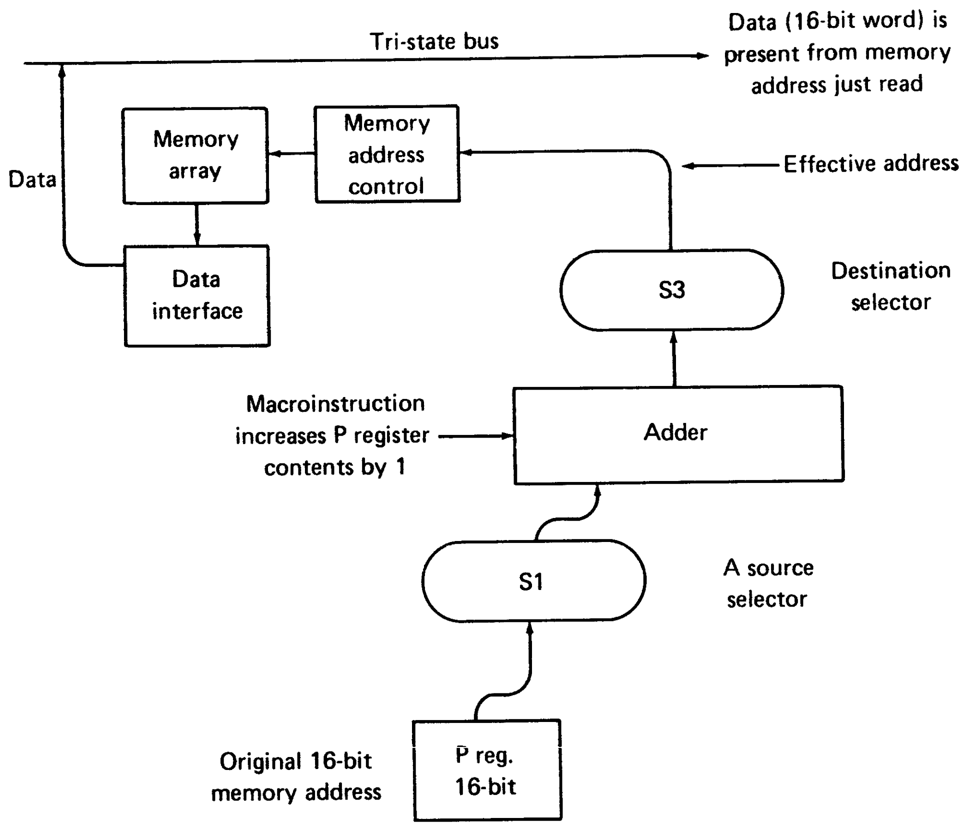
- ***The working registers*** (I, P, A, F, X and Q) provide temporary storage of data to and from the ALU and other areas of the microprocessor organization.
- ***File 1 (256 words) and file 2 (32 words)*** provide storage of sixteen-bit words. File 1 is addressed by the K register and file 2 is addressed by the N register.

Typical Data Flow

The contents of the ALU control field of the microinstruction being executed determine the general data flow through the ALU. A microinstruction, containing ALU control information, is encoded from a 1700 instruction by transform. After the microinstruction is formed it will be executed to perform an arithmetic or logic operation. For example, after being encoded into the microinstruction register, a 1700-type register reference instruction causes the contents of a working register in the ALU to be read, modified, or replaced. Two examples show data paths that can be taken with a typical 1700 instruction emulation.

Example 1

A 1700 add to A register operation (F = 8) requires that the sixteen-bit word contained in the A register be added to a sixteen-bit word from a specified memory location. Two steps take place in this operation–address modification and execution. Since this is a storage reference instruction, the address in memory from which the data is read must be formed. Figure 1-5 represents this first step. Assume that (P + 1) is the operand.



Note:

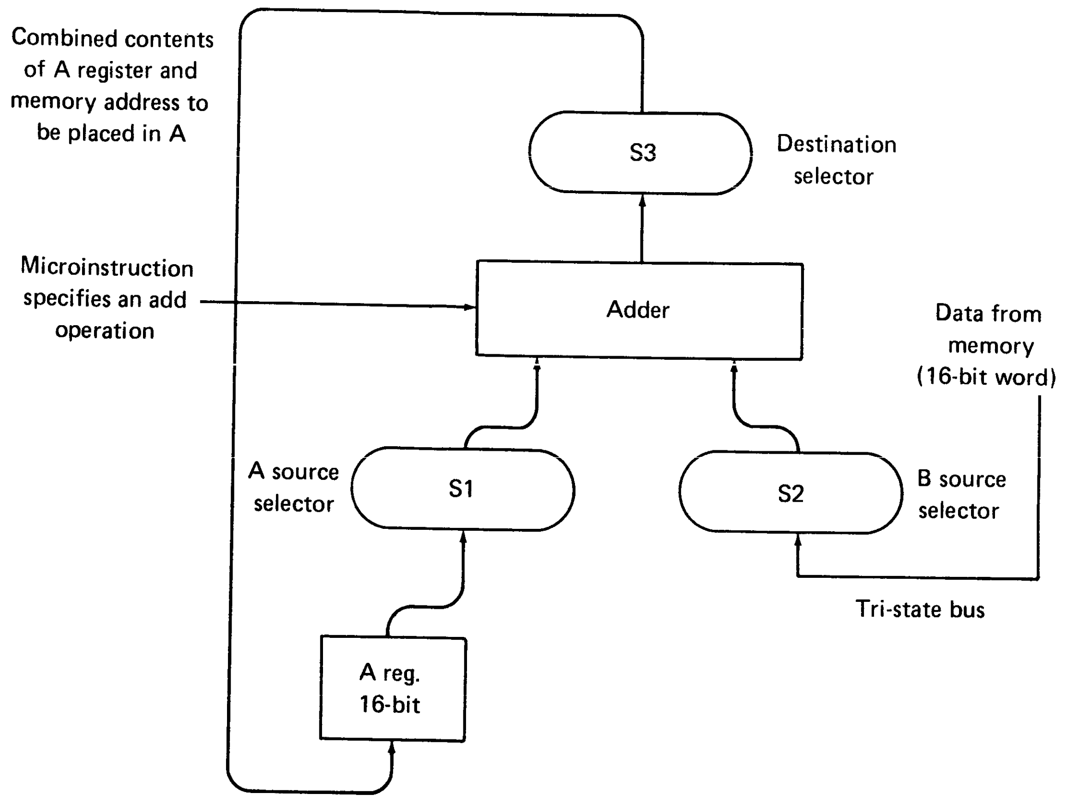
1. Increment the contents of the P register, and use this as the memory address.

Figure 1-5. Memory Address Formation

The contents of the P register are selected by S1 and gated into the adder, to be incremented by 1 through microinstruction control. The new address is transferred through the output selector to memory address control. The contents of the specified memory location is read and made available to the tri-state bus. Figure 1-6 shows the second step of the add to A register operation.

The contents of memory must be added to the contents of the A register by combining the two sources in the adder. The result is gated through the output selector back to the A register. The A register now contains the result of adding its original contents to the contents of the memory address read.

On completion of the add instruction, microprogramming jumps to read next instruction (RNI) programming. The next 1700-type instruction is read from memory and transform processing begins.


Combined contents of A register and memory address to be placed in A
S3
Destination selector
Microinstruction specifies an add operation
Adder
Data from memory (16-bit word)
A source selector
S1
S2
B source selector
Tri-state bus
A reg. 16-bit

Note:
1. Data from memory is added to contents of A register and placed back into A register.

Figure 1-6. Memory Address Contents Added to A Register Contents

Example 2

Figure 1-7 shows another example of data flow. A 1700 increase A instruction (F = 0, F1 = 9) is demonstrated.

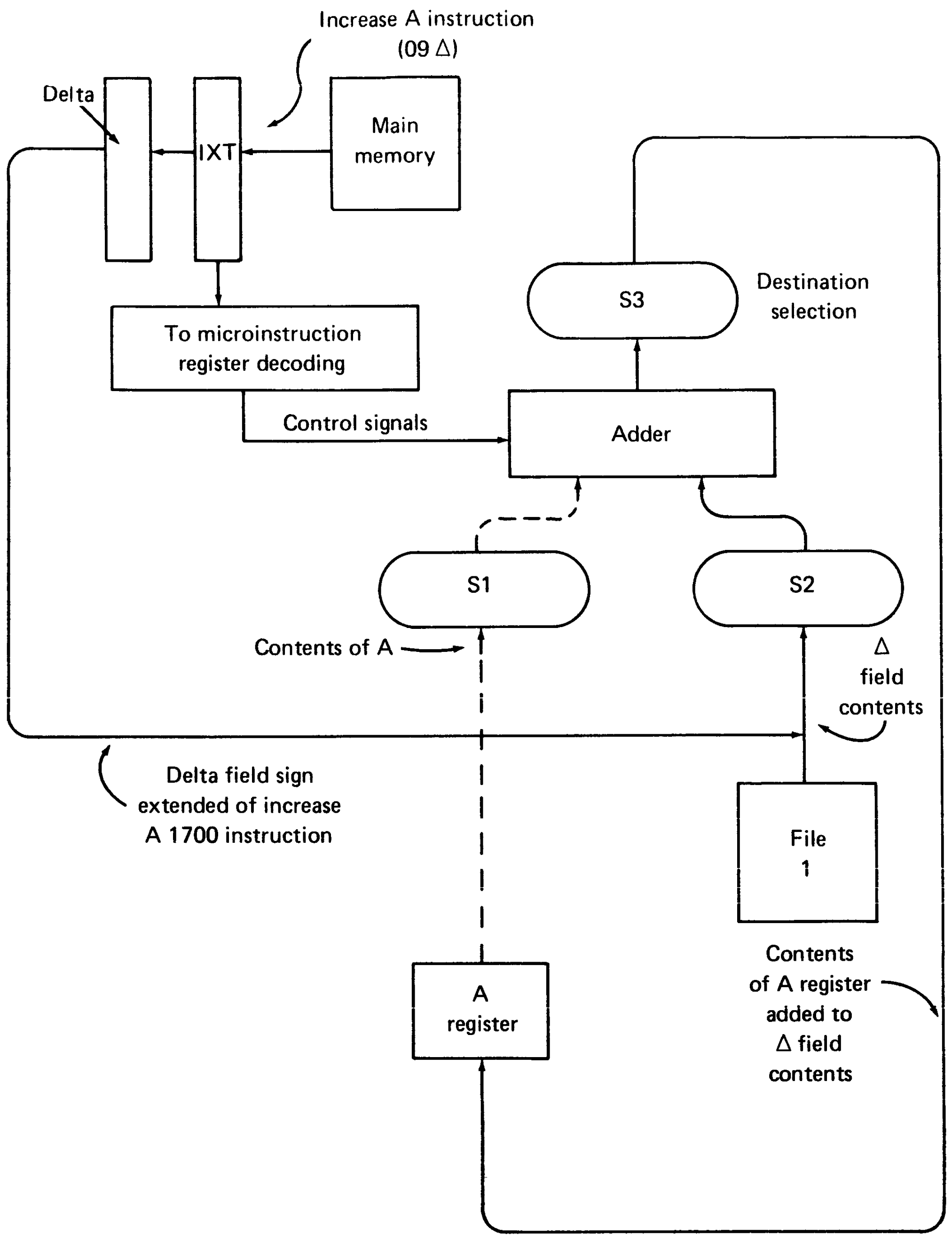

Increase A instruction
(09 Δ)
Delta
IXT
Main memory
To microinstruction register decoding
Control signals
Adder
S3
Destination selection
S1
S2
Contents of A
Δ field contents
Delta field sign extended of increase A 1700 instruction
File 1
A register
Contents of A register added to Δ field contents

Figure 1-7. Data Flow for Increase-A Instruction

This instruction causes the contents of the A register to be increased by whatever number value is contained in the delta field of the 1700 instruction. In the execution of the 1700 increase A instruction, the instruction is first read from memory and then decoded, using transform. The delta field of the instruction is transferred through the delta translator and made available to the shared output lines of file 1. Once the microinstruction has been formed, it causes the contents of the A register to be added to the delta field contents. The result is immediately available at the output selector, where it is gated back to the A register. The A register now contains its original contents plus the △ field contents of the increase A 1700 instruction. Again, RNI programming is entered and the next 1700 instruction processing begins.

Summary

Table 1-1 summarizes the information presented in this reading.

TABLE 1-1
Summary of ALU Functional Areas

Location	Purpose	Components	Data Flow
Slot M of microprocessor chassis	Data transfers Arithmetic and logic functions, i.e., addition, subtraction, overflow, logical comparisons, sign checks	• Selectors S1, S2, S3 • Adder • Working registers (I, P, F, X, A, Q) • File 1, file 2	Based on the macroinstruction

ALU Working Registers

Let's review some of what you know about the CYBER 18. You know that control provides the sequencing of events necessary to execute instructions, that main memory provides the storage necessary for 1700-type instructions and the data needed to execute a program, and that the ALU handles all arithmetic and logic operations and data transfer within the processor organization. This reading focuses on the registers used in ALU operations.

Register Descriptions

Registers in the ALU allow the main program to transfer sixteen-bit words from one area of the processor, such as memory, to another, or from the processor organization to some peripheral device. These registers also can contain the operands for arithmetic, logic, or shift operations. There are six such registers; each is sixteen bits wide, and is called a working register. (Refer to figure 1-8.)

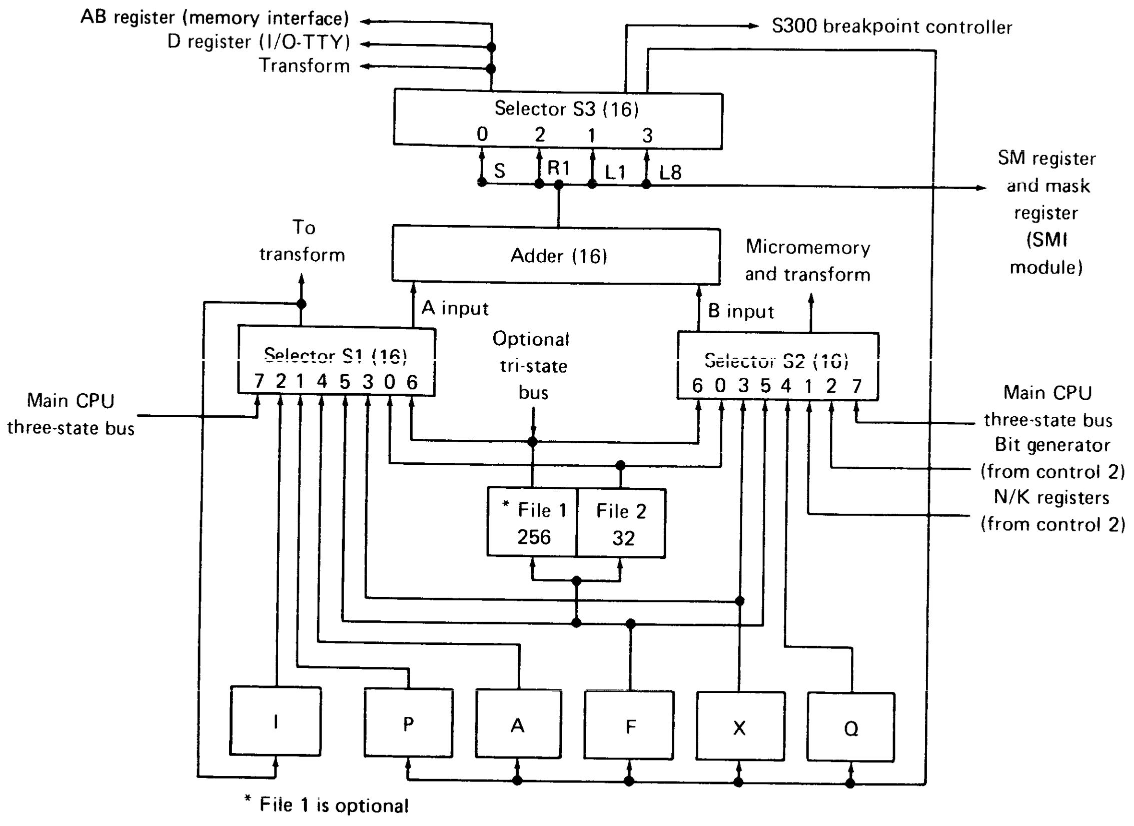
The ALU contains additional storage in the form of two file registers, which are actually small memories consisting of addressable RAMs. These file registers (file 1 and file 2) store sixteen-bit data words or hold constants (a constant is information assumed to be fixed or invariable that may be referenced in a given operation or calculation).

Figure 1-8 shows the location of each register in the ALU organization. The following describes each register and the two files. When the microprocessor acts as a 1700 emulator, these registers are defined as follows:

I Register. Input to register I comes directly from the output of selector S1. This selector enables data on the tri-state bus to be stored directly in register I and simultaneously input to the adder for some other operation. This operation is particularly useful in configurations using macromemory. The sixteen-bit output of the I register is available to selector S1.

P Register. This sixteen-bit register receives data from selector S3 and output to the A input of the adder via S1. In computer emulation configurations, it normally contains the macroprogram instruction counter (1700 P register).

A Register. The sixteen-bit A register may be used for data shifts, either by itself or in conjunction with the Q register as a double-length shift register. The shift function is independent of the adder and S3. This general-purpose register inputs from S3 and outputs to the A input of the adder via S1. This register is used as the 1700 A register.



* File 1 is optional

Notes:

1. The numbers inside the selector blocks indicate the selector position.
2. The numbers in parentheses indicate the width of registers and selectors.

Figure 1-8. ALU Module Block Diagram

F Register. This sixteen-bit, general purpose register receives its input from selector S3 and outputs to the adder through either selector S1 or selector S2. Selector S1 takes data contained in the F register and places it into the adder using input A, whereas selector S2 will cause the B input to the adder to be used. The F register also serves as an entry register for the File 1 or File 2 registers when they have been selected as destination registers of an ALU operation.

X Register. The X register is a sixteen-bit, general purpose register that receives its information through selector S3. The X register output is gated through selector S1 to the A input of the adder and through selector S2 to the B input of the adder.

Q Register. The sixteen-bit Q register is a general purpose register used as an input to the adder via selector S2. The Q register may be shifted left or right independent of the adder during Q/A shifts. The A register will also be used in conjunction with a shifting operation during F, A, B field shift operations. This register is used as the 1700 Q register.

File 1 Register. This general-purpose word-length memory contains 256 sixteen-bit words addressed by the contents of the K register located on control 2. The output of File 1 shares a tri-state bus with the output of the delta translator or transform. A status mode bit selects either the file 1 output or the delta translator. The data is sent to adder input A via S1 and/or to input B via S2.

File 2 Register. File 2 contains thirty-two sixteen-bit words addressed by the lower five bits of the N register. It delivers its output to adder input B via S2 and/or to adder input A via S1. File 2 is intended as a source for constants, but may be used as a general purpose file. The function control register (FCR) is contained in two addresses of this file.

Summary

A summary of A, Q, P, I, X, and F register characteristics is found in table 1-2.

TABLE 1-2
A, Q, P, I, X and F Register Characteristics

Register	Input From	Output To	Capabilities	Characteristics and Related Functions
A	S3	S1	Shift right or left with or without Q register, independently of ALU	Hold LSB during double-word length shift with Q register
Q	S3	S2	Shift right or left with A register independently of ALU	Hold MSB during double length shift with A register Shift right or left in conjunction with destination register (A, P, X, F) through adder
P	S3	S1	Holding register	May hold the software instruction counter for emulation
I	S1	S1	Holding register	May hold the software instruction for emulation
X	S3	S1, S2	Holding register	May be used for transferring information to I/O section
F	S3	S1, S2 F1, F2	Holding register	Hold data being stored in F1 or F2

ALU Organization

DIRECTIONS: After each of the following statements, write the name of the ALU functional block to which it corresponds.

1. This functional block selects one sixteen-bit data source from eight possible data sources available as input to the B side of the adder:________________________.
2. This functional block normally controls macromemory addressing by acting as a program instruction counter:________________________.
3. Although not a register, this block can store up to 256 sixteen-bit words and is addressed by the K register located on the control 2 module: ________________________.
4. An adder output can be gated to one of five working registers from this functional block:________________________.
5. Constants normally are contained in this functional block of the ALU module: ________________________.
6. This functional block can shift the output data from the ALU: ________________________.
7. A data word from the tri-state bus can be gated into this functional block without passing through the adder:________________________.
8. In conjunction with the Q register, this functional block of ALU may be used as a double-length shift register:________________________.
9. The output of the N/K registers of control 2 may be used as a data input to the ALU. This functional block determines whether that data source will be gated to the ALU:________________________.
10. Data can be gated from the ALU organization to transform by these functional blocks:________________________.

ANSWERS

1. Selector S2 2. P Register 3. File 1 4. Selector S3 5. File 2 6. Selector S3
7. I Register 8. A Register 9. Selector S2 10. Selectors S1, S2, S3

ALU Operation

The heart of any computer is the part that does the computing—the arithmetic logic unit. This adder provides for all arithmetic and logic operations the computer performs. This text explains the basic construction and organization of the adder and the look-ahead carry generator. The path data takes as it passes through the adder will also be examined.

Organization of the ALU

The basic adder is represented in figure 1-9. Control signals decoded from the microinstruction F field control the adder and determine the type of arithmetic or logical operation to be performed. From selectors S1 and S2, the adder accepts two sixteen-bit data words on which the operation will occur. After the microinstruction function is determined, the two operands obtained from the selectors may be added, subtracted, logically compared, and so forth. The sixteen-bit result of this operation is available immediately to selector S3, the SMI module, or both.

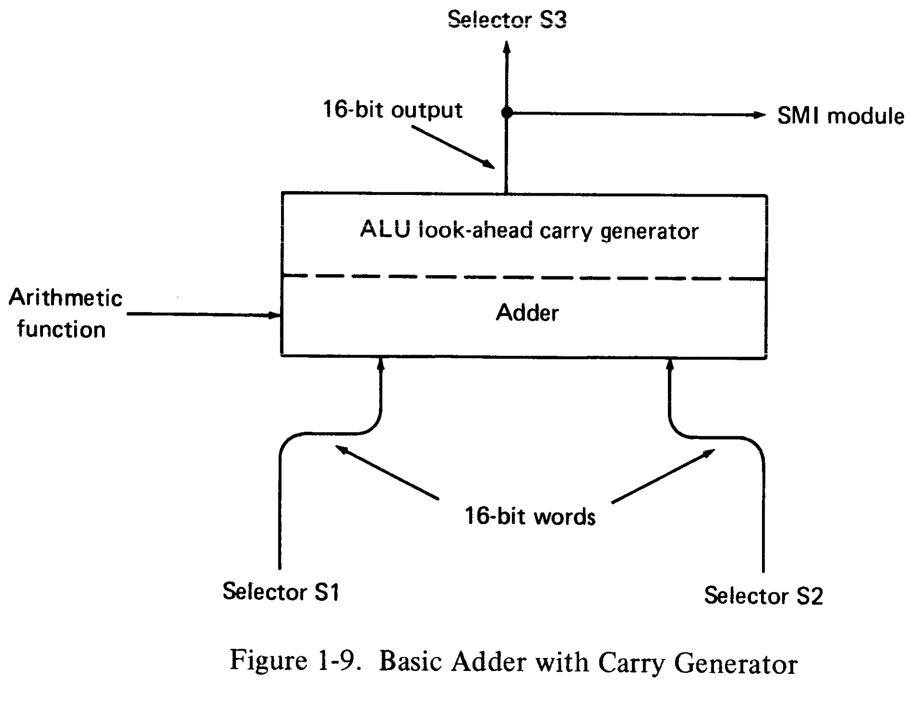


Figure 1-9. Basic Adder with Carry Generator

The complete adder consists of two logic blocks, the adder, consisting of four integrated circuit adder chips, and a look-ahead carry generator. (Refer to figure 1-10.) The carry generator monitors the four adder bit groups; each bit group is four bits wide. If a carry results from a four-bit group of the adder during an add, the carry generator is notified. This allows high speed carries between four-bit groups of the adder. The carry generator is not used in logic operations.

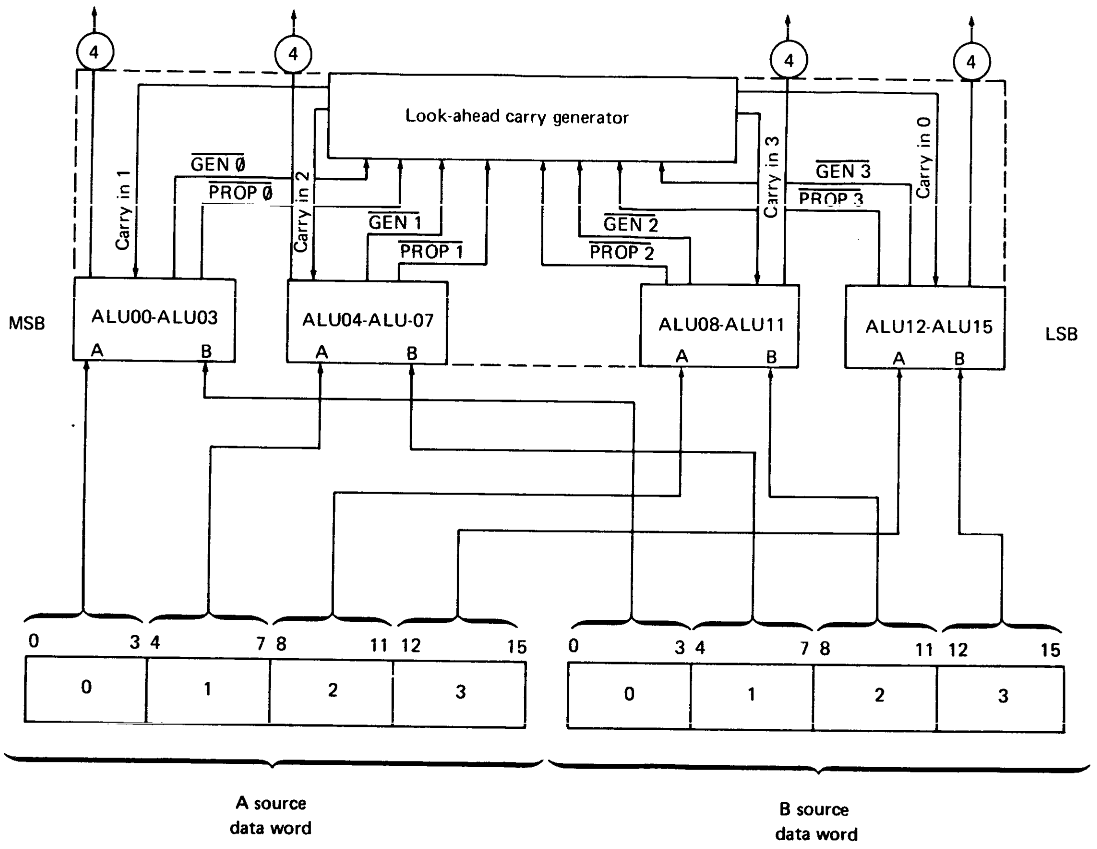


Figure 1-10. The Two Blocks of the ALU

Each of the four adder chips is able to perform arithmetic and logic operations on data from two separate four-bit data sources. In combination, the four chips provide arithmetic and logic operations on two sixteen-bit data input words. The adder chip used in the arithmetic unit is shown in figure 1-11.

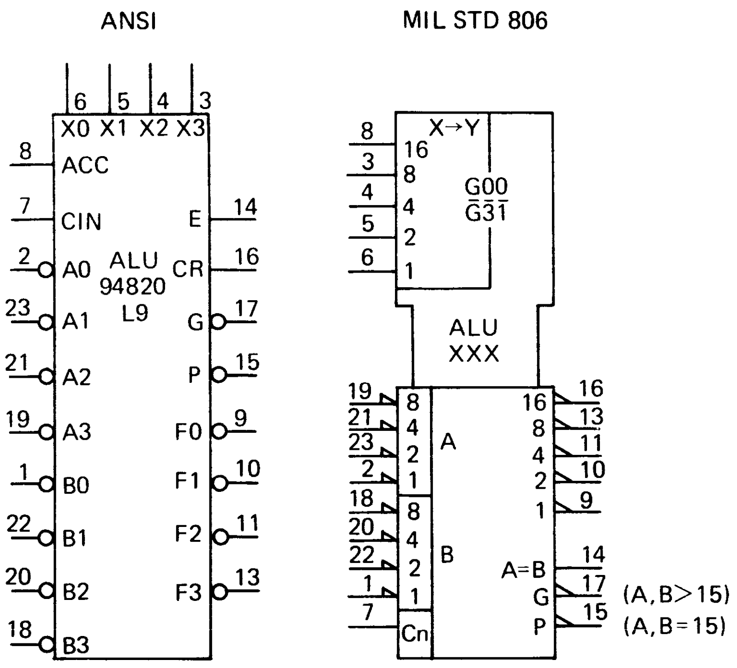


Figure 1-11. Four-Bit Adder

Table 1-3 shows the arithmetic and the logic operations that the adder chip can perform. As an example, when pin 8 (mode select) of the ALU chip is low, arithmetic operations are specified, and a high input causes logic operations to be performed. The definition of each signal into the ALU chip is as follows:

- ***CIN.*** If the preceding four-bit group generates a carry (G) output, a carry-in is brought through thc look-ahead circuit into the next higher adder chip.
- ***P.*** Propagate (pin 15) outputs if the result of an arithmetic operation performed by this chip does not allow a carry into the four-bit group this chip is monitoring.
- ***G.*** A generate signal indicates that the result of an arithmetic operation by this adder chip produced a carry-out condition.
- ***COUT.*** This signal is not used by the CYBER 18-20 system adder configuration.
- ***E.*** When the equality output is high, both the four-bit A input and the four-bit B input are equal in quantity.

Additional information on the adder chip is available in the logic circuits manual.

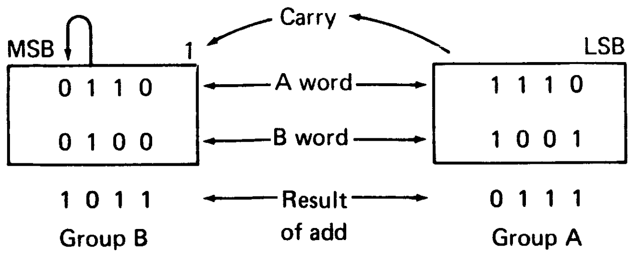
Look-Ahead Carry Generator

The look-ahead carry generator monitors each adder chip during an arithmetic operation to determine if a carry is to be generated to a particular four-bit group. The generator also determines whether the carry can be absorbed by the next four-bit data group of adder or must be passed to a subsequent group. The three conditions of the adder that are controlled by the look-ahead carry generator are as follows.

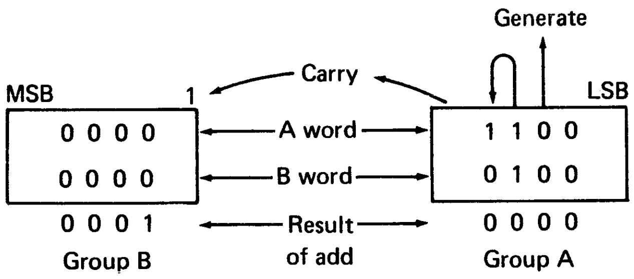
TABLE 1-3
ALU Function Selection

Function Select Signals				ALU Operation		
					Arithmetic ALUM = 0	
ALUS3-2 (pin 3)	ALUS2-2 (pin 4)	ALUS1-2 (pin 5)	ALUS0-2 (pin 6)	Logical ALUM = 1 (pin 8)	Cn = Low (No Carry)	Cn = High (No Carry)
0	0	0	0	$F = \overline{A}$	Not Used	
0	0	0	1	$F = \overline{AB}$		
0	0	1	0	$F = \overline{A} + B$		
0	0	1	1	$F = 1$		
0	1	0	0	$F = \overline{A + B}$		
0	1	0	1	$F = \overline{B}$		
0	1	1	0	$F = \overline{A \oplus B}$		
0	1	1	1	$F = A + \overline{B}$		
1	0	0	0	$F = \overline{A}B$		
1	0	0	1	$F = A \oplus B$		
1	0	1	0	$F = B$		
1	0	1	1	$F = A + B$		
1	1	0	0	$F = 0$		
1	1	0	1	$F = AB$	Not Used	
1	1	1	0	$F = AB$		
1	1	1	1	$F = A$		
0	1	1	0		$A - B - 1$	$A - B$
1	0	0	1		$A + B$	$A + B + 1$
1	1	1	1		A	$A + 1$

Carry. A carry is generated by a group of four bits when an arithmetic operation is performed and a logic 1 must be carried into the next group. For example:



Generate. A generate signal occurs when an arithmetic operation is performed resulting in a carry by a four-bit group that cannot be absorbed in that group. This signal is sent from the ALU chip to the carry generator. For example:



Propagate. A propagate signal is generated to the carry generator when an arithmetic operation results in all 1s and there is, therefore, no place for a carry to the group to be absorbed. Any carry into this group would be forced into the next group able to absorb the carry.

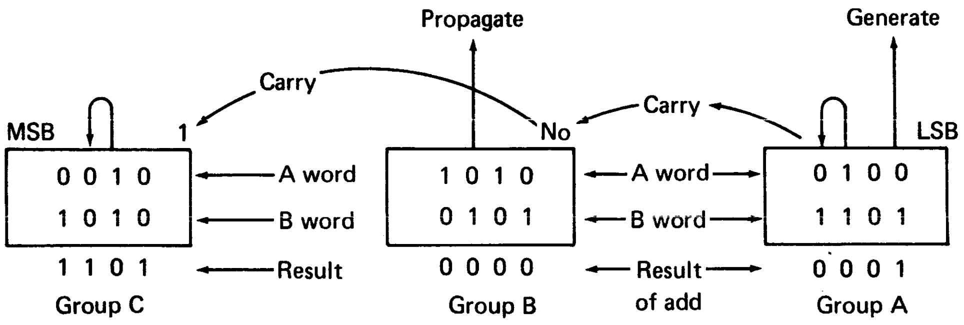


Figure 1-12 shows a typical add operation demonstrating the adder functions using look-ahead carry. In this example, a total of sixteen bits from two sources is added. The contents of each four-bit ALU chip is shown and the signals interfacing to the look-ahead carry generator are indicated.

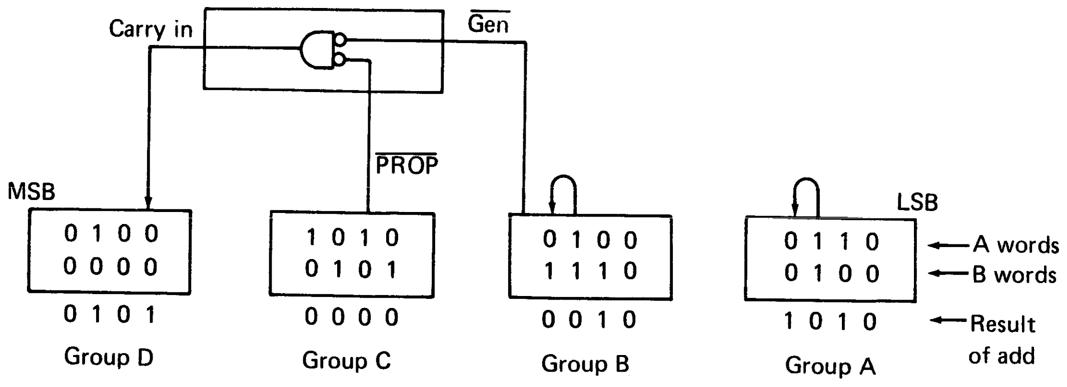


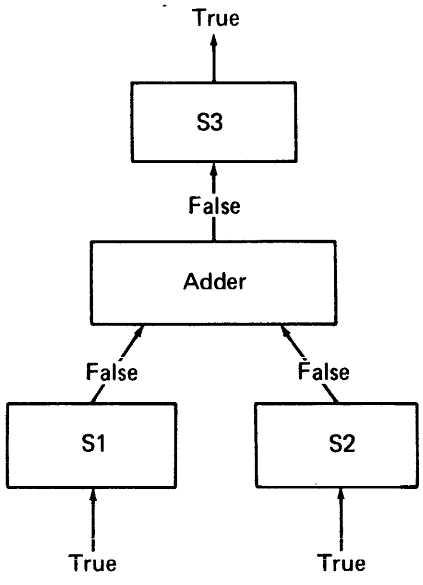
Figure 1-12. Typical Additive Operation Using Look-Ahead Carry

Each individual group above functions as follows:

- ***Group A.*** No carry is produced by the first group. Since a carry generated by this group is absorbed in the group, a generate is not produced. If a carry were brought into this group, it could be absorbed; therefore, a propagate signal would not originate from this group.
- ***Group B.*** A generate signal is produced by group B since a carry is produced. Normally, the carry would be carried to group C and absorbed.
- ***Group C.*** A carry from group B normally would be allowed into group C and absorbed. However, a carry cannot be absorbed by this group; this results in group C's producing a propagate signal, which in turn results in the carry's being forwarded to the next group able to absorb it.
- ***Group D.*** Group D absorbs the carry originated by group B. Since a carry can be absorbed by this group, a propagate signal will not be produced. Because no carry is generated by the group, no generate signal occurs.

Data Form

Data is not always in its true form as it passes through the ALU module. It is important to note where the data is in its true form and where it is false or in complement form. Figure 1-13 represents the general flow of data through the ALU section. Data is always true as it enters the input selectors from various microprocessor sources. Data is always inverted as it leaves the input selector; thus, it enters the adder as false or complemented data. The adder output data is also in its complemented form. The output is either made available to the status mode interrupt in this form or is recomplemented as it passes through selector S3 and is available in true or complemented form.



Note:

1. May substitute the word complement for false.

Figure 1-13. Data Inversion Through ALU

Summary

In basic terms, the adder consists of four integrated circuit adder chips and one look-ahead carry generator. Two sixteen-bit word inputs are available to the adder from selectors S1 and S2. The adder is controlled by a function input derived from the microinstruction in execution and produces an output to either the SMI module or selector S3.

The look-ahead generator monitors each adder chip to determine whether a carry is to be generated to a particular four-bit group. It controls three basic conditions: carry, generate, and propagate.

Data enters the adder and leaves it in its complemented form.

ALU Output Selector and Register Data Paths

By now, you know the purpose of most of the functional areas on the ALU module. One area still to be discussed is selector 3. This selector can both select the destination of the ALU output and shift the output right or left. This reading describes selector 3 and reviews the working areas of the ALU already studied.

ALU Review

The functional areas of the ALU discussed so far and their purposes are as follows:

- ***Working Registers.*** The working registers–six in all–are used for sixteen-bit word transfers through the microprocessor organization. In the ALU these registers are used as temporary storage registers to hold operands for, or the results of, an arithmetic operation. Two of these registers (A and Q) can shift their contents left or right either individually or in combination.
- ***File Registers.*** The two file registers (or, more accurately, memories) extend the storage capabilities of the ALU module. File 1 can store 256 sixteen-bit words, which are available to the ALU input selectors; File 2 can store 32 sixteen-bit words, which are generally provided as constants to the processor organization.
- ***Input Selectors S1 and S2.*** These selectors provide sixteen-bit inputs to the A and B inputs of the adder. The source of these inputs is controlled by the A and B fields of the microinstruction under execution.
- ***Adder*** Along with the look-ahead carry generator, the adder proper provides the arithmetic and logic capabilities for the microprocessor. Two sixteen-bit operands can be added, logically ANDed, ORed, or exclusive ORed. Data destined for I/O, memory, and most destination registers pass through adder.

Destination Selection and Shifting (S3)

The output of the adder can be gated directly to the status mode or mask registers of the status mode interrupt module, or it can be made available to some other destination through selector S3 of the ALU module. Selector S3 can transfer the adder output to a number of destinations within the microprocessor organization, and it can also shift that output right or left before the transfer.

The destinations for selector S3 output are as follows:

- Working registers A, P, X, F, or Q
- IXT register of transform

- Macromemory address and data logic
- I/O via the I/O-TTY module
- Panel interface

The destination of the output depends on the D, D', and D'' decoding of the micro-instruction D field. (Refer to figure 1-14.)

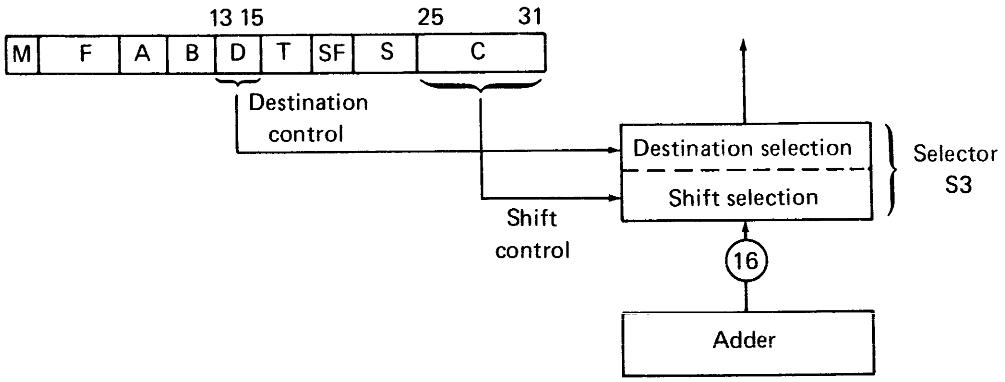


Figure 1-14. Selector S3 Control

Shifting of sixteen-bit words through the output selector is accomplished by a shift register that may cause data from the adder to be shifted left one place, right one place, left eight places end around, or straight through. The type of shifting operation that takes place is determined by C' decoding of the C field in the microinstruction being executed. Refer to table 1-4.

TABLE 1-4
C' Shift Formats*

C' Codes	Mnemonics	Actions
1 1 1 0 0 0 0 or 1 1 1 0 0 0 1	RQLXN	The Q register and one destination register (P, A, F, or X) provide a double-length register. The combined register is shifted left one bit position. The arithmetic logic unit sign bit complement is entered at the least significant bit of the Q register.
1 1 1 0 0 1 1	RQR1E	Shift the combined destination and Q registers right one bit; enter 1 (0) in the sign position of destination register.
1 1 1 0 0 1 0	RQR0E	Shift the combined destination and Q registers right one bit; enter 1 (0) in the sign position of destination register.
1 1 1 0 1 0 0	RL0E	Shift the destination register left one bit; enter 1 (0) in the lowest bit position.
1 1 1 0 1 0 1	RL1E	Shift the destination register left one bit; enter 1 (0) in the lowest bit position.
1 1 1 0 1 1 0	RR0E	Shift the destination register right one bit; enter 1 (0) in the lowest bit position.
1 1 0 1 1 1 1	RR1E	Shift the destination register right one bit; enter 1 (0) in the lowest bit position.

*Format 1, bit 19 = 0

Examples of typical operations that the ALU can perform using selector S3 follow.

Example A. The partial contents of the microinstruction register are shown in figure 1-15. The C' decoding of the C field indicates that the Q register and one destination (P, A, F, or X) specified by the A and D fields are used as a double length register. This combined register is shifted left one bit position. In the process, the least significant bit (LSB) of Q is replaced by the complement of the most significant bit (MSB) of adder. The MSB of the P register ends off into the bit bucket. Figure 1-15 shows an example of the data contained in P and Q before and after shifting.

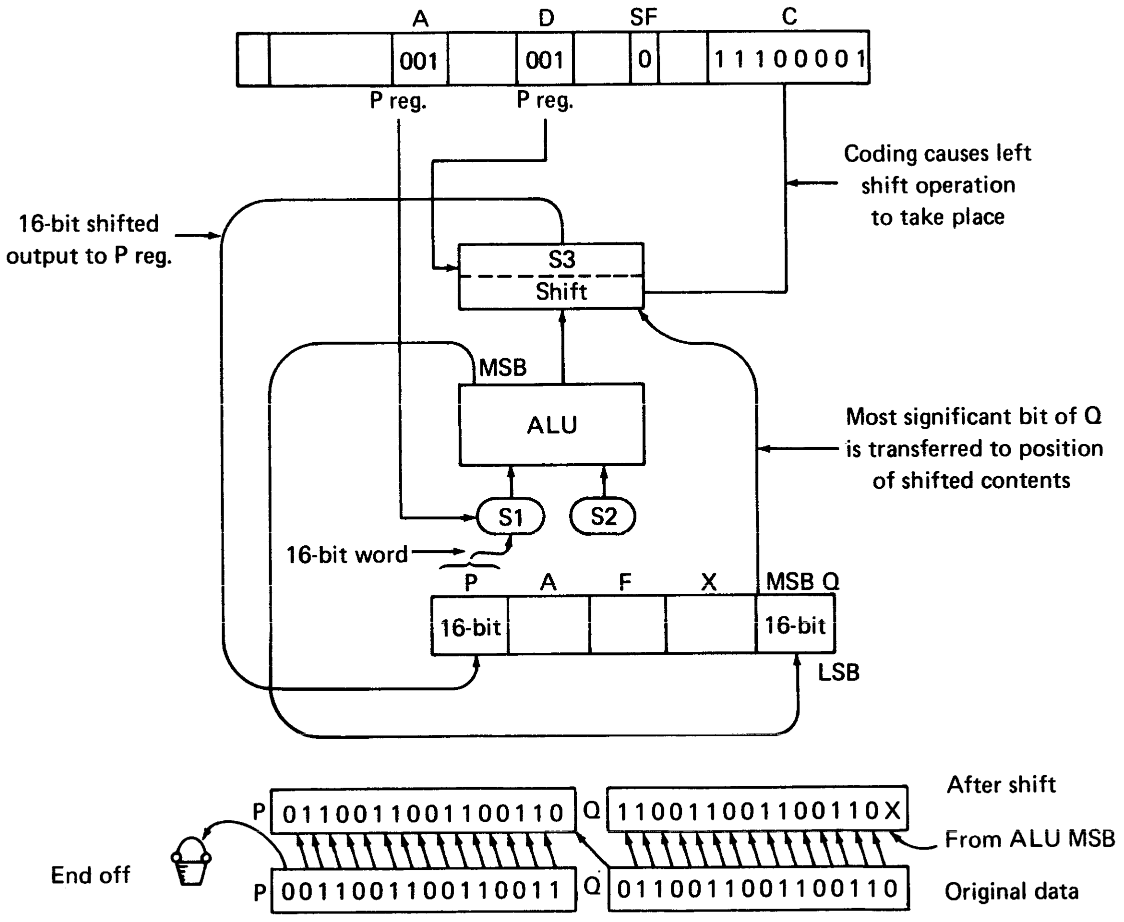


Figure 1-15. Double-Length Register Shift (P and Q)

Example B. Another example of a shift instruction is shown in figure 1-16. The contents of the microinstruction register shown specifies that the destination register called for by the A and D field is shifted right one bit position. In the example, the X register is shifted and a 1 or 0 is entered in the most significant bit position of X. The bit originates from MIR31 of the microinstruction register. A right shift causes the LSB of the sixteen-bit operand to be ended off into the bit bucket.

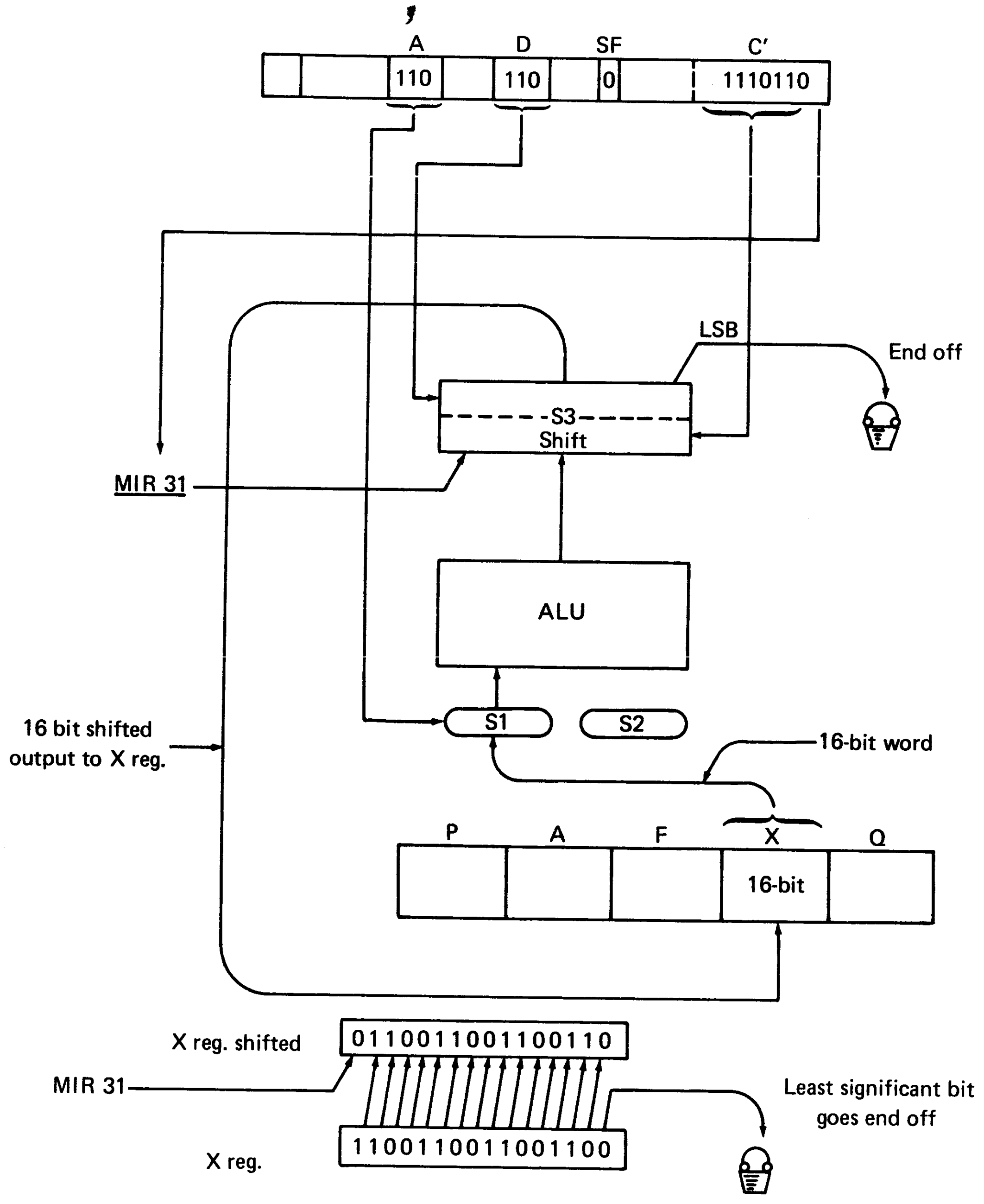


Figure 1-16. Single Register Right Shift

Example C. A straight transfer without a shift results if the C field does not contain the conditions shown in table 1-4. Figure 1-17 shows the partial contents of a micro-instruction specifying a straight transfer. This arithmetic field calls for an add operation of two operands. One is contained in the F register and the other is an address of file 2. The address of file 2 is specified by the N register located on the control 2 module. Through selector S3, the result of the add is transferred both to the X register and the AB interface of macromemory. The data flow through the ALU module is shown.

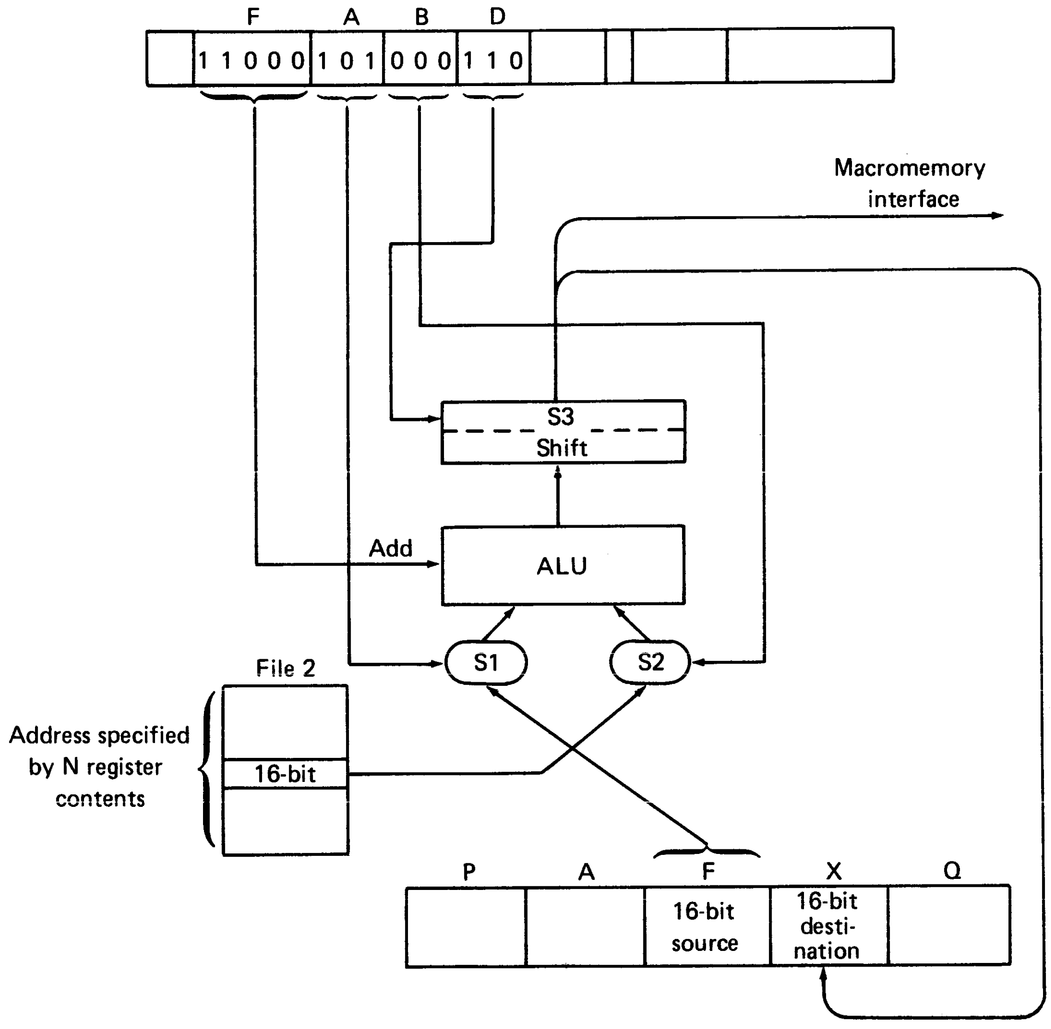


Figure 1-17. Data Add and Transfer Paths

Summary

Selector S3 of the ALU module transfers the adder output to its destination without a shift or shifts that output left or right before the transfer. The D field of the micro-instruction controls the destination; the C field controls the shifting.

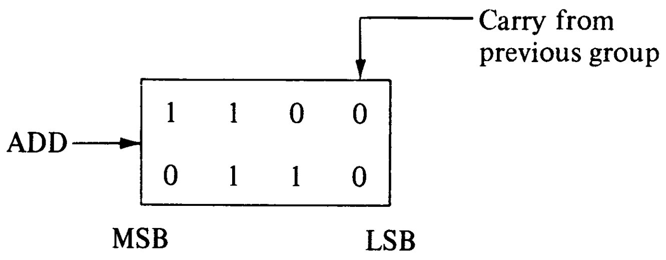
ALU Register Data Paths

DIRECTIONS: Answer each question as indicated.

1. Describe the purposes of the look-ahead carry generator associated with the ALU.

   ____

2. The four-bit add operation performed below would result in the following (circle the letters of ALL answers that apply):



   a. A propagate signal being generated
   b. A generate signal being generated
   c. A carry-in being absorbed from previous group

3. Which field of the microinstruction controls one bit position shifting of the sixteen-bit data output from ALU?__________.

DIRECTIONS: Mark the following statements T for true or F for false.

________ 4. An adder chip is capable of performing arithmetic and logic operations on sixteen-bit operands from two separate data sources.

________ 5. Shifting that must be performed as part of a multiply or divide operation is done by output selector S3.

________ 6. Selector S3 can gate a sixteen-bit output to all of the working registers located on the ALU module.

ANSWERS

1. To detect a carry being generated by a four-bit group, to detect a propagate signal from a group, to pass a carry from one group to the next. 2. b and c 3. C field
4. F 5. T 6. F

PROGRESS CHECK

QUESTIONS

1. The ______ register <u>cannot</u> receive the result of an ALU operation.

   a. A
   b. I
   c. Q
   d. X

2. Which working register supplies its output to both selectors S1 and S2?

   a. A register
   b. Q register
   c. P register
   d. X register

3. How many adder chips are contained in the CYBER 18-20 ALU?

   a. 2
   b. 4
   c. 8
   d. 16

4. What microinstruction field controls a shift operation at selector S3?

   a. A
   b. B
   c. C
   d. D

5. What ALU logic function gates the result of an arithmetic operation to its destination?

   a. Selector S1
   b. Selector S2
   c. Selector S3
   d. File 2

6. What working register receives information directly from the tri-state bus without the information first passing through the adder?

   a. I register
   b. F register
   c. Q register
   d. X register

7. What pin on the adder chip selects whether an arithmetic or logic operation is performed?

   a. 8
   b. 17
   c. 7
   d. 15

8. The ______ and ______ registers may be used for shifting data.

   a. A, Q
   b. P, I
   c. I, X
   d. F, P

9. The function of the file 2 register is to ______.

   a. hold 256 sixteen-bit words
   b. store contents
   c. be used as an adder
   d. be used as a shift register

10. The output selector S3 can be ______.

   a. shifted by control signals originating from the ALU control field of the microinstruction
   b. gated to registers on the SMI module
   c. shifted using control signals decoded from the microinstruction C field
   d. gated to any of six working registers contained on the ALU module

ANSWERS

1.	Correct Answer:	b	
	Resource:	Text:	<u>CYBER 18-20 Arithmetic Logic</u> SRM, page 1-3.
2.	Correct Answer:	d	
	Resource:	Text:	<u>CYBER 18-20 Arithmetic Logic</u> SRM, page 1-11.
3.	Correct Answer:	b	
	Resource:	Text:	<u>CYBER 18-20 Arithmetic Logic</u> SRM, page 1-15.
4.	Correct Answer:	c	
	Resource:	Text:	<u>CYBER 18-20 Arithmetic Logic</u> SRM, pages 1-22 and 1-23.
5.	Correct Answer:	c	
	Resource:	Text:	<u>CYBER 18-20 Arithmetic Logic</u> SRM, page 1-21.
6.	Correct Answer:	a	
	Resource:	Text:	<u>CYBER 18-20 Arithmetic Logic</u> SRM, pages 1-10 and 1-11.
7.	Correct Answer:	a	
	Resource:	Text:	<u>CYBER 18-20 Arithmetic Logic</u> SRM, page 1-16.
8.	Correct Answer:	a	
	Resource:	Text:	<u>CYBER 18-20 Arithmetic Logic</u> SRM, page 1-10.
9.	Correct Answer:	b	
	Resource:	Text:	<u>CYBER 18-20 Arithmetic Logic</u> SRM, page 1-12.
10.	Correct Answer:	a	
	Resource:	Text:	<u>CYBER 18-20 Arithmetic Logic</u> SRM, page 1-22.

Block 2

Microinstruction Decoding

ALU Control Field Functions

Each field of a microinstruction provides part of the overall control for the microprocessor. This reading explains the ALU control operations contained in bits 2 through 15 of the microinstruction.

The ALU Control Field

In the CYBER 18, the contents of ALU control field are initially encoded from the 1700 instruction presently being executed (see figure 2-1). The ALU control field consists of four subfields (see figure 2-2).

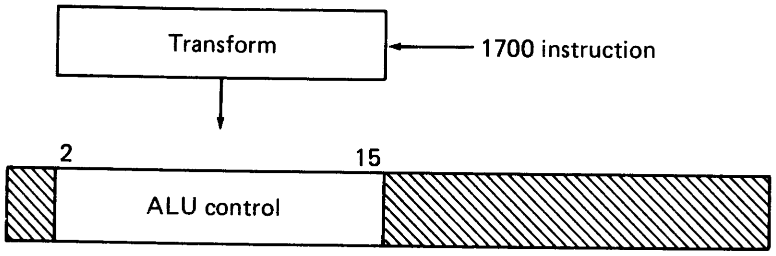


Figure 2-1. Encoding of ALU Control Field

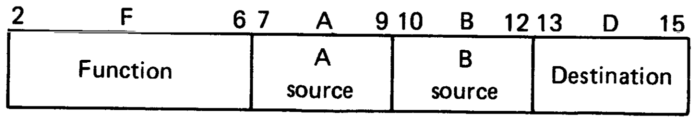


Figure 2-2. ALU Control Subfields

The purpose of each field for arithmetic and logic operations is as follows:

- F operation or function
- A A source of operand
- B B source of operand
- D Destination of operand

The ALU control field also can perform shift and scale operations. In such instances, the ALU control field format is that shown in figure 2-3.

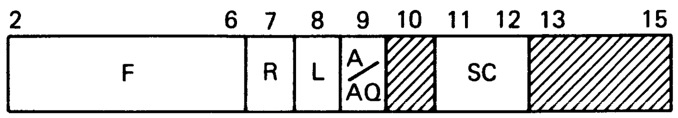


Figure 2-3. ALU Control Field Format for Shift and Scale Operations

The purpose of each subfield of ALU control for a shift or scale operation is:

- F Operation (shift or scale)
- R Right shift
- L Left shift
- A A-register shift
- A/Q A- and Q-register shift*
- SC Shift control

Arithmetic and Logical Control—F Field

The F field is five bits in length and controls the adder itself (see figure 2-4).

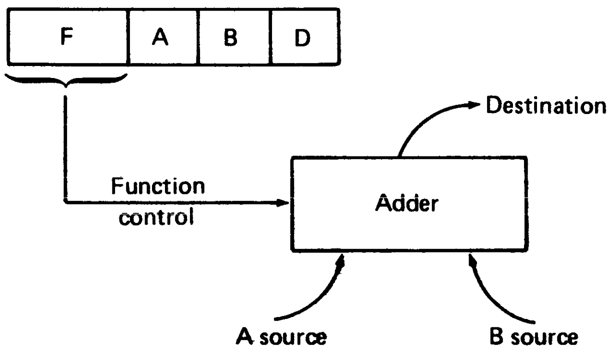


Figure 2-4. The ALU F Field

Arithmetic or logic control operations are controlled by code in the F field. Table 2-1 represents all arithmetic operations controlled by the F field, while table 2-2 represents all logic operations. In table 2-2, -B means the same as $\overline{B}$.

* Note: Macro shift instructions should not be confused with micro shift instructions. When A and Q registers are shifted with the macroinstruction, the most significant bit is in Q and the least significant bit is in A (Q/A). The micro level shift holds the most significant bit in A and the least significant bit in Q (A/Q).

TABLE 2-1
Arithmetic Operations

F Codes	Mnemonics	Operations
1 0 1 0 0	SUB	Subtract B input from A input.
1 1 0 0 0	ADD	Add A and B inputs.
1 0 1 0 1	SUBT	Subtract with an overflow test.
1 1 0 0 1	ADDT	Add with an overflow test.
1 0 1 1 0	SUB–*	Perform A–B–1 input (two's complement only).
	SUB–C*	A–B with forced carry-in (one's complement only).
1 1 0 1 0	ADD+*	Perform A+B+1 (forced carry-in).
1 0 1 1 1	SUB–T	Perform SUB– with an overflow test (two's complement only).
	SUB–TC*	A–B with forced carry-in (one's complement only).
1 1 0 1 1	ADD+T*	Perform ADD+ with an overflow test.

*If split adder mode is selected, the most significant (upper) adder performs an ADD or SUB without forced carry-in. Forced carry-in is defined as an unconditional hardware logical 1 (used for two's-complement arithmetic) gated to the (hardware) adder carry-in input.

TABLE 2-2
Logic Operations

F Codes	Mnemonics	A input 0011 Bit Result B input 0101
0 1 1 0 0	ZERO	0 0 0 0
0 1 1 1 0	A ● B	0 0 0 1
0 1 1 0 1	A ● –B	0 0 1 0
0 1 1 1 1	A	0 0 1 1
0 1 0 0 0	–A ● B	0 1 0 0
0 1 0 1 0	B	0 1 0 1
0 1 0 0 1	EOR	0 1 1 0
0 1 0 1 1	A+B	0 1 1 1
0 0 1 0 0	–A ● –B	1 0 0 0
0 0 1 1 0	–EOR	1 0 0 1
0 0 1 0 1	–B	1 0 1 0
0 0 1 1 1	A+ –B	1 0 1 1
0 0 0 0 0	–A	1 1 0 0
0 0 0 1 0	–A+B	1 1 0 1
0 0 0 0 1	–A+ –B	1 1 1 0
0 0 0 1 1	ONE	1 1 1 1

The arithmetic operations listed in table 2-1 operate on single precision operands, or sixteen-bit data words and use the ALU module previously described. An option is available to add a second ALU module to perform double precision arithmetic. This means the computer would be able to handle two sixteen-bit words together as though they were a single 32-bit word. This option is not present in the CYBER 18 configuration being used. An additional option present in the CYBER 18 is coded into F field and is called an overflow test, that is, the sign bit of the two inputs to the ALU is compared with the sign of the result. If the sign has changed an error has occurred which sets a status mode bit, indicating that the result is not consistent. The status mode overflow bit is set to 1 when an overflow occurs and must be reset to 0 by a microinstruction.

A Field

During arithmetic or logic operations, A-field bits 7 through 9 specify the input to selector S1 and thus the A side of the adder. (See figure 2-5.)

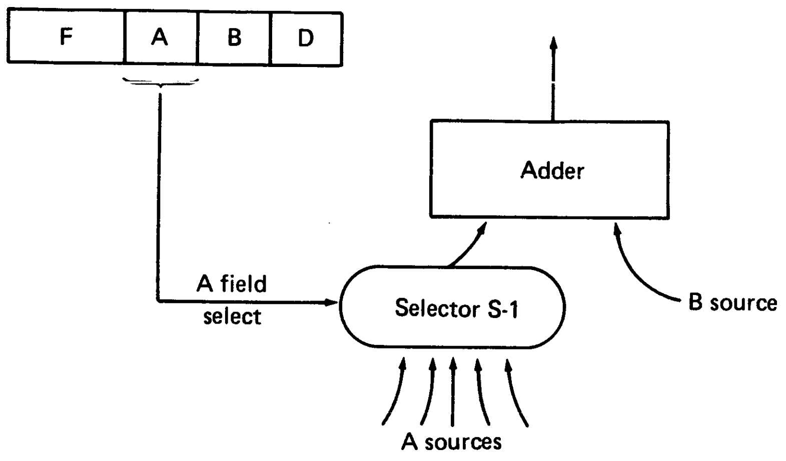


Figure 2-5. The Adder A Field

Table 2-3 contains the A codes and their descriptions.

TABLE 2-3
A-Input Operations

A Codes	Mnemonics	Operations
000	F2*	Use the contents of the file 2 register as the A source input. The current value of the N register is used to address register file 2. If the value of N is changed in the current microinstruction, its initial value is used to reference the file register. F2 must not have been written during the previous instructions.
001	P	Use the contents of the P register as the A source.
010	I	Use the contents of the I register as the A source.
011	X	Use the contents of the X register as the A source.
100	A	Use the contents of the A register as the A source.
101	F	Use the contents of the F register as the A source.
110	F1*	Use the contents of the optional file 1 register or external source as the A source. The current value of the K register is used to address register file 1. If the value of K is changed in the current microinstruction, the initial value of K is used to reference the file register. SM111 controls the selection of F1/external. F1 must not have been written during the preceding microinstruction.
111	MEM**	Obtain data read from macromemory and use it as the A source.

*Restriction: The value of the addressing register (N or K) cannot have been modified by a C' increment or decrement command in the preceding microinstruction.
**Restriction: If the macromemory READ command was not given in the preceding microinstruction, all 1's are input to the A source. If the B source is a prime code, the B source data also is input to S1. This command is restricted to a microinstruction, with type A, B, or C execution time.

The A field also can have a different meaning if the S field of the microinstruction equals 1010 or 0111. This causes A' coding to be interpreted as shown in table 2-4.

TABLE 2-4
A' Input Operations

A' Codes	Mnemonics	Operations
000	SM1	Use the contents of SM register 1 as the A source.
001	M1	Use the contents of interrupt mask register 1 as the A source.
010	SM2	Use the contents of SM register 2 as the A source.
011	M2	Use the contents of interrupt mask register 2 as the A source input.
100	A*R8	Use the contents of the double-precision A register, shifted right eight bits with end-around carry, as the A source. The A* register remains unshifted.
101	A*	Use the contents of the double-precision A* register as the A source.
110	X*	Use the contents of the double-precision X* register as the A source.
111	Q*	Use the contents of the double-precision Q* register as the A source.

*The A' codes are specified by the S field equal to 0 1 1 1 or 1 0 1 0.

B Field

During arithmetic and logic operations, the B field, bits 10 through 12, control the input to S2 and thus the B side of the adder. (See figure 2-6.)

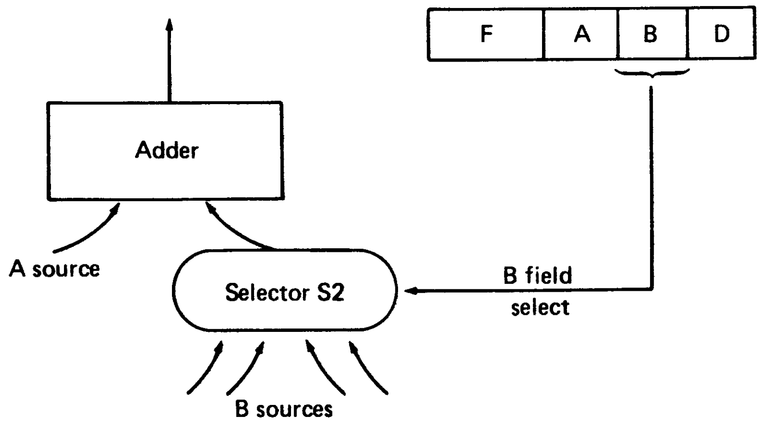


Figure 2-6. The Adder B Field

Coding translations available for the B field are the B codes and B' codes. B' coding is specified if the S field of the microinstruction equals 1000. Note that a code of 001 in the B field is expanded by bits 28 and 29 of the microinstruction and this enables the contents of the N or K register to selector S2. As long as there is no conflict, the outputs from the N and K registers may be used in conjunction with commands or constants in the C field. The B and B' codes are given in tables 2-5 and 2-6, respectively.

TABLE 2-5
B Input Operations

B Codes	MIR28-MIR29	Mnemonics	Operations
000		F2*	Use the contents of the file 2 register as the B source. The value of the N register, before the instruction is executed, is used to address register file 2. If the value of N is changed in the current microinstruction, its initial value is used to reference the file register. F2 must not have been written during the previous instruction.
001	1 1	Zero	The B source is all zeros.
001	1 0	N**	Use the contents of the N register as the B source. Since N is an eight-bit register, this source uses N as the upper eight bits and zeros as the lower bits.
001	0 1	K**	Use the contents of the K register as the B source. Since K is an eight-bit register, the upper bits are zeros and K serves as the lower eight bits.
001	0 0	N,K**	Use the contents of the N and K registers as the B source. These registers are combined with the N register as the upper eight bits of source and with K as the lower eight bits.
010		BG	Use the contents of the BG generator as the B source. The generator has only one bit set to 1, and the position of the bit in the BG register is specified either on value in the N register or by a number in the C field, depending on the state of the controlling SM register bit.
011		X	Use the contents of the X register as the B source.
100		Q	Use the contents of the Q register as the B source.
101		F	Use the contents of the F register as the B source.

TABLE 2-5 (cont'd.)

B Codes	MIR28-MIR29	Mnemonics	Operations
110		F1	This code is similar to F2, but it uses the contents of optional file 1 register addressed by (K) or an external source as the B source input. SM111 controls the selection of F1/external. F1 must not have been written during the preceding microinstruction.
111		MEM***	This code obtains data read from macromemory and uses it as the B source.

*Restriction: The value of the addressing register (N or K) cannot have been modified by a C increment or decrement command in the preceding microinstruction.

**The most significant sixteen bits of the 32-bit processor are zeros. These codes control only the two lower-order eight-bit bytes in the sixteen least significant bits of the B source input in a 32-bit processor.

***Restriction: If the macromemory READ command was not given in the preceding microinstruction, all ones are input to the B source. Exception: If the A source is a prime code, the A prime code source data is input to S2. This command (MEM) is restricted to a microinstruction with type A, B, or C execution time.

TABLE 2-6
B' Codes

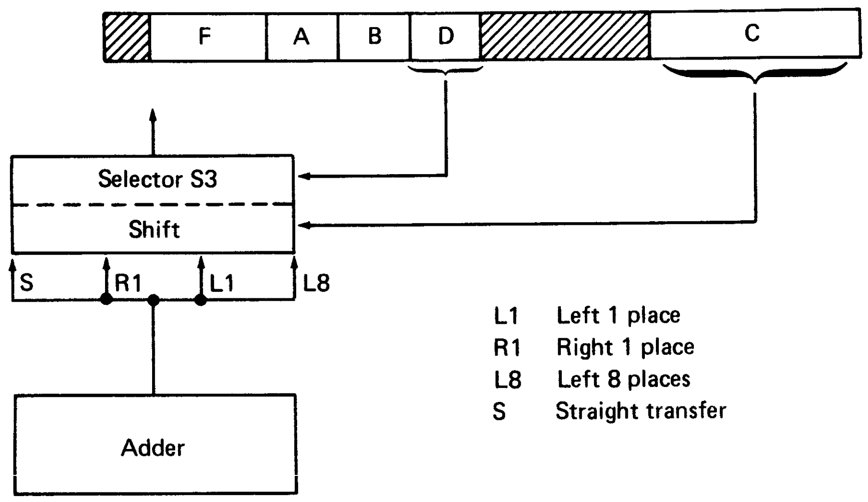
B' Codes*	Mnemonics	Operations
000	OPEN	
001	CRTJ	Transfer the complement of the RTJ register to the twelve least significant bits of S2. Transfer is to the four most significant bits of S2.
010	INRO	Input data/status from the I/O channel.
011	INRS	Input to S2 I/O response signals.
100	MMU	Transfer the upper sixteen bits of data from the micromemory to the X register in the microprocessor. The 32-bit processor transfers the total 32-bit word. The F field must make a reference to the B source. The address is specified by transform or NK. The D field must be an NOP. (See micromemory operand references section for further details.)
101	MML	Transfer the lower sixteen bits of data from the micromemory to the X register in the sixteen-bit processor. This code is not operative in a 32-bit processor.
110 or 111	INTA**	Use the contents of the interrupt address encoder as the B source. The output of this encoder represents the complement of the interrupt address of the highest priority interrupt line that is active having its corresponding mask bit set.

*The B' codes are specified by the S field equal to 1000.
**Restriction: An INTU test command must be given in the preceding microinstruction.

D Field

Bits 13 through 15 specify the destination of information from the ALU organization. The two sources of this information are:

- Selector S3
- Selector S1



L1	Left 1 place
R1	Right 1 place
L8	Left 8 places
S	Straight transfer

Figure 2-7. D Destination

The D destinations are given in table 2-7.

TABLE 2-7
D-Code Transfers

D Codes	Mnemonics	Operations
000	NOP	Do not transfer data to any destination.
001	P*	Transfer output of S3 to P, AB (macromemory address buffer register).
010	I	Transfer output of S1 to I, AB.
011	Q	Transfer output of S3 to Q, AB.
100	F1**	Transfer output of S3 to F register, AB, and write this data in file 1 at the address specified by K at the completion of this instruction.
101	A	Transfer output of S3 to A, AB.
110	X	Transfer output of S3 to X, AB.
111	F	Transfer output of S3 to F, AB.

*If a D-field command to load AB is issued in the next microinstruction following the microinstruction with this command, the transfer to AB is inhibited.
**Data is written into the file 1 register during the first part of the next microinstruction, taking advantage of the updated value of K from this microinstruction. The next microinstruction must not specify a read of file 1.

D' destinations are specified by the D field whenever the S field of the microinstruction equals 1001 or 1010. Table 2-8 gives the D' destinations.

D'' and double D (DD) destinations are required when double precision arithmetic is performed. Since this option is not present in our configuration, D'' coding is not used.

TABLE 2-8
D' Code Transfers

D' Codes	Mnemonics	Operations
000	IOD	Transfer the output of S3 to the I/O data register.
001	IOA	Transfer the output of S3 via the I/O data register to the I/O address register. This destroys the contents of the I/O data register.
010	MMU	Transfer the output of S2 to the upper sixteen bits of micromemory in the sixteen-bit processor, or transfer the output of S2 to the 32-bit word in micromemory in the 32-bit processor. (See micromemory operand reference section for further details.)
011	MML	Transfer the output of S2 to the lower sixteen bits of micromemory location in the sixteen-bit processor, or transfer the output of S2 to the 32-bit word in micromemory in the 32-bit processor. (See micromemory operand reference section for further details.)
100	M1	Transfer the output of ALU to mask register 1.
101	SM1	Transfer the output of ALU to SM register 1.
110	M2	Transfer the output of ALU to mask register 2.
111	SM2	Transfer the output of ALU to SM register 2.

Note: Outputs to the mask and SM registers are direct from the ALU and are not shiftable.

Shift Operations

The ALU control field can contain shift and scale control information. A shift operation is performed if the F subfield equals 11110. This function causes the remaining subfields of the ALU field to assume a different meaning than they did for arithmetic and logic operations. These bit meanings are shown in figure 2-8.

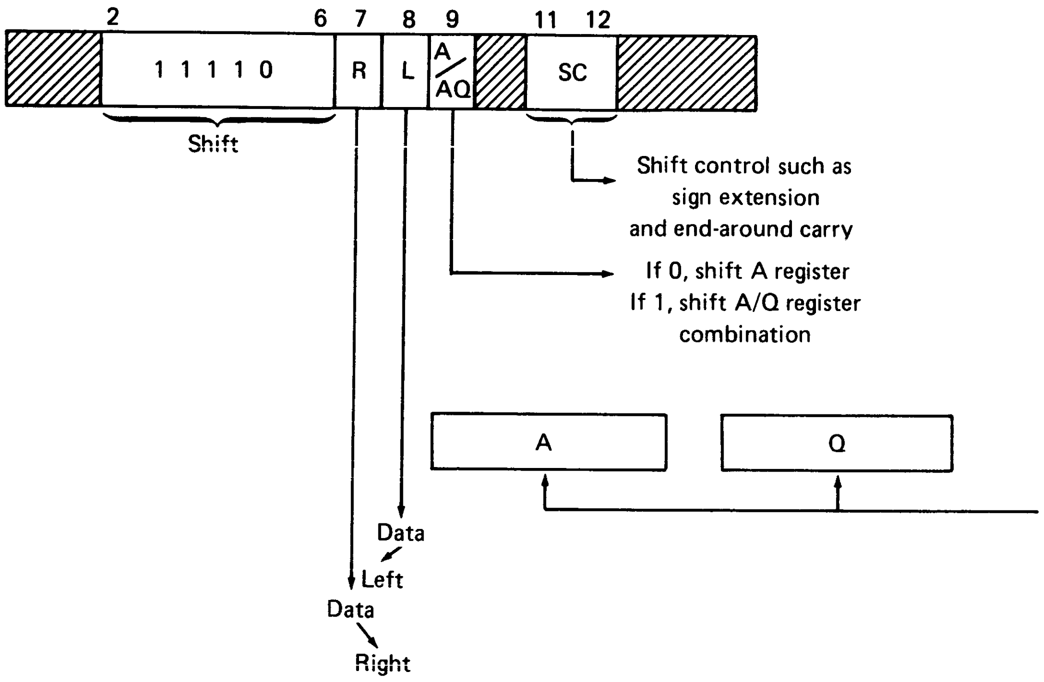

2
6
7
8
9
11
12
1 1 1 1 0
R
L
A/AQ
SC
Shift
Shift control such as sign extension and end-around carry
If 0, shift A register
If 1, shift A/Q register combination
A
Q
Data ↙ Left
Data ↘ Right

Figure 2-8. ALU Control Subfield Meanings for Shift and Scale Operations

The type of shift performed is determined by the coding of bits 7 through 12 of the microinstruction, while the amount of shift depends on the number contained in the N register. The two types of shift performed are an A register shift and an A/Q register shift. During a shift operation, the adder is not used. Table 2-9 defines shift operations.

TABLE 2-9
Shift Operations

Bit Codes		Mnemonics	Operations
7 8 9	11 12		
1 0 0	0 0	AR0E	A is right-shifted (N) bits, with 0 entered as the most significant bit.
1 0 0	0 1	ARSE	A is right-shifted (N) bits, with sign extension.
1 0 0	1 0	AREA	A is right-shifted (N) bits, with end-around carry.
0 1 0	0 0	AL0E	A is left-shifted (N) bits, with 0 entered as the least significant bit.
0 1 0	0 1	AL1E	A is left-shifted (N) bits, with 1 entered as the least significant bit.
0 1 0	1 0	ALEA	A is left-shifted (N) bits, with end-around carry.
1 0 1	0 0	AQR0E	A/Q is right-shifted (N) bits, with 0 entered as the most significant bit in A.
1 0 1	0 1	AQRSE	A/Q is right-shifted (N) bits, with sign extension.
1 0 1	1 0	AQREA	A/Q is right-shifted (N) bits, with end-around carry.
0 1 1	0 0	AQL0E	A/Q is left-shifted (N) bits, with 0 entered as the least significant bit in Q.
0 1 1	1 0	AQLEA	A/Q is left-shifted (N) bits, with end-around carry.

Note: (N) = Contents of register N.

Scale Operations

Most computers operate with a fixed range of numbers. For example, the CYBER 18 has only 16 bits of data, with values ranging from 0000 to $FFFF_{16}$. When data has a decimal (radix) point, it is very important to keep track of the placement of the point. On very large numbers, only the most significant digits may be used. A common method used by scientists and engineers to keep track of the radix point and to represent very large numbers is called scientific notation. This form consists of two parts: the most significant digits represented as a value between 1 and 9 and the exponent, which specifies the position of the radix point. The number represented is equal to the significant digits multipled by the exponent. For example, the value $186{,}000_{10}$ could be represented as 1.86×10^5.

A similar method of notation called floating point arithmetic is used in scientific computer data. The most significant digits are represented as a fractional value of 0.1_2 with an exponent to specify the radix point position. For example, the value 1010.1_2 could be represented as $.10101 \times 2^4$. With this method, the radix point for all data is always at the same position in the computer register. The purpose of the scale operation is to convert integer (whole) numbers into normalized floating point (fractional) numbers with exponents.

Scale operations are similar to shift operations, but the scale is stopped when the upper two bits of "A," bits A00 and A01, are not equal. (The scale point is normally between bits 0 and 1 of the A register.) The maximum number of bits to scale is contained in the N register and, on completion of the scale, N is decremented by the number of shifts necessary to scale the number. The scale operation is performed as follows:

1. Processor logic checks N for zero and scale point of the A register (bits 0 and 1) for unlike bits. If N is zero or scale point is satisfied, i.e., $A00 \neq A01$, shifting is terminated.

2. If shifting is terminated, continue at step 4; if not, continue at step 3.

3. The N register is decremented, followed by a one-bit position shift in A or A/Q as specified in the A and B instruction fields. Processing continues from step 1.

4. Following the termination of scale (an extended timing operation), the execution of remaining field codes (M, D, S, C, T) is completed. The next instruction is selected by the normal sequence control codes.

   Note: If the number being scaled is comprised of all 0's or all 1's (i.e., the number cannot be scaled), then the scale operation is terminated when N = FE_{16} (after passing through N = 0). To avoid executing the microinstruction before the scale operation is completed, N should be at least equal to the number of bits in the word to be scaled.

The type of scale operation is coded in bits 7 through 12 of the microinstruction in the same manner as the shift operation, and allows the same left shift options. When the N register is zero, the scale operation is terminated and the next microinstruction is executed. All scale operations are performed when the F code equals 11111. The scales are given in table 2-10.

TABLE 2-10
Scale Operations

Bit Code		Mnemonic	Operation
7 8 9	11 12		
0 1 0	0 0	SL0E	A is scaled left, with 0 entered as the least significant bit.
0 1 0	0 1	SL1E	A is scaled left, with 1 entered as the least significant bit.
0 1 0	1 0	SLEA	A is scaled left, with end-around carry.
0 1 1	0 0	SDL0E	A/Q is scaled left, with 0 entered as the least significant bit in Q.
0 1 1	1 0	SDLEA	A/Q is scaled left, with end-around carry.

Summary

Table 2-11 summarizes the four subfields found in the ALU control field when it controls arithmetic and logic operations.

TABLE 2-11
Summary of ALU Control Field Functions
Arithmetic and Logic Operations

Fields	Purposes	Bits	Operations
F	Operation of function	2-6	Listed in tables 2-1, 2-2
A	A source	7-9	Listed in tables 2-3, 2-4
B	B source	10-12	Listed in tables 2-5, 2-6
D	Destination	13-15	Listed in table 2-7

The ALU control field also can be used for shift or scale operations, in which case the following subfields occur:

TABLE 2-12
Summary of ALU Control Field
Shift or Scale Operations Functions

Fields	Purposes	Bits	Operations
F	Operation, shift or scale	2-6	Must be a 11110 or 11111
R	Right shift	7	
L	Left shift	8	Listed in tables 2-8, 2-9
A,A/Q	Register or shift	9	
SC	Shift control	11-12	

ALU Control Fields

DIRECTIONS: This exercise presents several examples of the ALU control field. Analyze these examples, and determine the correct answer to each question.

1. $\boxed{10100\ 010\ 100\ 011}$

   ALU CONTROL FIELD

   Given the contents of the ALU control field, determine the final contents of Q after instruction execution.

   MSB LSB

   I = $\boxed{1\ 1\ 0\ 0}$

   Q = $\boxed{0\ 0\ 1\ 1}$

2. $\boxed{11001\ 100\ 011\ 110}$

   ALU CONTROL FIELD

   Would the operation performed by the ALU control field cause an overflow condition to be generated? Why?

   MSB LSB

   A = $\boxed{1\ 1\ 0\ 0}$

   X = $\boxed{0\ 0\ 0\ 0}$

3. $\boxed{00011\ 001\ 011\ 001}$

   ALU CONTROL FIELD

   Given the contents of the ALU control field and the P and and X registers, determine the final contents of the P register after instruction execution.

   MSB LSB

   P = $\boxed{1\ 0\ 0\ 1}$

   MSB LSB

   X = $\boxed{1\ 1\ 1\ 1}$

4. $\boxed{00101\ 101\ 100\ 011}$

   ALU CONTROL FIELD

   Given the contents of the ALU control field and the F and Q registers, determine the final contents of the Q register after instruction execution.

   MSB LSB

   F = $\boxed{1\ 1\ 1\ 0}$

   MSB LSB

   Q = $\boxed{0\ 0\ 1\ 0}$

5. $\boxed{11110\ 101\ 010\ 000}$

ALU CONTROL FIELD

MSB LSB
N = $\boxed{0\ 0\ 0\ 1\ 1}$

MSB LSB
A = $\boxed{0\ 0\ 1\ 0}$

MSB LSB
Q = $\boxed{1\ 1\ 1\ 1}$

Given the contents of the ALU control field, the N register, and A and Q registers, determine the final contents of the A/Q register combination.

6. $\boxed{11111\ 011\ 000\ 000}$

ALU CONTROL FIELD

MSB LSB
N = $\boxed{0\ 1\ 1\ 1\ 1}$

MSB LSB
A = $\boxed{0\ 0\ 0\ 0}$

MSB LSB
Q = $\boxed{0\ 1\ 1\ 0}$

Given the contents of the ALU control field, the N register, and A and Q registers, determine the final contents of the A/Q register combination.

ANSWERS

1. Q = $\boxed{1\ 0\ 0\ 1}$

2. NO–no overflow condition is generated for a subtract when both operands have the same sign or for an add when both operands have a different sign.

3. P = $\boxed{1\ 1\ 1\ 1}$

4. Q = $\boxed{1\ 1\ 0\ 1}$

   MSB LSB

5. A = $\boxed{1\ 1\ 1\ 0}$

   MSB LSB

   Q = $\boxed{0\ 1\ 0\ 1}$

6. A = $\boxed{0\ 1\ 1\ 0}$

   Q = $\boxed{0\ 0\ 0\ 0}$

ALU Control Field Operations

You are familiar with the various functional portions of the ALU module: the working registers, files, input and output selectors, and, most important, the adder itself. You are also familiar with the ALU control field of the microinstruction and you know that each subfield of ALU control provides operational control information for some functional area of the ALU module. This reading provides an understanding of <u>how</u> each subfield determines sources of input data, destination of output data, and control of shift operations.

Three of the four subfields of ALU control provide data control. These are the A, B, and D subfields. If the F field specifies an arithmetic or logic operation, the A and B field provide the data source to the adder, while the D field determines a destination for output from the adder. When a shift or scale operation is specified, the A and B fields contain shift control information. The D field is not used for these operations.

ALU control information contained in the MIR is decoded on the control 1 module, which sends control information to the ALU module. Each data control field will be examined in turn to see how field decoding interfaces with the ALU module.

A Field

The A field (bits 7, 8, and 9 of MIR) provide control information to the ALU as shown in figure 2-9.

The function contained in the F field causes the A field to be interpreted as follows:

- Data source selection for arithmetic and logic operations
- Shift select information for shift and scale operations

Data source selection is decoded by A-field bits. Signals (S1S0, S1S1, and S1S2) that provide this selection originate from control 1. These signals provide the data selection to the A source of adder. A source selection, which occurs in selector S1, consists of sixteen 8 to 1 multiplexers as shown in figure 2-10. There are eight possible data sources for selector S1. Each stage of S1 is able to gate one bit from these sources. A' decoding of the A field (S = 0111 or 1010) causes some data source on the tri-state bus to be enabled as A source bus data (BUS00-BUS15).

If a shift or scale is specified by the function code (F field), shift control information is contained in the A field. A shift control multiplexer located on control 1 decodes bits of the A field and generates control signals to the A and Q registers of ALU. The control signals force right or left shifts for the A register or the A/Q register combination. Figure 2-10 shows that data to the A and Q registers originates from selector S3. The shifted output of A and Q is made available to selector S1 and S2, respectively.

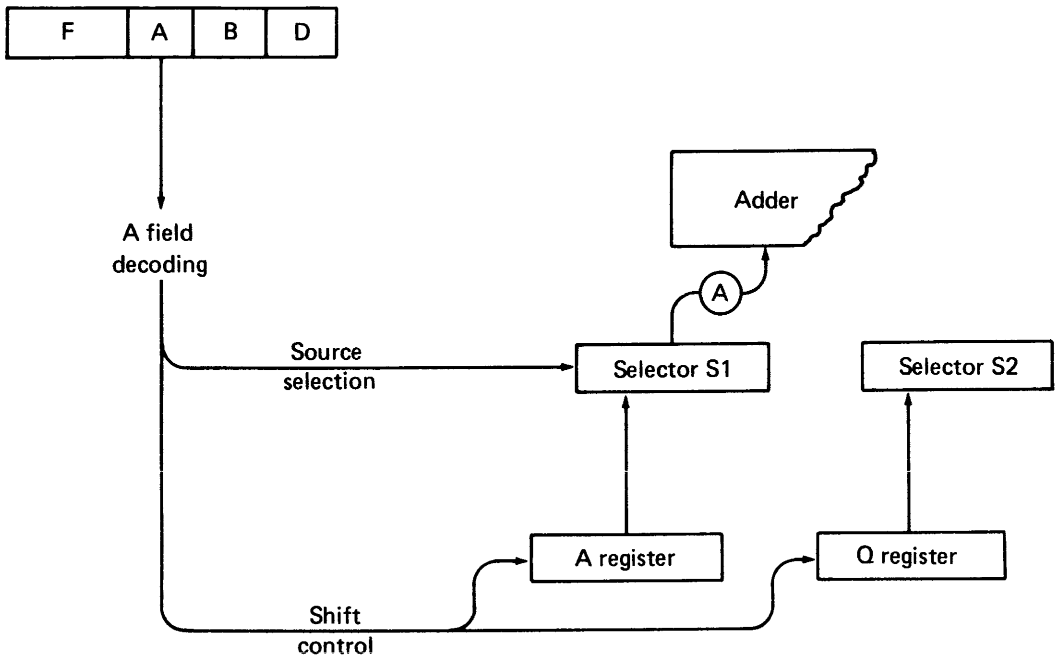


Figure 2-9. A-Field Decoding

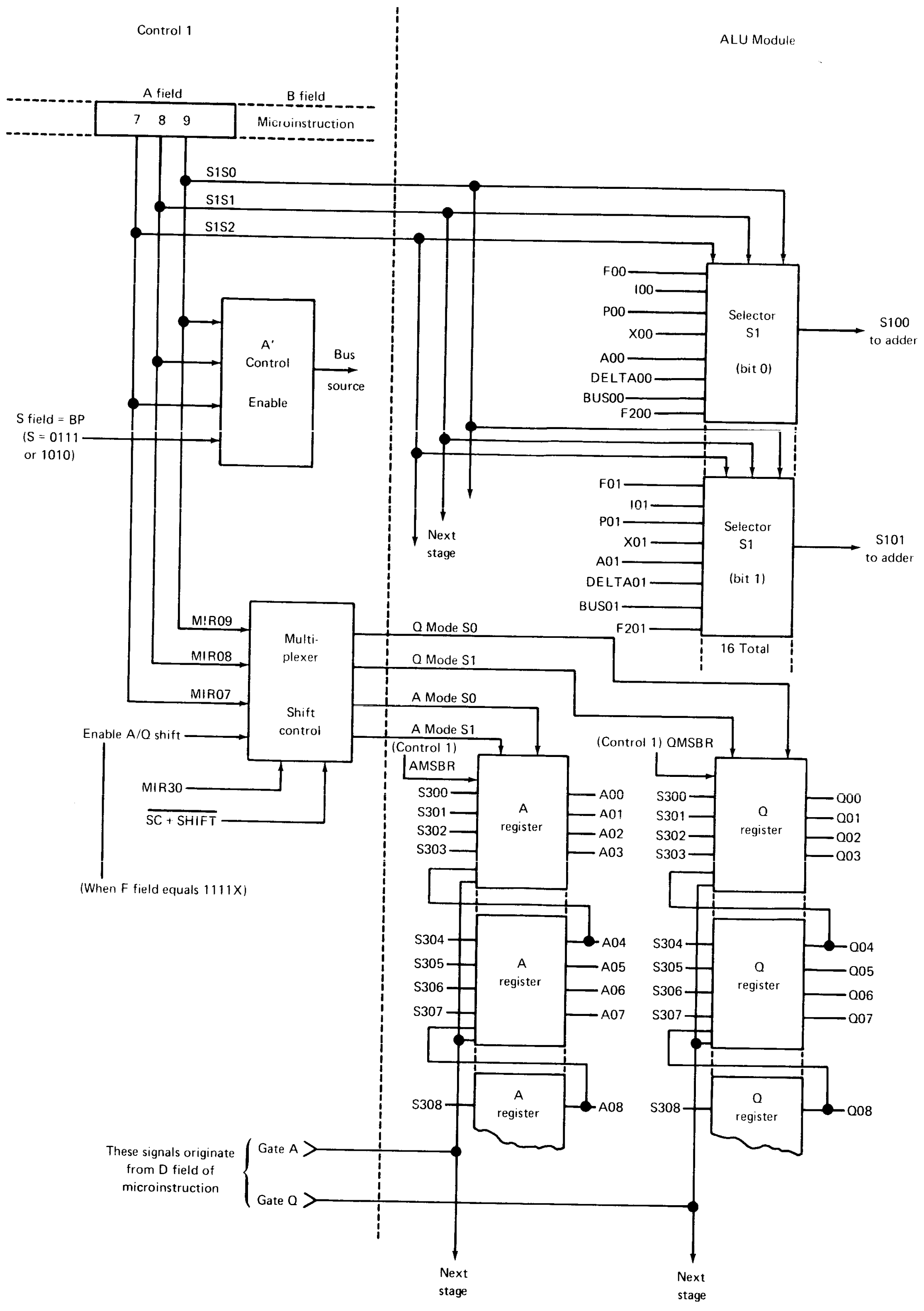


Figure 2-10. A Source Selection

B Field

The B field of ALU control (bits 10, 11, and 12) determines the source of data to the B source of the adder, as shown in figure 2-11.

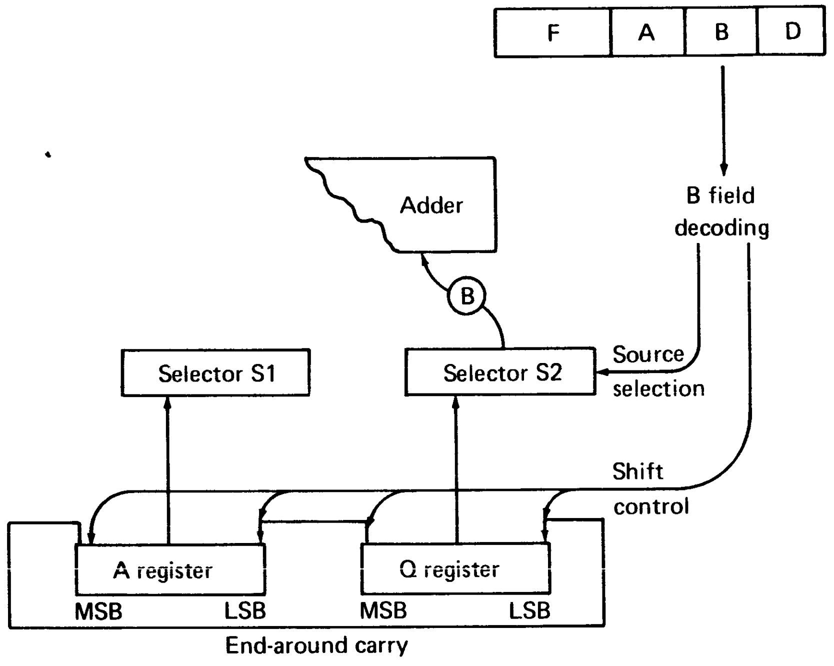


Figure 2-11. B-Field Decoding

Data source selection is decoded by B-field bits. Either B or B' decoding determines the source of data. B decoding selects ALU register sources while B' decoding (which occurs when S = 1000) selects a sixteen-bit data source from the tri-state bus to selector S2 of ALU. (Refer to figure 2-12.)

During shift or scale operations of the A register, bits 11 and 12 provide shift control to the MSB or LSB of the A register. During shift operations using the A/Q register combination, the LSB and MSB of A and Q are controlled by shift control. Shift control bits are interpreted as follows:

Bit 11	Bit 12	Result
0	0	0 is entered in MSB or LSB of A or Q registers
1	0	End-around carry of A or A/Q register
0	1	Sign extension of LSB of A register

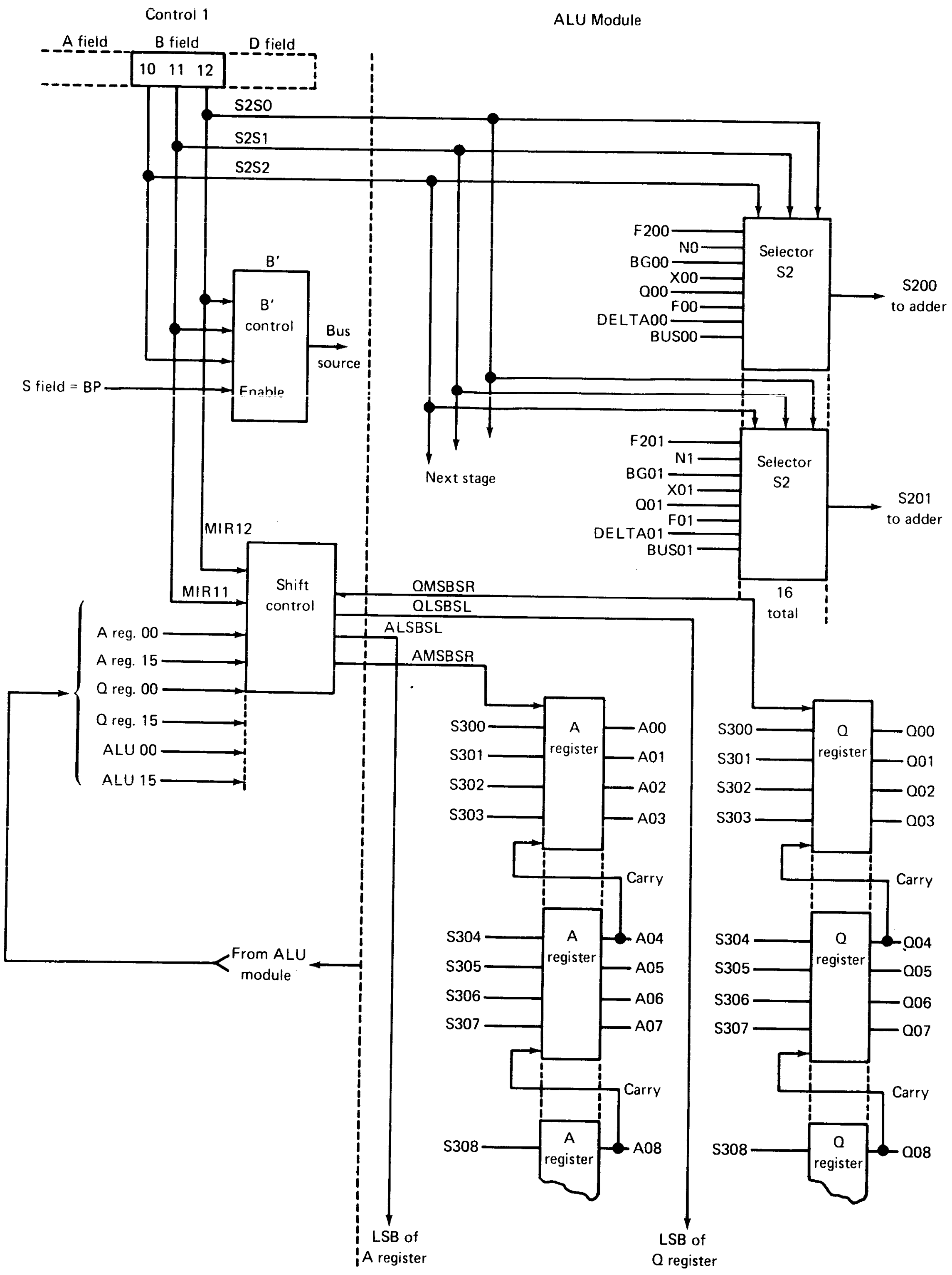


Figure 2-12. B Selection

Shift control monitors the LSB and MSB of the A and Q registers to decide whether bits should be carried from the A register to Q during right shifts or Q register to A during left shifts. Remember that shift control provides the correct bit information to the least and most significant bit positions of the A and Q registers during shift operations.

D Field

Figure 2-13 represents the control exercised by the D field of the microinstruction.

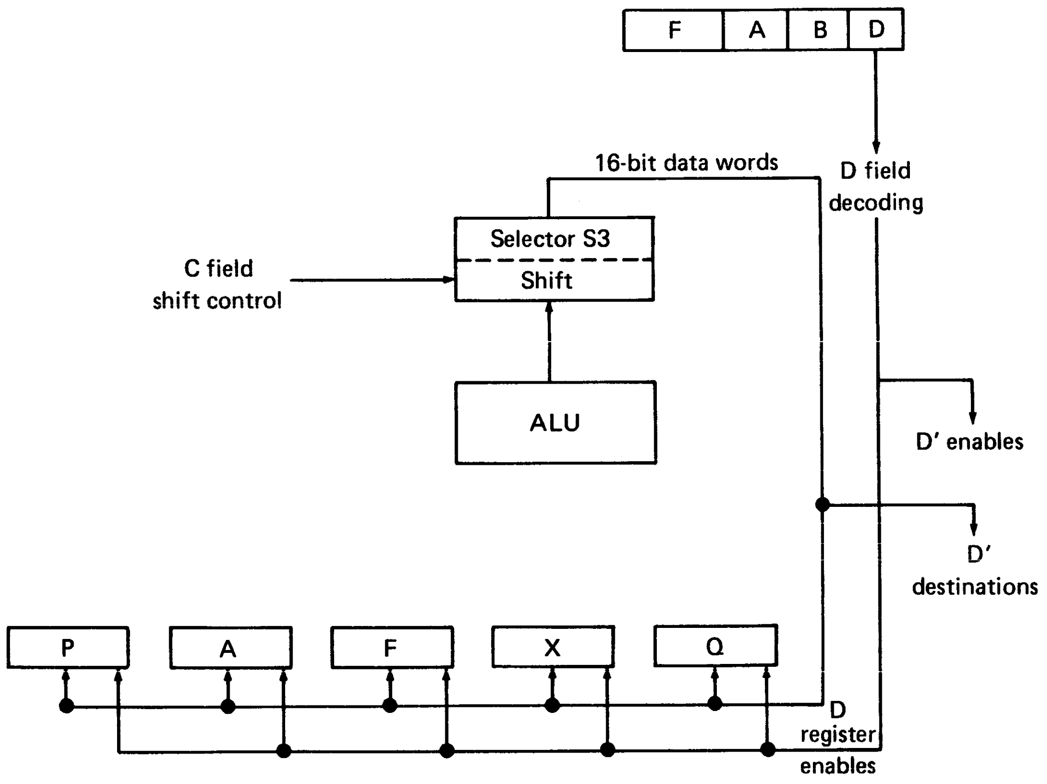


Figure 2-13. D-Field Destination Control

The decoded D field enables a sixteen-bit data word to a destination register. Bits 13, 14, and 15 are decoded as either D or D' codes. D codes gate the output of selector S3 or S1 to one of the working registers on ALU, while D' codes provide gating to destinations in the microprocessor that are not part of ALU. Figure 2-14 is a representation of the logic used to decode the D field and the enabling signals supplied by this decoding.

The D field of ALU control is not used during A register or A/Q register shift operations. However, selector S3 is supplied with a shift network used during arithmetic operations such as multiply and divide. Figure 2-14 shows this shift network. It is controlled by C field contents to provide left and right shifts or straight transfer of ALU output data. The output of the shift network is made available to selector S3 and gated to some destination controlled by the D field contents.

Summary

The A field either selects a source of data or provides shift select information. It selects a source through selector S1 on ALU. During shift operations, a multiplexer on control 1 will decode it and generate shift control signals to A and Q registers.

The B field determines the source of B data to the ALU or provides shift select information. The source of B data is either an ALU register, or some external source available to selector S2. In a shift or scale operation, the B field provides control of MSB and LSB of the A and Q registers.

The D field enables an output to a destination either within the ALU or outside it.

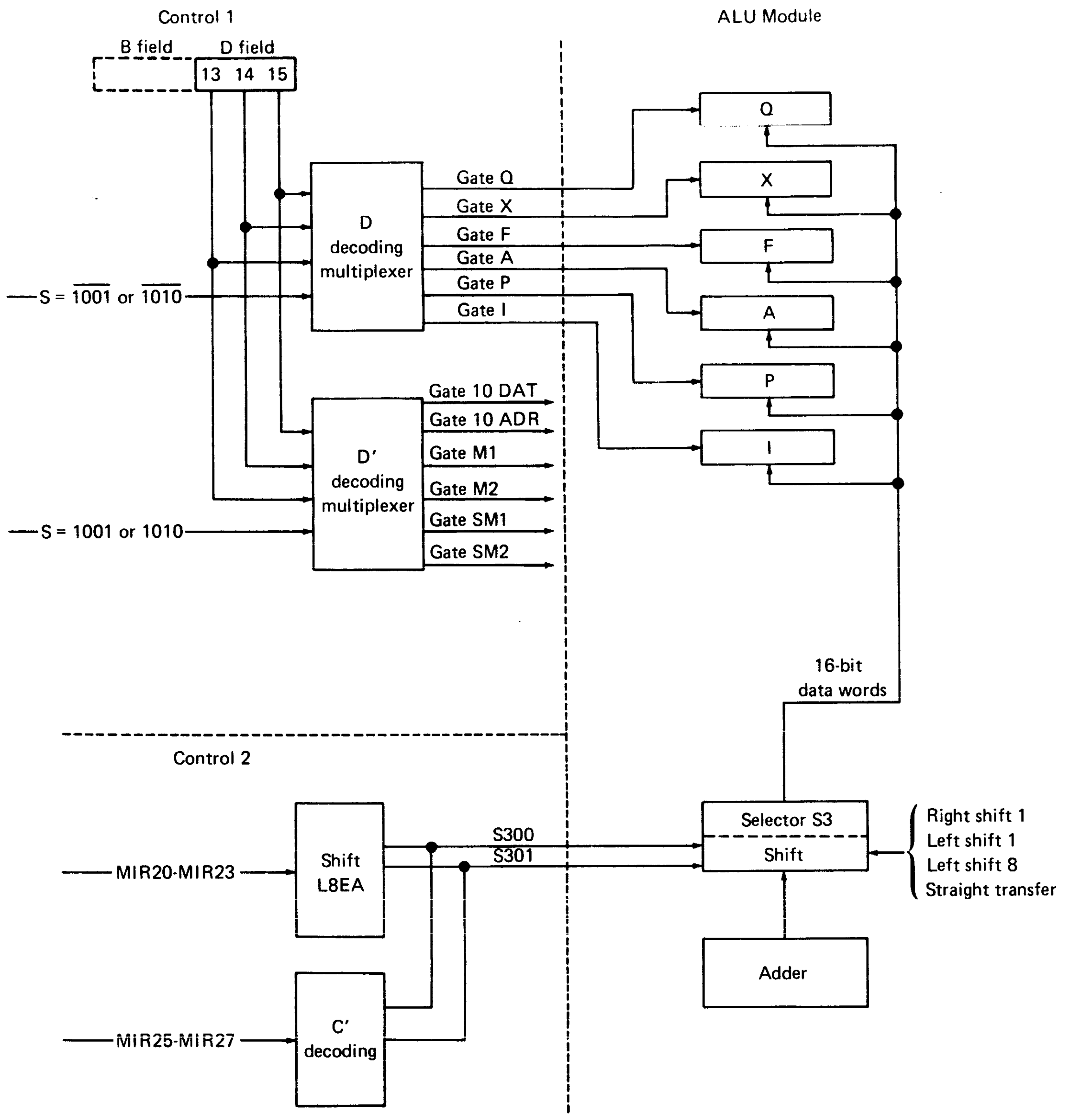


Figure 2-14. D-Field Decoding

ALU Data Flow and Operating Modes

DIRECTIONS: Answer the following questions, using T for true or F for false.

________ 1. If the F field of ALU control equals 11111, A source information is decoded from the A field of the microinstruction.

________ 2. Only one function code is required to shift the A register contents and gate the result to the A source of the adder.

________ 3. When the F field of a microinstruction equaling 11110 is executed, bit 10 of the B field has no meaning.

________ 4. B' decoding selects a data source to the B side of adder other than a working register.

________ 5. The contents of the microinstruction C field determine whether B or B' decoding takes place.

________ 6. The D field contents control the output of the adder for a right shift, left shift, or straight transfer.

________ 7. An A and Q register shift does not require a destination field to be specified in the microinstruction.

________ 8. During a shift A operation, the least significant bits and most significant bits of both the A and Q registers are monitored by the shift control multiplexer.

ANSWERS

1. F 2. F 3. T 4. T 5. F 6. F 7. T 8. F

Arithmetic Operations

You are familiar with microinstruction coding--but not how programs are written and intrepreted. In this reading, a number of microinstruction programs that perform arithmetic operations are analyzed to see what actual activity is generated by the microinstruction code.

Format of Microinstruction Program Listing

A microprogram in the CYBER 18 emulates 1700 macroinstructions. Just one micro-instruction or a series of microinstructions may be required to emulate a 1700 instruction, depending on the microoperations required. Table 2-11 is a representation of a typical microprogram.

TABLE 2-11
Microinstruction Listing

```
                a     b                         c                  d          e     f
             \____/ \______/              \___________/        \______/     \__/ \_____/
CARD VALUE T P/MA  MICRO-MEM   LOCATION  F      A      B      D      S      C        MT    COMMENT
1637                           ****************************************************************************
1638                           *                                                                          *
1639                           *   S T O R E   Q   R E G I S T E R                         F=4            *
1640                           *                                                                          *
1641                           ****************************************************************************

1642  088                                ORG    4X*2+MEMREF1
1643       0  088  54E7 8484 G +STQ      B             Q      F      WRITE EFFADRFF  BTU   STORE (Q), CHECK IF EA=FF

1645  088                                ORG    4X*2+MEMREF1
1646       1  088  54E7 8484 G -         B             Q      F      WRITE EFFADRFF  BTU   STORE (Q), CHECK IF EA=FF

1648       0  089  AC79 4658 B +         SUB-   P      MEM    P      F2WP   RNI      J     SAVEI, P+1 TO P, GOTO RNI
1649       1  089  AC79 4058 B -         SUB-   P      MEM    P             RNI      J            P+1 TO P, GOTO RNI
```

a. Specifies micromemory instruction location (in hexadecimal). The P/MA column contains three digits. The first is the page address; the second two are the micromemory address within a page. The T column specifies the upper 32-bit word when T = 0; T = 1 specifies the lower 32-bit word.

b. The contents in hexadecimal of the 32-bit instruction located at T P/MA.

c. The arithmetic subfields are represented with their mnemonic contents. Operation, data sources, and destination of output are specified.

d. The S and C fields contain the microprocessor operations that take place during instruction execution. Mnemonics are used to represent this operation.

e. The MT field represents the operation contained in the M (mode) and T (test) field of the microinstruction. Of the three characters, the first represents M field contents while the other two represent T field operations.

f. COMMENT is a summary of the microinstruction stating approximately what the microinstruction is accomplishing.

The series of microinstructions represented in table 2-11 emulate a 1700 store Q register function (macroinstruction F field = 4). The explanation of the microcoding is divided in the listing under a series of headings. Each heading describes part of the operation that takes place during a microinstruction execution as follows:

T	Since each micromemory instruction pair is sixty-four bits in length, a 32-bit upper and lower instruction exist at one address. If T = 0, the upper 32-bit word is selected; if T = 1, the lower word is selected.
P/MA	The three-digit address specifies the page and pair address within the page of memory that is read.
MICRO-MEM	The contents of the memory location (thirty-two bits) specified by P/MA is displayed.
ALU CONTROL	A summary of ALU control is contained under F, A, B, and D subfields.
S and C	Microprocessor control operations are represented in these fields in mnemonic code.
MT	A summary of operations specified by the M and T fields is represented in mnemonic code.
COMMENT	A statement explains what the particular microinstruction is accomplishing.

Reading a Listing

It is important to understand how to read instruction listings, since most computer instruction sets are written in a way similar to those in table 2-11. To interpret an instruction listing, refer to figure 2-15, where an example of a logical operation performed by the microprocessor is given.

```
CARD  VALUE  T  P/MA  MICRO-MEM    LOCATION  F     A     B     D     S     C       MT   COMMENT            DIAGNOSTICS
  7                                          LOGICAL OPERATIONS EXAMPLE
  9          0  000   5326 0000              EOR   A     Q     X                        X = (Q) EOR (A)
 10          1  000   48DE 2000              -A    X           X                        COMPLEMENT X
 11          0  001   4AE3 0000              -B          Q     Q                        COMPLEMENT Q
CARD  VALUE  T  P/MA  MICRO-MEM    LOCATION  F     A     B     D     S     C       MT   COMMENT            DIAGNOSTICS
```

Figure 2-15. Example of a Logic Operation

The micromemory address (P/MA) is 000. The first instruction at that address (upper thirty-two bits) is 5326 0000_{16}. The ALU control mnemonics show that the operation that takes place is an exclusive OR of the A and Q register. The result of the exclusive OR is placed in the X register. The comment field states that the X register contents equals the result of an exclusive OR of Q register contents and A register contents. Figure 2-16 represents the contents of the ALU control field during this operation.

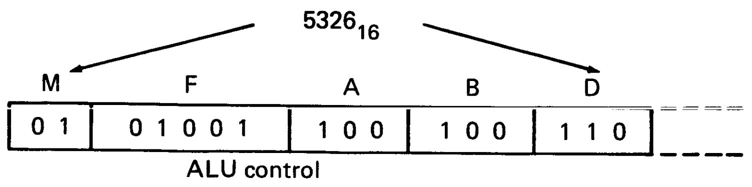


Figure 2-16. Exclusive OR Instruction Execution

Each field has the following content:

M	A sequential instruction is addressed following the execution of the current instruction.
F	This field indicates an exclusive OR takes place using an A source sixteen-bit word and a B source sixteen-bit word.
A	The A source specified is the A register.
B	The B source specified is the Q register.
D	The D field contents causes the result of the exclusive OR operation to be gated to the X register.

The next sequential microinstruction located at address 000 lower executes next. This instruction when executed causes the A source (X register) to be complemented and returned to X. The final instruction (address 001) upper causes the B source (Q register) contents to be complemented and returned to Q.

Scale Operation Example

Figure 2-17 demonstrates a scale operation. The microinstructions that perform this scale are at memory address 006 and address 007. You might recall that a scale performs a shift operation that stops the shift when the two bits at the scale point in the A register are not equal. The scale point normally is specified as being between bits 0 and 1 in the A register. The maximum number of bits to be scaled is contained in the N register. On completion of the scale, the N register contains the original specified maximum minus the number of shifts necessary to position the number so the bits at the scale point are unequal.

```
CARD  VALUE  T  P/MA

CARD  VALUE  T  P/MA  MICRO-MEM    LOCATION  F    A    B    D    S    C        MT   COMMENT                         DIAGNOSTICS
 60                                *         SCALE EXAMPLE ONES COMPLEMENT ARITHMETIC 16 BIT NP
 62          0  006   D8D8 1020                                       N = 32        SET MAXIMUM SHIFT
 63          1  006   7ED0 2800 E            SDLEA                                  N = 32 - NUMBER OF SHIFTS
 65                                *         SCALE EXAMPLE TWOS COMPLEMENT ARITHMETIC 32 BIT NP
 67          0  007   D8D8 1040                                       N = 64        SET MAXIMUM SHIFT
 68          1  007   7FC8 2800 E            SMOE                                   N = 64 - NUMBER OF SHIFTS
CARD  VALUE  T  P/MA  MICRO-MEM    LOCATION  F    A    B    D    S    C        MT   COMMENT                         DIAGNOSTICS
```

Figure 2-17. Example of Scale Operation

Examine the two instructions at address 006. The upper one is shown in figure 2-18.

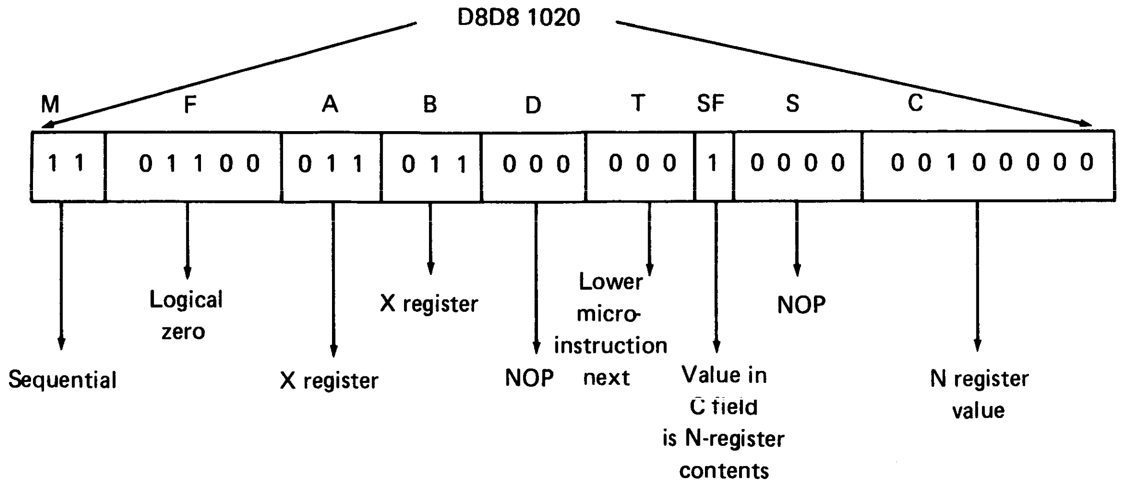


Figure 2-18. N Register Value Set to 32_{10} (20_{16})

This upper instruction places the value of the C field into the N register upon execution. N equals thirty-two or the maximum number of shifts for the A/Q register combination.

The lower instruction shown in figure 2-19 scales the A/Q register combination with end-around carry. After the scale point has been reached or thirty-two shifts have taken place without finding a scale point, the upper instruction at the next address called for in the microprogram is read.

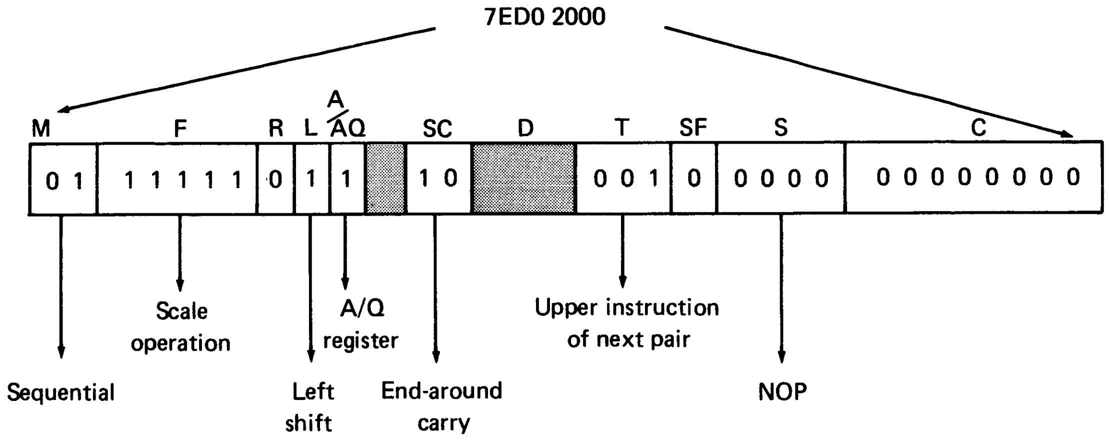


Figure 2-19. Scale Operation

More Examples

Figure 2-20 shows an add to the A register operation. Actually, two microinstructions must be executed to complete this operation. The first instruction forms the effective address (EA) of main memory, from which data is read. The second instruction performs the add operation of memory contents and the A register. The result of the add is stored in A. The second micrȯinstruction also determines the next microinstruction to be executed in the program.

```
CARD  VALUE  T P/MA  MICRO-MEM    LOCATION  F     A     B     D     S     C         MT   COHMENT

1701                              **********************************************************************************
1702                              *                                                                                *
1703                              *  A D D    T O    A    R E G I S T E R                F = 8                     *
1704                              *                                                                                *
1705                              **********************************************************************************

1706   090                                  ORG   8X*2+MEMREF1
1707         0 090   6C79 2300 G  +ADD      SUB-  P     MEM   P     READ             U   READ (FA), P+1 TO P
1708         0 091   B33D 4058 C  +         ADDT  A     MEM   A           RNI       J    A+(EA) TO A, GO TO RNI

1710   090                                  ORG   8X*2+MEMREF1
1711         1 090   B31D 2058 C  -         ADDT  A     X     A           INI       J    A + EA (IM. OPR.) TO A
```

Figure 2-20. Add to A Register Routine

Each of the two instructions is analyzed in turn, that is, the upper instruction at address 090, and at 091. The instruction at address 090 is broken down into its respective fields in figure 2-21.

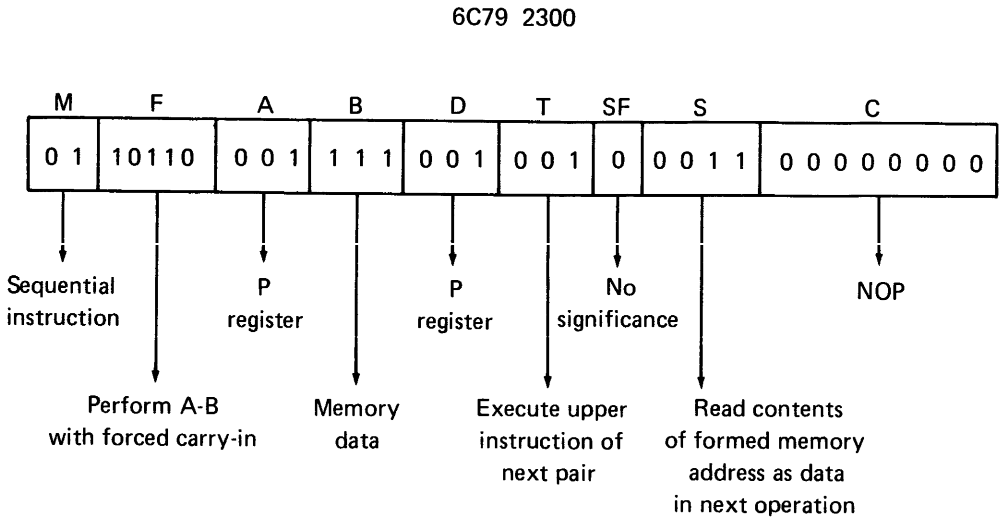


Figure 2-21. Effective Address Use

A summary of each field also is given. The M field contains 01 bit combination, indicating a sequential instruction is the next executed. ALU control calls for an A source minus B source operation with a forced carry-in. A result is the formation of an updated 1700 program count. The address contained in AB addresses the data to be added to A in the next instruction. The T field contents causes the next instruction address' upper instruction to be executed next. The S field specifies that a read of data contained at the main memory address stored in AB takes place.

The breakdown of the second instruction in this sequence is shown in figure 2-22. Again, a summary of each field is given. The instruction performs operations as follows: The M field (M = 10) specifies a jump; that is, the next microinstruction to be executed is not in sequence. The next instruction referenced is that whose address is contained in the C field of the present instruction. ALU control contents does an add of the A register and memory contents, with the result of the add placed in A. The T field specifies that the lower instruction of the next pair will be referenced with the C field containing the address of the next instruction pair.

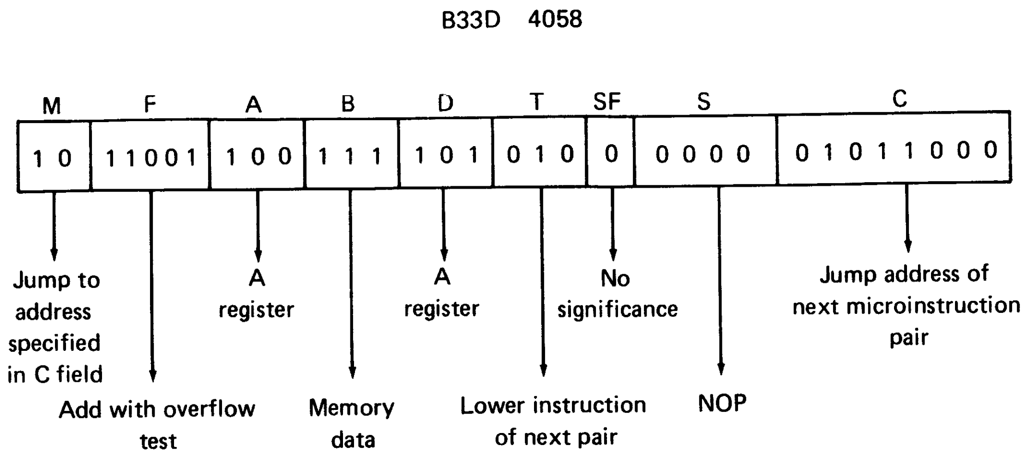


Figure 2-22. Add Memory Data to A Register

Summary

Correct interpretation of instruction listings is not a difficult process. Instruction listings can become a good troubleshooting aid for computer technicians, allowing them to interpret processor failures. In this reading, you have examined a number of listings involving various types of operations. Do not continue to the next activity unless you understand what kind of information is found under each of the following headings:

- T
- P/MA
- MICRO-MEM
- ALU Control
- S and C
- MT
- COMMENTS

PROGRESS CHECK

QUESTIONS

1. What ALU control field identifies the arithmetic or logic operation to be performed?

   a. A
   b. B
   c. D
   d. F

2. What microinstruction field controls selector S1 when it is used to gate information to "I"?

   a. A
   b. B
   c. C
   d. D

3. What D field code gates the result of an arithmetic operation to "X"?

   a. 000
   b. 010
   c. 110
   d. 111

4. What three select signals determine the 16-bit data source through selector S1?

   a. S1S0, S1S1, S1S2
   b. GATEQ, GATEX, GATEF
   c. S300, S301, S302
   d. S2S0, S2S1, S2S2

5. When the F field equals 11110, the information contained in MIR bit positions 7, 8, and 9 is used to ______.

   a. specify an A source input to selector S1
   b. control the direction of a shift operation and the registers to be used in the shift operation
   c. control the number of bits to shift during a scale operation
   d. select the type of logic operation to be performed

6. What circuit controls the number of bit positions to be shifted?

   a. File 2
   b. File 1
   c. "N"
   d. "I"

7. What is the maximum number of shifts that can take place when the "A/Q" combination is used?

   a. 8
   b. 16
   c. 24
   d. 32

8. What field in a microinstruction listing contains information that explains what a particular microinstruction is accomplishing?

   a. Comment
   b. MT
   c. ALU control
   d. T

9. When B' decoding is used, the source of data selected is ______.

   a. a working register
   b. file 1
   c. file 2
   d. a data source on the tri-state bus

10. What F field code causes the A source to the adder to be complemented during a logic operation?

    a. 00000
    b. 00101
    c. 01100
    d. 10100

ANSWERS

1.	Correct Answer:	d	
	Resource:	Text:	CYBER 18-20 Arithmetic Logic SRM, page 2-2.
2.	Correct Answer:	d	
	Resource:	Text:	CYBER 18-20 Arithmetic Logic SRM, page 2-13.
3.	Correct Answer:	c	
	Resource:	Text:	CYBER 18-20 Arithmetic Logic SRM, page 2-13.
4.	Correct Answer:	a	
	Resource:	Text:	CYBER 18-20 Arithmetic Logic SRM, pages 2-23 through 2-25.
5.	Correct Answer:	b	
	Resource:	Text:	CYBER 18-20 Arithmetic Logic SRM, pages 2-15 and 2-16.
6.	Correct Answer:	c	
	Resource:	Text:	CYBER 18-20 Arithmetic Logic SRM, page 2-15.
7.	Correct Answer:	d	
	Resource:	Text:	CYBER 18-20 Arithmetic Logic SRM, page 2-37.
8.	Correct Answer:	a	
	Resource:	Text:	CYBER 18-20 Arithmetic Logic SRM, page 2-37.
9.	Correct Answer:	d	
	Resource:	Text:	CYBER 18-20 Arithmetic Logic SRM, page 2-26.
10.	Correct Answer:	a	
	Resource:	Text:	CYBER 18-20 Arithmetic Logic SRM, page 2-4.

Block 3

Arithmetic and Logic Operations

ALU Operations

You are already familiar with the operation of each functional area of the ALU and with the data flow paths through it. This reading familiarizes you with each functional block on the ALU at a logic level.

Read the explanation of the various logic figures and then examine carefully each one's operation. Figure 3-1, a block diagram of the ALU module, will help you to understand the relationships among logic blocks in the ALU. The functional areas to be examined in this reading are:

- P register (as an example of a typical working register)
- A register with shift control
- File 2
- Selector S1
- Adder with look-ahead carry
- Selector S3 with shift control

P Register

Six working registers are contained on the ALU module: I, P, A, F, X, and Q. These registers serve as temporary storage and data transfer registers. Except for the A and Q registers, which are capable of shift operations, all working registers operate similarly.

The P register is typical of a working register. It consists of D-type flip-flops, half of which are represented in the logic diagram in figure 3-2. Data input, which comes from selector S3, is gated into P whenever a GATE P signal is present. This signal is decoded from the D field of the microinstruction. The Q, or true, output of each P register flip-flop is available to selector S1 as a source of data to the A input of the adder.

A Register

Figure 3-3 is a logic representation of another type of working register. This register not only can store a data word but also can shift its contents. The A register (like the Q register), is a set of four shiftable register ICs controlled by AMODES0 and AMODES1 enabling signals. These signals originate in the decoding of the A and B fields of the microinstruction when an arithmetic or shift operation takes place. Input from selector S3 to the A register flip-flops occurs when a GATE A signal is present. The output from A is available to selector S1 as an A source input to the adder.

During a shift operation, data contained in the A register is shifted internally either right or left with serial-by-bit inputs to and from each four-bit shiftable register IC as indicated. During A/Q register shifts, the MSB (left shift) and LSB (right shift) of Q are also enabled to the respective bit position of A to allow for serial end-around operations during combined register shifts.

File 2

Figure 3-4 is a logic diagram representation of file 2, an addressable RAM memory with thirty-two sixteen-bit locations. File 2 consists of eight RAM chips having sixteen addressable locations, each able to store four bits of information. Addressing for file 2 originates from the lower five bits of the N register (N3 through N7). N3 selects an upper or lower set of four RAM chips while N4 through N7 determine which of the sixteen locations of that four-chip group is being referenced. Data available to file 2 is the F register data and is written into file 2 whenever $\overline{\text{WEF2}}$ is low at the write enable input (WE). The contents of an F2 location can be used as a data source for selector S1 or S2, depending on the arithmetic operation taking place. The data output is available whenever the enable (E) signal is low at the chip input.

Selector S1

Selector S1 consists of sixteen separate multiplexers, each used to select one bit of a sixteen-bit word. Eight word sources are available to selector S1. Figure 3-5 shows four of the sixteen S1 multiplexers. The select enables (S1S0 through S1S2) control which of the eight sources will be made available at the multiplexer output. These enables are created from the decoded A field of the microinstruction. The $\overline{Q}$ output of selector S1 is made available to the A source inputs of the adder for arithmetic and logical operations. Figure 3-5 also shows a four-bit portion of the I register. This is the only working register of the ALU which does not receive its input from output selector S3. Both data input to the I register and data output from the I register are gated through selector S1.

Adder With Look-Ahead Carry

The adder with look-ahead carry generator is shown in figure 3-6. The adder is made up of four adder chips. Each chip is able to process four bits of data from two separate sixteen-bit data sources (A Source S1 and B Source S2). The internal data process is determined in the adder chips by four adder Select inputs (ALU0 through ALUS3) and adder mode control (ALUM) originating in the decoding of the F field on control 1. The data output of each adder chip (F0 through F3) is available to selector S3 for shifted or nonshifted data flow to some destination. The adder output may also be gated directly to the status mode interrupt module (SMI). The carry-in generator chip monitors the propagate output and the generate output from each adder chip. These two signals cause the carry-in generator to produce the necessary carry inputs to adjoining adder chips during arithmetic operations.

Selector S3

Once data is available at the output of the adder, selector S3 may perform one of four operations (figure 3-7) on that data by enabling:

- Direct transfer inputs D0A and D0B
- Shift left one place inputs D1A and D1B
- Shift right one place inputs D2A and D2B
- Shift left eight places inputs D3A and D3B

A logic diagram representation of eight bits of selector S3 is shown in figure 3-8.

Selector S3 consists of eight separate multiplexers, four of which are represented in figure 3-8. Select signals S3S0 and S3S1 are used to enable one of four possible input sources. (Refer to the truth table which demonstrates the sources enabled to selector outputs by these select signals.) The select signals originate in the decoded C field (MIR20 through MIR31) of the microinstruction. Data output from selector S3 is made available to the working registers, the microprocessor tri-state bus, and I/O logic.

Summary

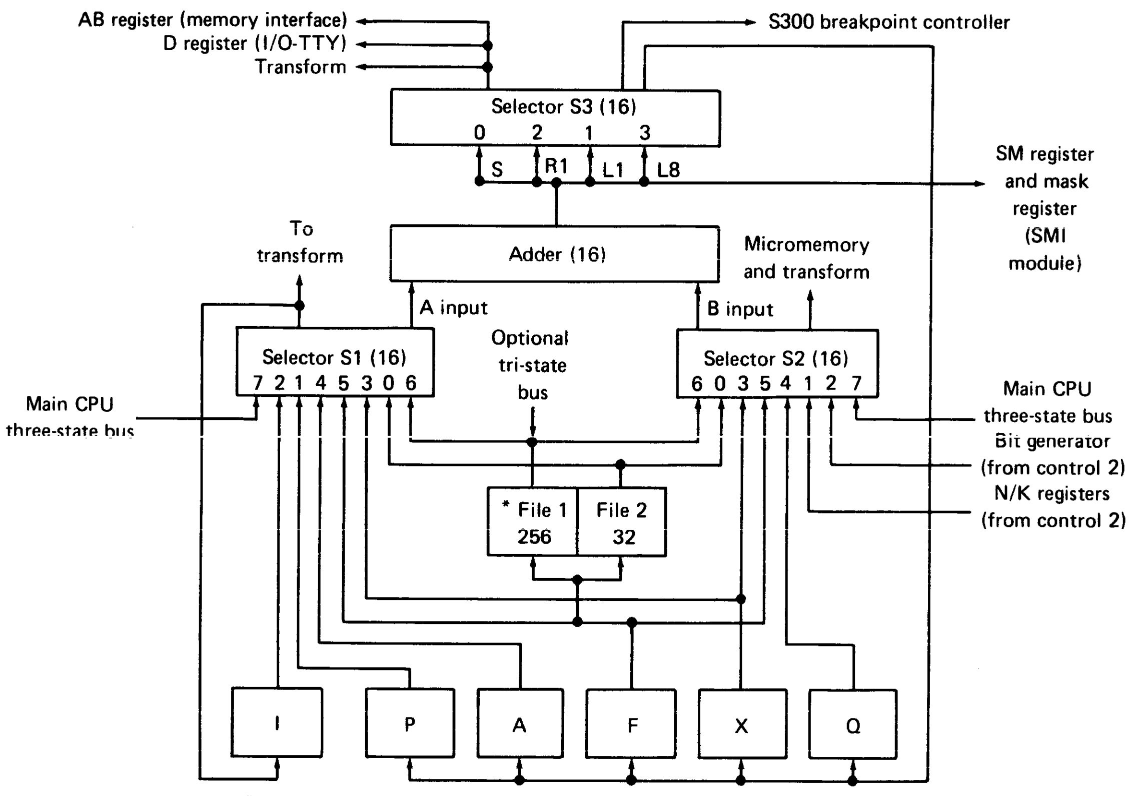
Table 3-1 summarizes the logic level of each part of the ALU's functional block diagram.

TABLE 3-1
ALU Block Diagram at Logic Level

Parts	Purposes	Components
P register	Like other working registers, this register provides temporary storage and data transfer. Output available to selector S1.	16 D-type flip-flop
A register with shift control	Like the Q register, can store a data word or shift its content. Output available to selector S1.	Four shiftable register ICs

TABLE 3-1, Cont.
ALU Block Diagram at Logic Level

Parts	Purposes	Components
File 2	Addressable RAM memory with thirty-two sixteen-bit locations. Output available to selector S1 or S2.	Eight RAM chips
Selector S1	Selects each bit of a sixteen-bit word. Output made available as A source input.	Sixteen multiplexers
Adder with look-ahead carry generator	Processes data from A and B sources. Output made available to selector S3.	Four adder chips One carry generator
Selector S3	Shifts or directly transfers input. Output available to working registers, microprocessor tri-state bus, and I/O logic.	Eight separate multiplexers



* File 1 is optional

Notes:

1. The numbers inside the selector blocks indicate the selector position.
2. The numbers in parentheses indicate the width of registers and selectors.

Figure 3-1. ALU Module Block Diagram

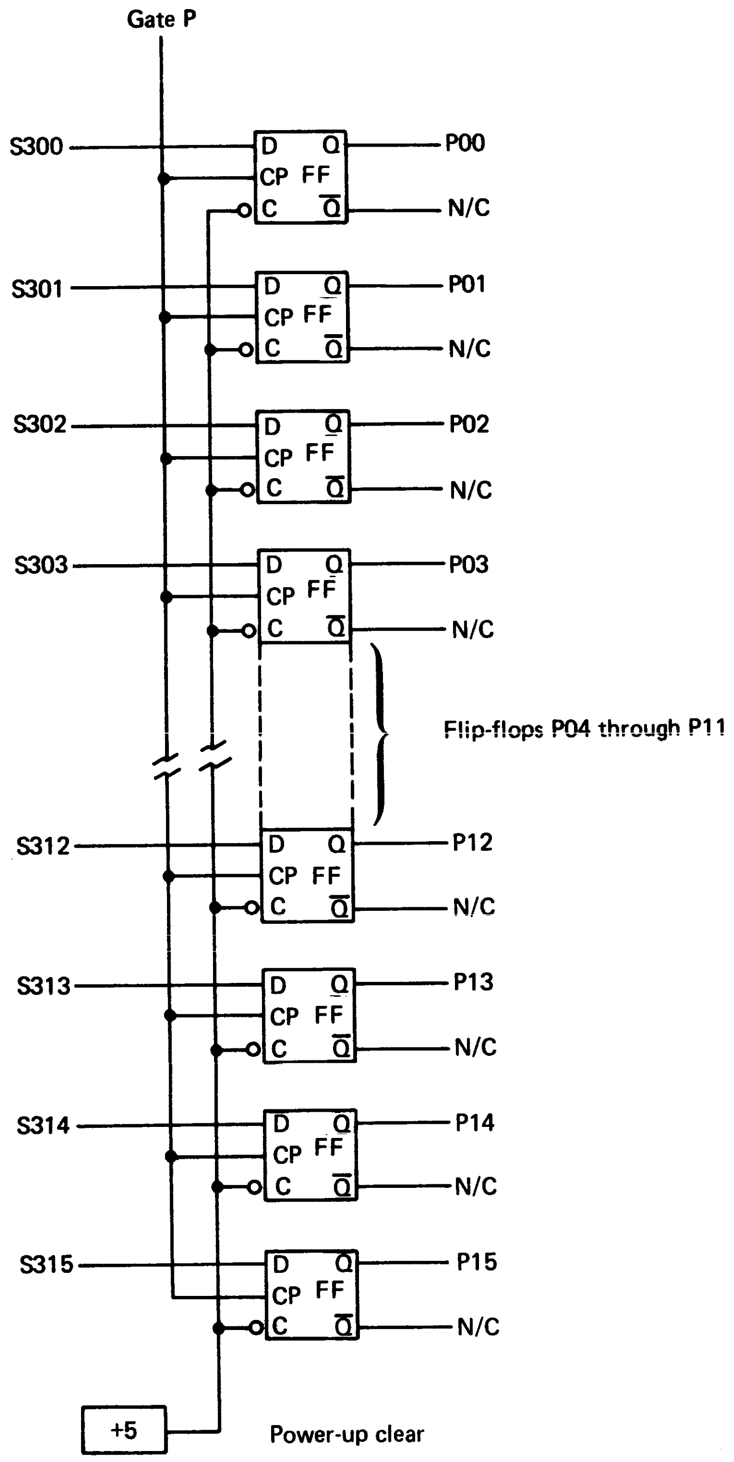


Figure 3-2. P Register

Control 1

S0	S1	Mode
H	H	Parallel Load
H	L	Shift right (down)
L	H	Shift left (up)
L	L	Hold

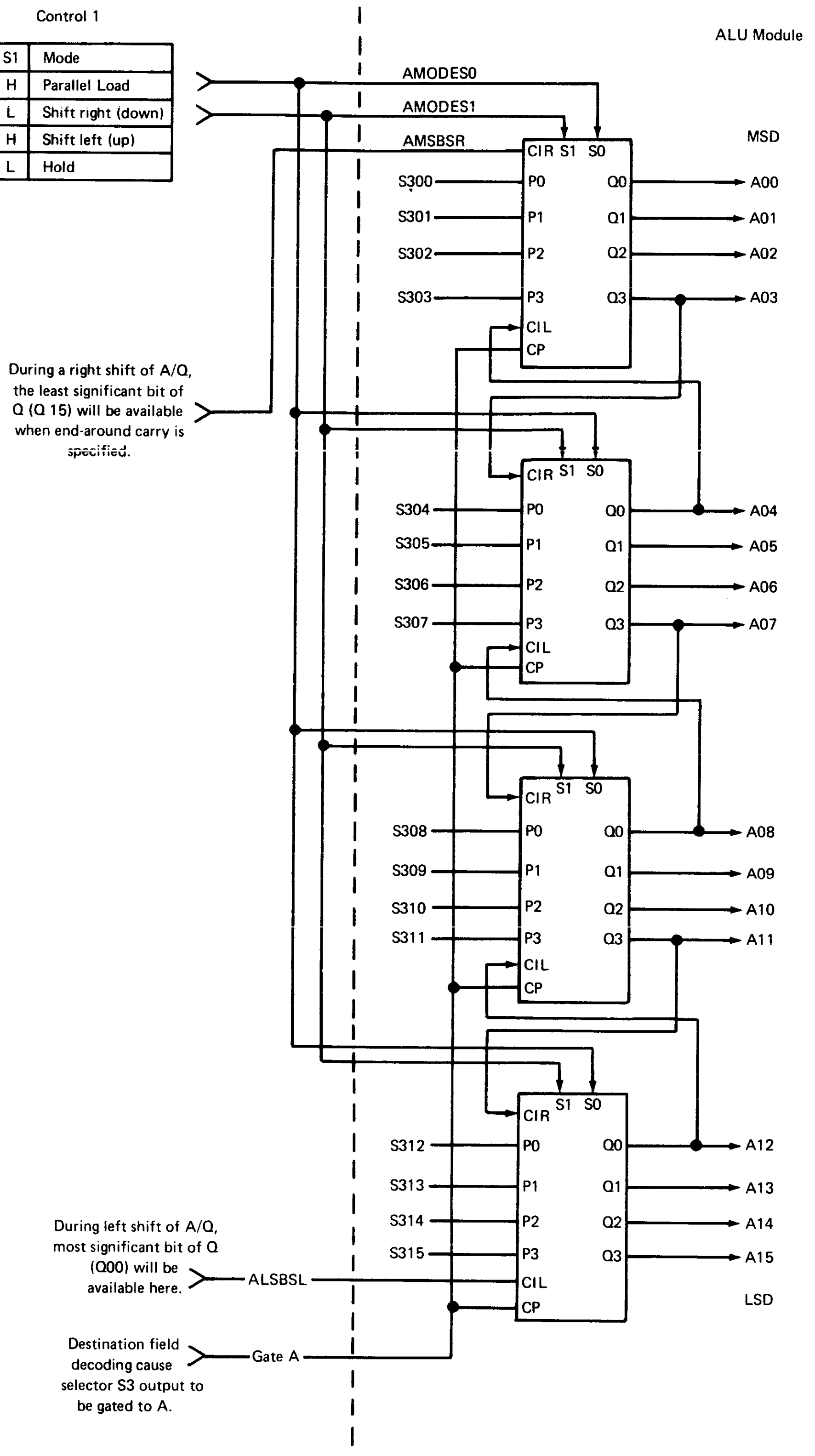

ALU Module
AMODES0
AMODES1
AMSBSR
MSD
LSD
CIR S1 S0
P0 P1 P2 P3 CIL CP
Q0 Q1 Q2 Q3
S300 S301 S302 S303
S304 S305 S306 S307
S308 S309 S310 S311
S312 S313 S314 S315
A00 A01 A02 A03
A04 A05 A06 A07
A08 A09 A10 A11
A12 A13 A14 A15
During a right shift of A/Q, the least significant bit of Q (Q 15) will be available when end-around carry is specified.
During left shift of A/Q, most significant bit of Q (Q00) will be available here.
ALSBSL
Destination field decoding cause selector S3 output to be gated to A.
Gate A

Figure 3-3. A Register and Control Signals

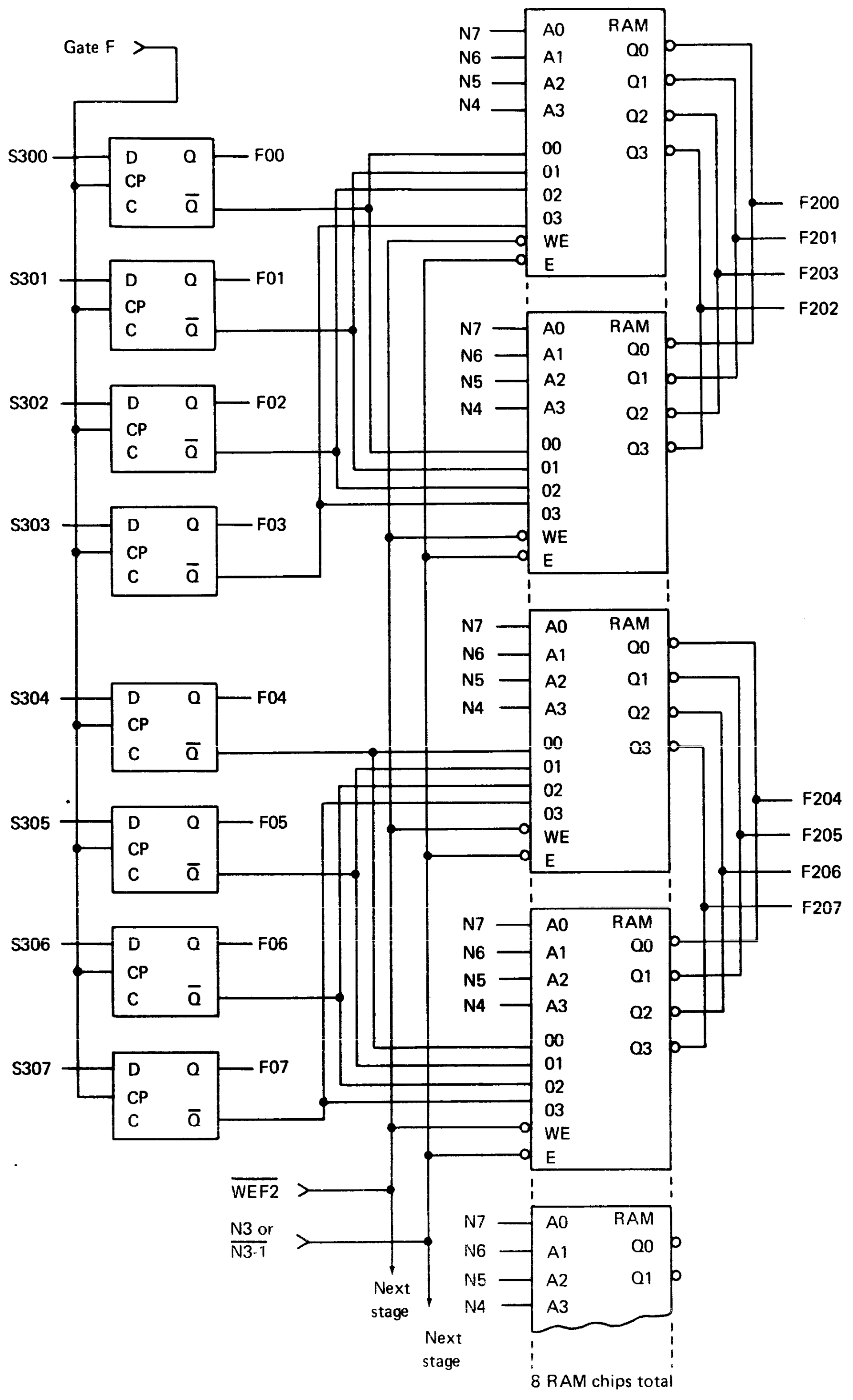


Figure 3-4. File 2 with Control Signals

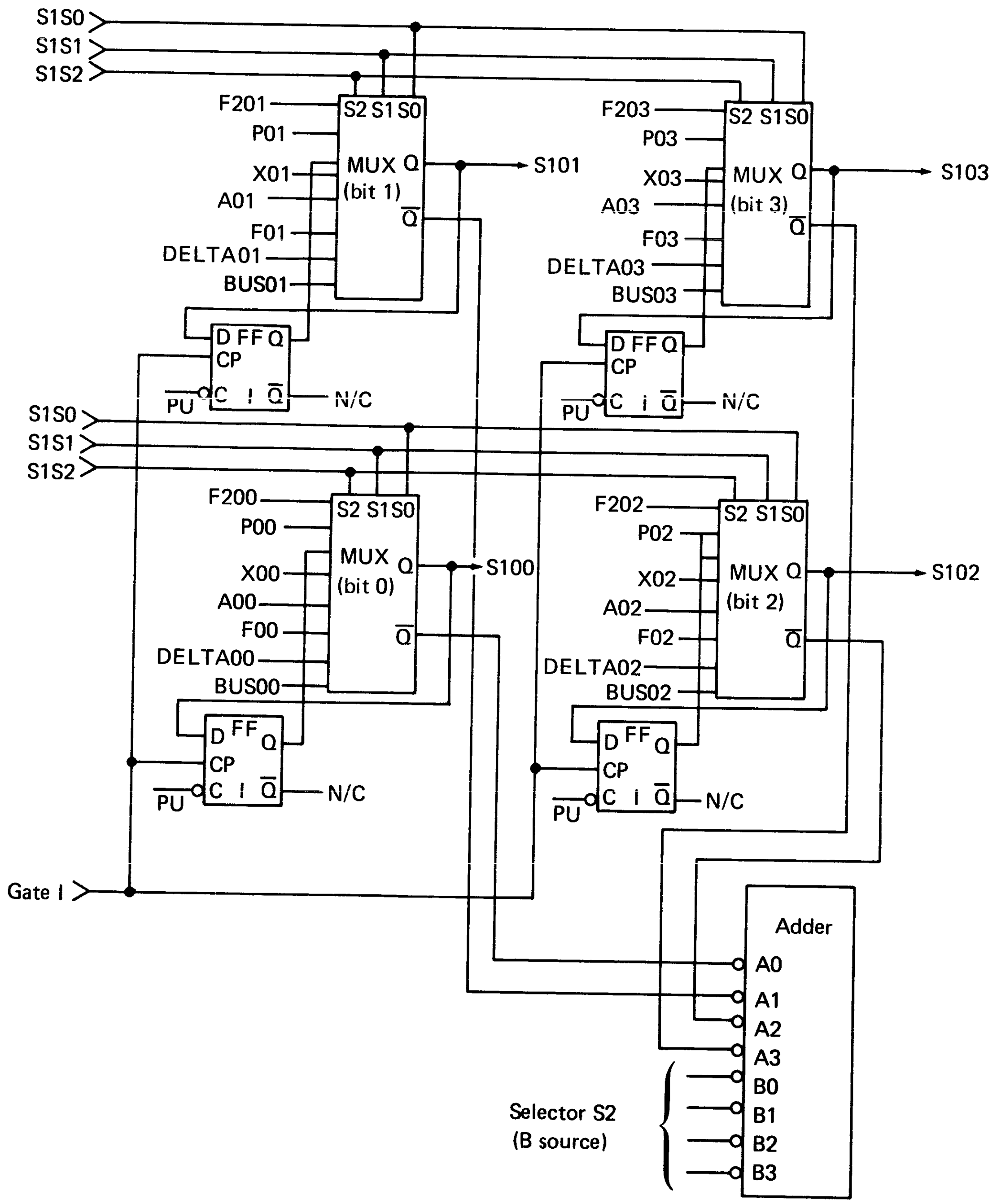


Figure 3-5. Selector S1 with Control Signals

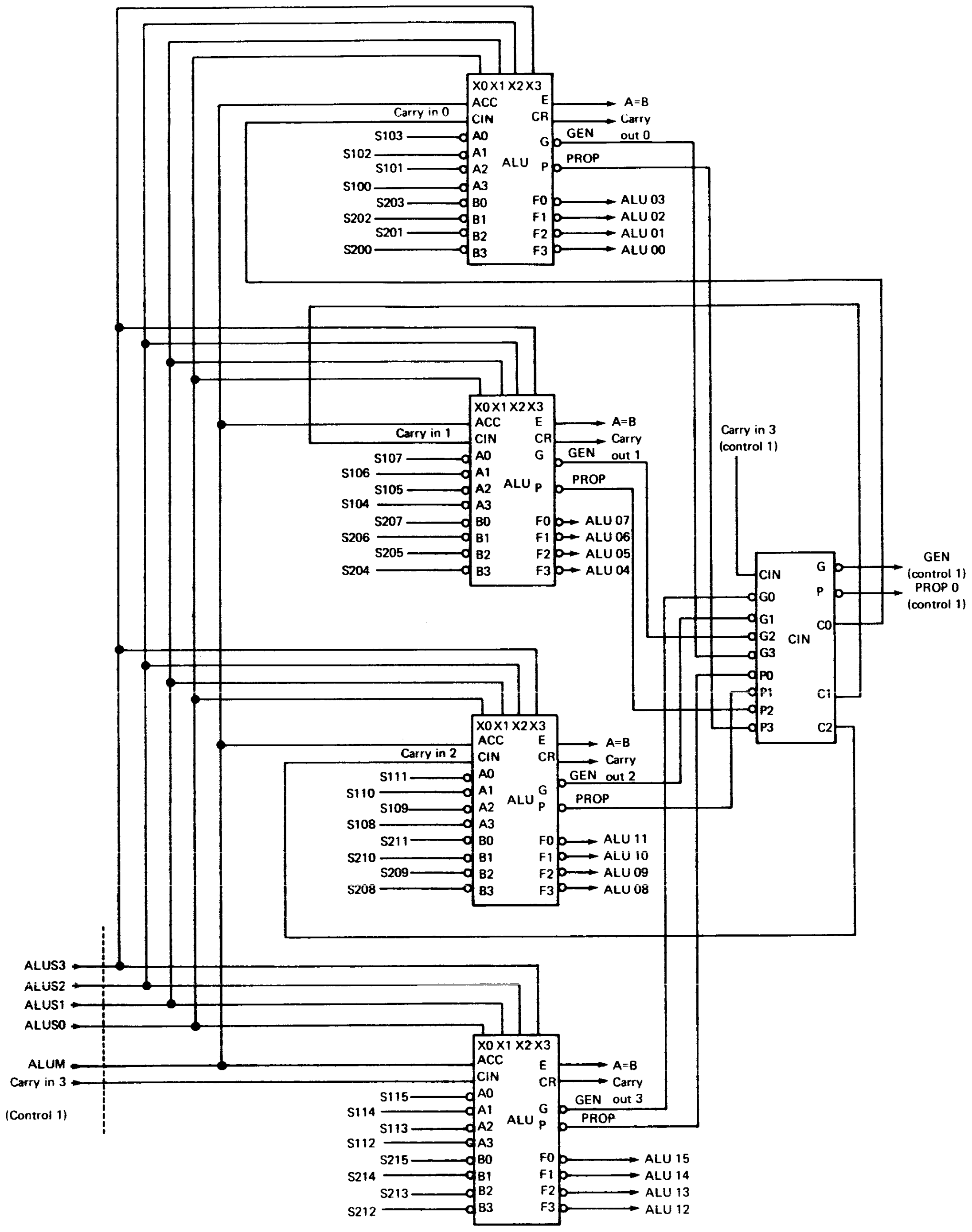


Figure 3-6. Adder with Look-Ahead Carry

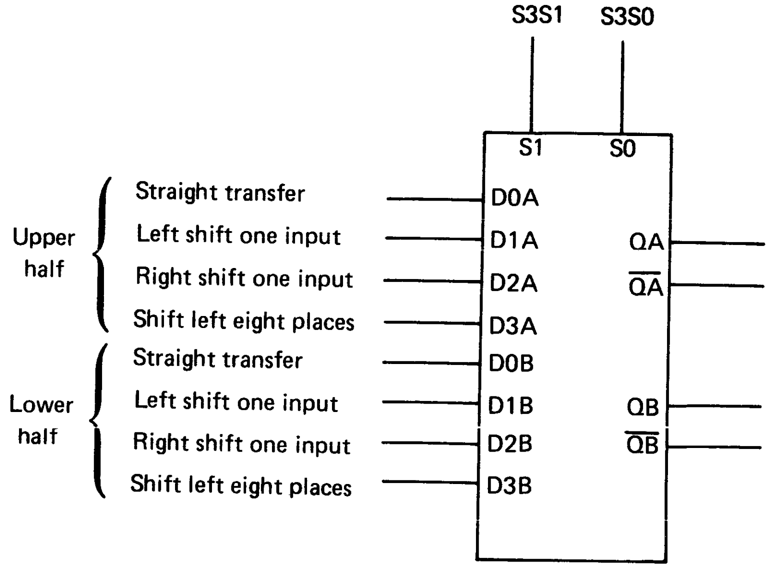
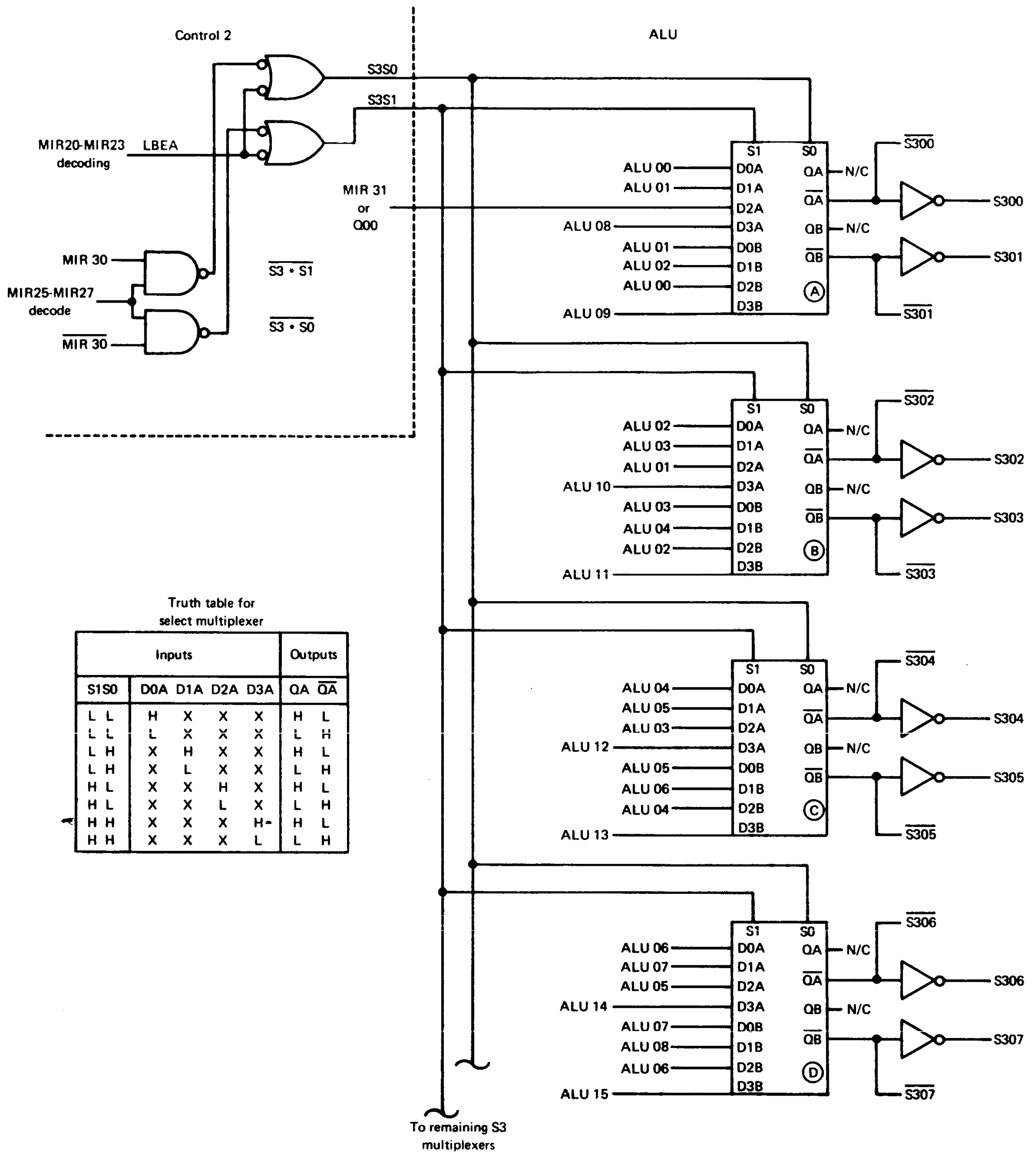


Figure 3-7. Selector S3 Shifting Multiplexer



Truth table for select multiplexer

Inputs					Outputs	
S1S0	D0A	D1A	D2A	D3A	QA	$\overline{\text{QA}}$
L L	H	X	X	X	H	L
L L	L	X	X	X	L	H
L H	X	H	X	X	H	L
L H	X	L	X	X	L	H
H L	X	X	H	X	H	L
H L	X	X	L	X	L	H
H H	X	X	X	H	H	L
H H	X	X	X	L	L	H

Figure 3-8. Selector S3 with Shift Control

ALU Instruction Example

In this reading, you look at the decode process for the ALU control field and the signals that result from this decode. The ALU control field is decoded on the control 1 module. Control signals are made available to the ALU to form data paths to the ALU, control specific adder operations on that data, and determine the destination of the results.

Decoding the ALU Control Field

Decoding of the ALU control field is a simultaneous operation; that is, enables are supplied to the various functional areas of the ALU at the same time so that the following occurs in one operation:

- Data is made available to the ALU
- The adder manipulates this data
- Selector S3 provides a destination for the results

A typical add operation is shown in figure 3-9. The operands to be added are indicated by the microinstruction A and B fields, with the result of this add placed in a destination specified by the D field. Each subfield of this instruction is examined to reveal the decode method used. The signals resulting from the decode are located in the ALU logic diagrams along with the functional areas they control. The first subfields to be examined will be the A and B fields, since they supply the source of operands for arithmetic and logical operations.

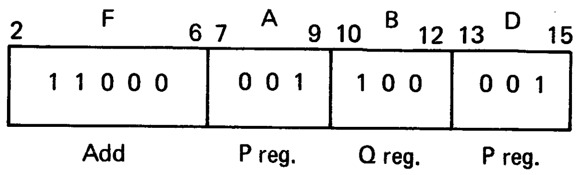


Figure 3-9. Add Operation of P and Q. Result to P

A-Field Decoding

The A field decoder (figure 3-10) monitors bits 7, 8, and 9 of the MIR. The decoder provides enable signals (S1S0-1 through S1S2-1) to the select inputs of selector S1 of the ALU to determine the A source of data. Decoding is accomplished using three OR functions enabled by either T0 (high) or $\overline{\text{AP}}$ (low). $\overline{\text{AP}}$ is present when A' decoding is specified (S field = 0111 or 1010). A' decoding specifies that data to the A source originates from the SMI module via the tri-state bus. When the $\overline{\text{AP}}$ signal to the decoder is low and MIR07 through MIR09 are logic 1's, the memory data bus is enabled as the data source through selector S1.

Turn to page 6 of the ALU logic diagrams and locate selector S1. Four multiplexers, L5, L6, M5, and M6, control the four most significant bit sources of selector S1. There are eight separate data sources present at each multiplexer input. Multiplexer outputs (pin 6) are connected directly to the adder chip L9 (page 6, location C7). Note also that the I register flip-flops provide a data source to selector S1 and receives input from the selector S1 output (pin 5). Other multiplexers that make up selector S1 are located on pages 9, 12, and 15 of the logic diagrams.

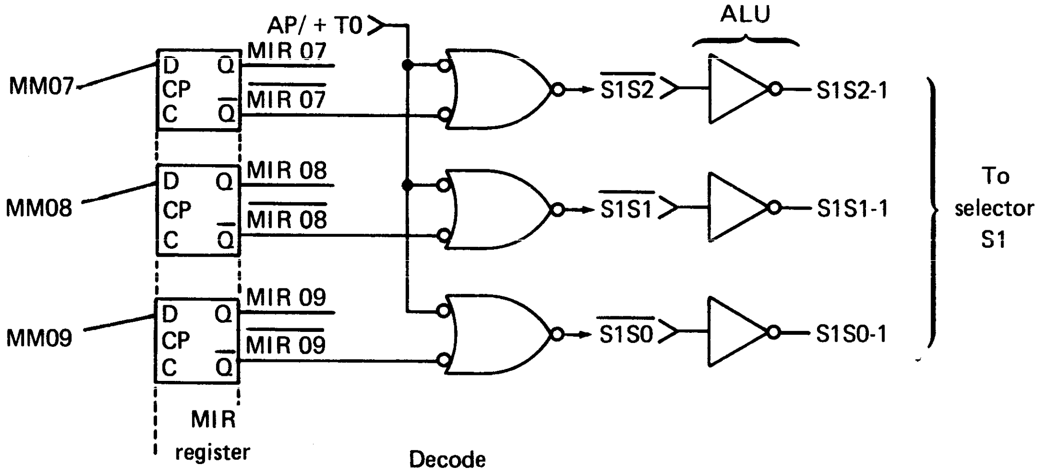


Figure 3-10. A-Field Decoder

B-Field Decoding

The B field (bits 10 through 12) is decoded in much the same way as the A field. (Refer to figure 3-11.) Three OR functions interpret the contents of the MIR B field. The OR function outputs (S2S0-1 through S2S2-1) control data source to the B side of the ALU. ENABLE S2 signal is generated to enable selector S2's lower and/or upper eight bits, respectively. This allows eight bits from two sources through the selector to form a sixteen-bit output (ZERO, N, K, N-K).

Turn to page 7 of the ALU logic diagrams and locate multiplexers L7, L8, M7, and M8. These four multiplexers control the four most significant bit inputs to the ALU B source. Control signals S2S0-1 through S2S2-1 determine which source data will be used. The multiplexers are enabled (E input) whenever the $\overline{\text{ENABLE S2}}$ signal from page 18 is present. The ENABLE S2 signals are provided as follows:

- $\overline{\text{ENABLE S2-1}}$ enables the eight MSB's through S2 (N register) if MIR bit 29 is present (WORD1).
- $\overline{\text{ENABLE S2-2}}$ enables the eight LSB's through S2 (K register) if MIR bit 28 is present (WORD0).

Selector S2 consists of a total of sixteen multiplexers located on pages 7, 10, 13, and 16 of the logic diagrams. One data source to selector S2 which is significant is the bit generator (BG). The bit generator is located on the control 2 module and its main function is to generate a 1 bit at any position in a word as input to the B side of the ALU. Control to drive the bit generator is derived either from microinstruction bits (MIR27 through MIR31) or from the lower five bits of the N register (N03 through N07). BGGP0 and BGGP1 enable the bit generator output to either the lower eight-bit word group or the upper eight-bit word group.

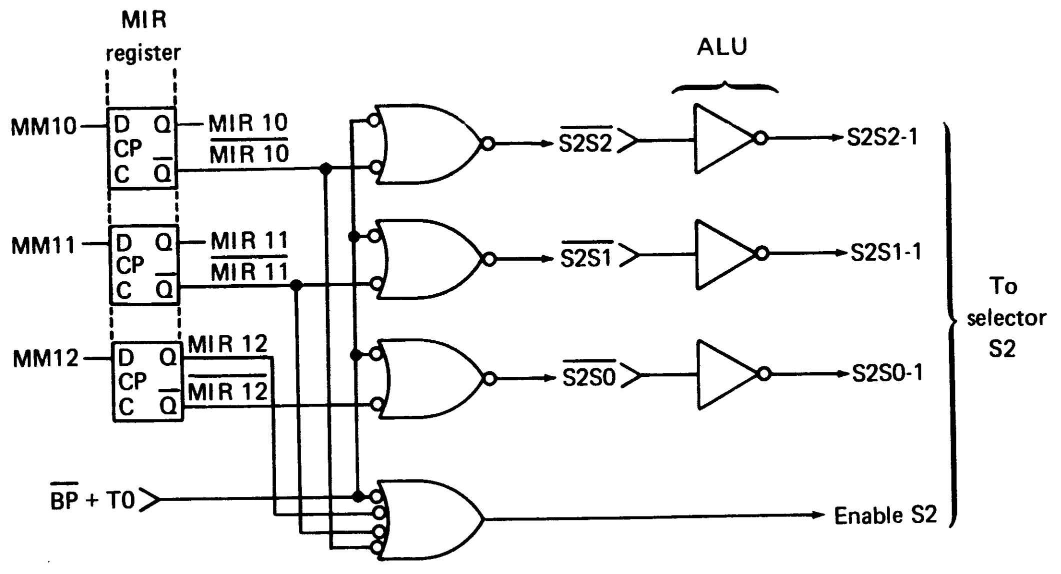


Figure 3-11. B-Field Decoder

F-Field Decoding

F field decoding provides control for ALU arithmetic and logical operations. However, if the F field (bits 2 through 6) should contain either 11110 or 11111, decoding of the A and B fields will control shifting of the A and Q registers. The arithmetic and logical decoder is shown in figure 3-12. When decoded, MIR02 through MIR06 supply control signals to the adder chips located on the ALU module.

Refer to page 15 of the logic diagrams. At location C7, one of the four ALU chips is represented. In an earlier learning activity, adder operation was explained. All that will be said here is that ALUS0-2 through ALUS3-2 signals select the operation (such as an add) which will be performed on four bits of data per adder IC from two different sources. Note the data inputs to the adder. Four bits are supplied through selector S2 and four through selector S1. The adder chip also monitors CARRYIN when an add on a previous chip results in the generation of a carry condition. The output of the adder chip is four data bits and three control signals. $\overline{\text{PROP}}$ and $\overline{\text{GEN}}$ are monitored by the look-ahead carry generator (page 18) while the A = B condition results when A source data equals B source data. The combination of four adder chips and one look-ahead carry generator controls all arithmetic and logical operations except shift. The remaining adder chips are located on pages 6, 9, and 12 of the logic diagrams, while the look-ahead carry generator is on page 18.

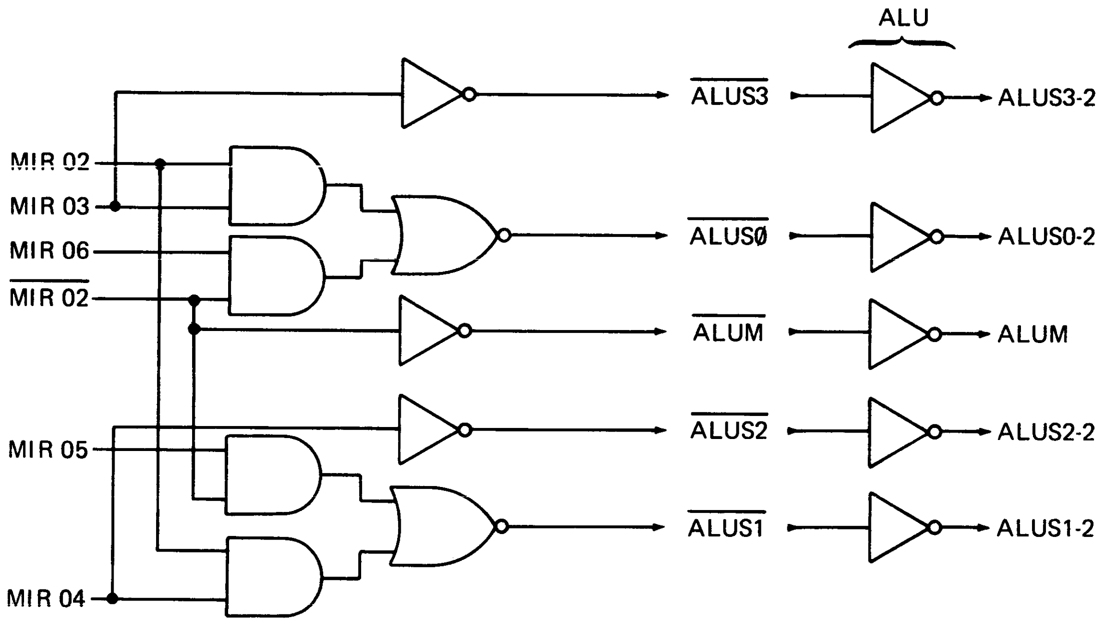

ALU
MIR 02
MIR 03
MIR 06
$\overline{\text{MIR 02}}$
MIR 05
MIR 04
$\overline{\text{ALUS3}}$
$\overline{\text{ALUSØ}}$
$\overline{\text{ALUM}}$
$\overline{\text{ALUS2}}$
$\overline{\text{ALUS1}}$
ALUS3-2
ALUS0-2
ALUM
ALUS2-2
ALUS1-2

Figure 3-12. F-Field Decoder for Arithmetic and Logical Operations

D-Field Decoding

D field decoding gates output data from the adder, selector S3, or selector S1 to some destination register. The destination register may be one of six working registers on the ALU module, the I/O data registers on the SMI module, or the macromemory buffer register. Figure 3-13 represents part of the decoding mechanism for the D field (bits 13, 14, and 15). It consists of two multiplexers; one is used when D decoding is called for; the other, D' decoding. (D' decoding is specified when the DP signal is present; that is, when S equals 1001 or 1010.) Enabling signals are provided to the destination registers, depending on the D-field contents. Some of the enables supplied to the ALU module can be seen on page 7 of the logic diagrams (location A8). The $\overline{\text{GATEQ}}$ and $\overline{\text{GATEX}}$ signals from the decoder are inverted and supplied to the logic components which make up the Q and X registers. These signals enable selector S3's output to either the Q or X registers when present.

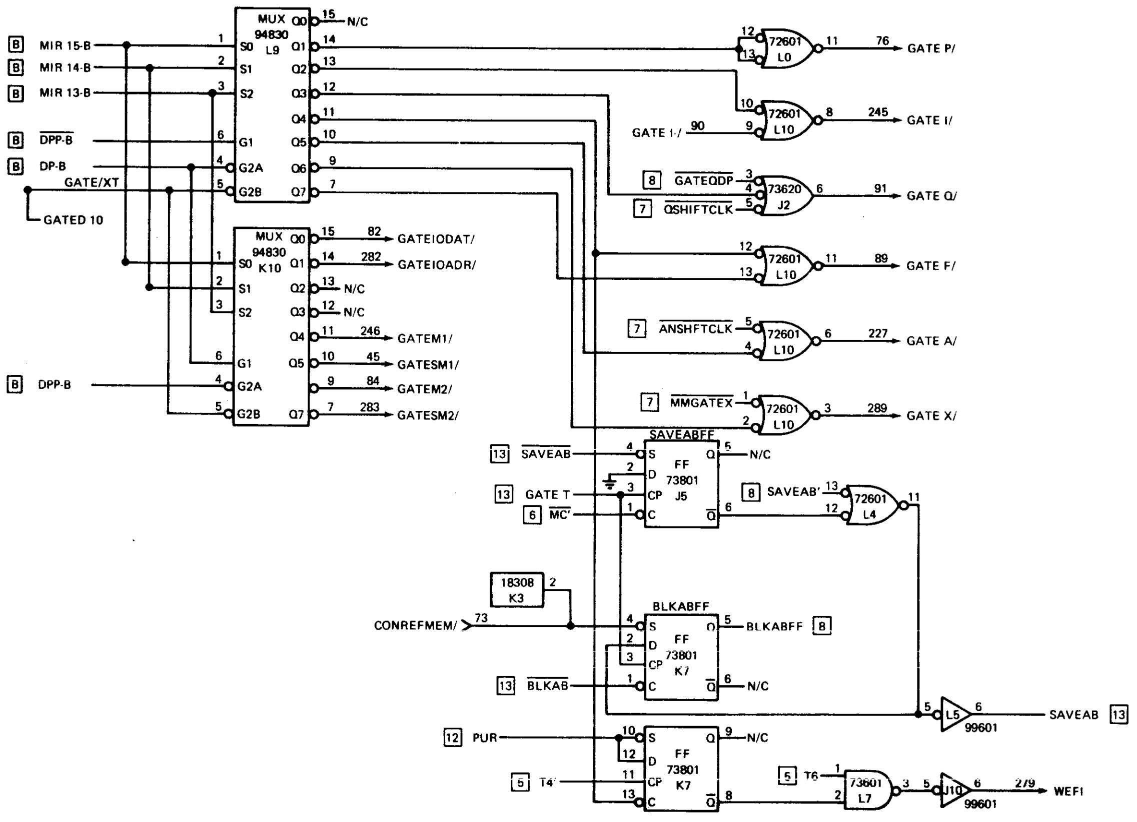


Figure 3-13. D-Field Decoder

Summary

This text has shown how the ALU control field is decoded, the resulting control signals generated, and the functional areas of the ALU affected by these control signals. It should be apparent how the control section of the microprocessor interfaces with the ALU to provide control of arithmetic and logical operations. The next learning activity will contain a closer analysis of the ALU logic diagrams to show how the ALU performs its various operations.

ALU Logic Diagram Data Flow

DIRECTIONS: The following questions are referred to at selected stops in the accompanying audiotape. They are intended to be completed by you as you play the tape. Answer each question by filling in the blanks or picking the best answer from a choice of answers.

1. (a) What enable signal provides gating of output data from selector S3 into the X register? (b) From which field of the microinstruction does this enable originate? ________________________________

2. Pin 7 of logic term JK8 (page 7 of ALU logic diagrams) is labeled Q04. (a) From which logic term does this signal originate? (b) What is its purpose?

   ____

   ____

3. (a) What is the logic term number and pin number of the signal that supplies data bit 4 to the RAM chip, term G3? (b) What working register does this logic term represent? ________________________________

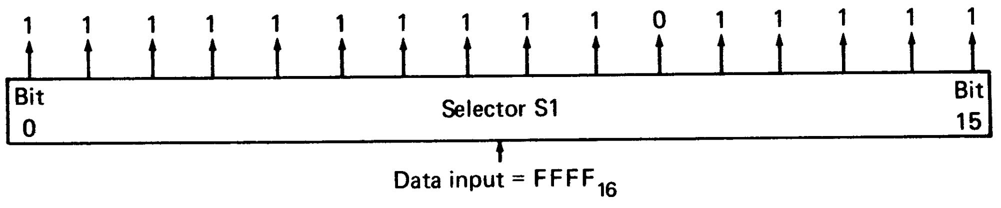
4. Assume the following conditions: 1. Selector S1 select signals enable pin 1 of each multiplexer as the data input source. 2. All working registers equal FFFF hexadecimal. If the output of selector S1 is as indicated below, which logic term or terms would you most likely suspect of failure?

   ____

   ____

   ____

   ____



5. Why does the signal from logic term ALU08, pin 13 (page 12 of the logic diagrams), go to four different multiplexer components of selector S3?

   ____

   ____

   ____

ANSWERS

1. (a) GATEX signal, (b) originates from D-field decoding of microinstruction
2. (a) JK7 on page 10 of the logic diagrams, (b) The Q04 signal is available to JK8 to allow bit shifting during a left-shift operation. The most significant bit of JK7 (Q04) would be shifted into the least significant bit position of JK8, that is, Q03.
3. (a) Logic term H4, pin 11, (b) the F register of the ALU
4. The data is enabled to pin 1 of selector S3 multiplexers in the X register. Because bit 10 is a logic 0 at selector S1's output, either the flip-flop representing bit 10 of the X register (term E4, page 13) is not set or the selector S1 multiplexer (term F5, page 12) has failed to transfer the correct data.
5. There are four types of data transfers performed by selector S3: a one-place right-shift, a one-place left-shift, an eight-place left-shift, and a straight transfer operation. Each ALU output goes to the four selector S3 inputs that allow these transfer conditions.

Macroarithmetic Instruction Execution (Text)

The CYBER 18 computer, an emulation processor, emulates the operations performed by a 1700 computer. The emulation process can be broken down into two major activities:

- Decode the 1700 macroinstruction into a microcode and a micromemory address.
- Decode the resulting microinstructions to enable operations which emulate those performed by the macroinstructions.

Through examples, this learning activity demonstrates the emulation process. Microcode, which produces microprocessor gating, closely duplicates the 1700 instructions of a 1700 computer. For example: first, a 1700 read memory instruction generates the microcode; this, in turn, generates a macromemory read command; as a result, contents of a memory location are placed into a designated working register. The basic process used to emulate a 1700 read memory instruction is demonstrated in figure 3-14.

The 1700 instruction is first read from main memory by a read command contained in the microinstruction labeled "1." It is then gated to the transform module of the processor by a transform command in the microinstruction labeled "2." Three main operations are performed by the transform, with control of these operations derived from the bit arrangement of the 1700 instruction. The three operations performed are as follows.

A. ***Microaddress Formation.*** This address will depend upon the function code of the 1700 macroinstruction. The address selected will contain the emulation program for that instruction.

B. ***MIR Encoding.*** The MIR encode will result in a new ALU control field. When decoded, this controls the formation of an effective address for main memory, from which an operand will be read. It also contains the address of the next microinstruction.

C. ***Delta Translation.*** The delta field of the 1700 instruction is made available to an ALU source (either A or B) and may be used in operand address formation as immediate data or as extended operation code.

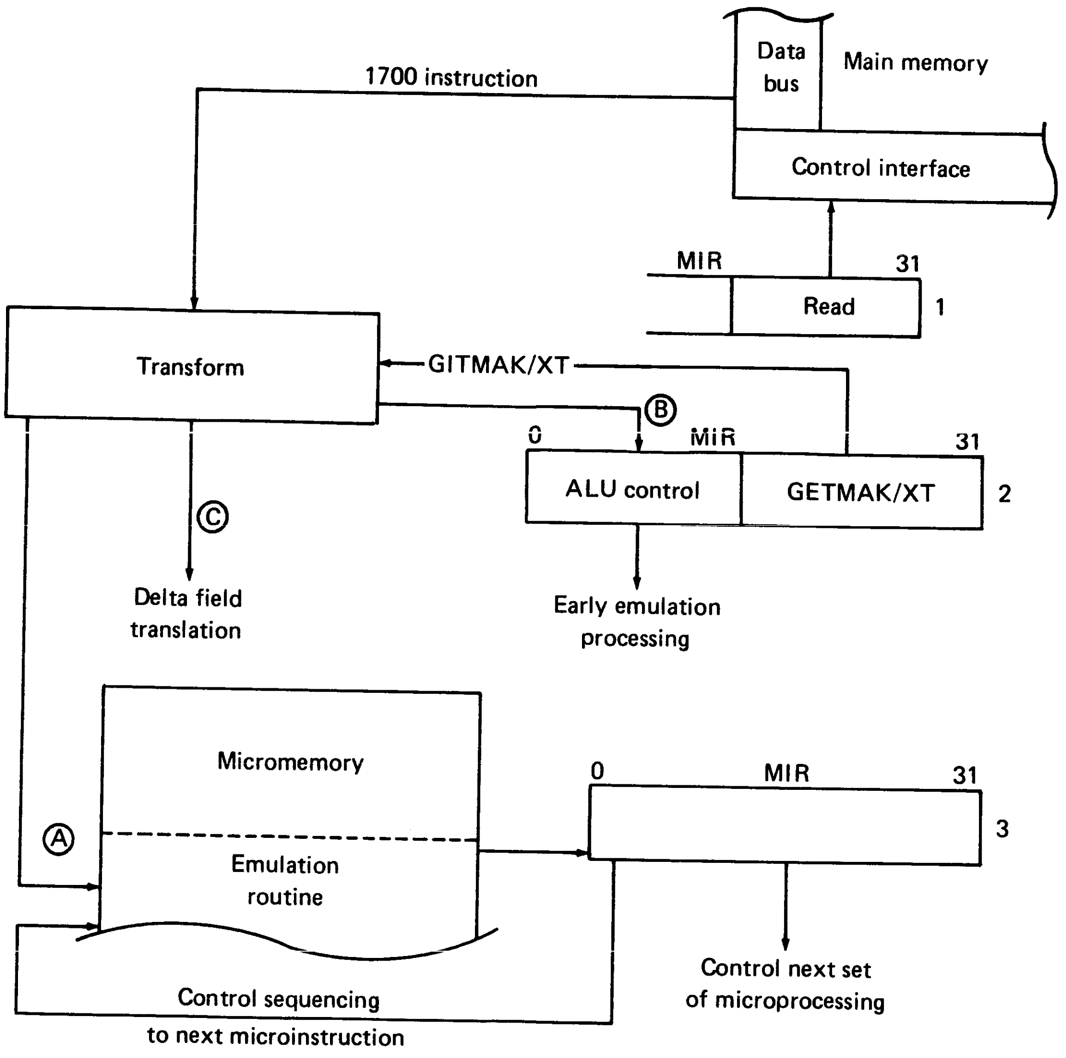


Figure 3-14. Basic Emulation Process for a 1700 Read Memory Operation

Assume, for the sake of our example, that a memory reference 1700 instruction has been read. Once the effective address of main memory has been formed and its contents read, a third microinstruction (number 3) is formed by reading a micromemory location. This read is forced by an enable from the second microinstruction (number 2). Microinstruction number 3 (its address determined by the function code of the 1700 instruction being emulated) commands the contents read from main memory into a specific working register. Another part of the same microinstruction contains information used to locate and read the next address of micromemory. That micromemory location may contain an instruction that is part of the 1700 emulation or the beginning of an RNI routine that will read the next 1700 macroinstruction to be executed (similar to instruction number 1). The number of emulation microinstructions (number 3 and beyond) in an emulation routine is determined by the complexity of the 1700 instruction being emulated.

Sequence of Emulation Activities

A 1700 instruction ultimately controls selection of its own emulation microprogrammed routine. Because of this, the transform process (steps 1, 2, and 3) performed by the microprocessor will vary depending on the function code of the 1700 instruction being emulated. To understand the complete emulation of an instruction, it is necessary to analyze each of the six steps involved in the emulation process:

1. Form the address of the next macroinstruction to be emulated.
2. Read the macroinstruction to be emulated.
3. Select transform options.
4. Perform the transform of the macroinstruction.
5. Execute the emulator program in micromemory.
6. Return to step 1.

Typical of many emulations is the 1700 LOAD Q (LDQ) instruction. When executed, it places the contents of a macromemory location into the Q register of ALU. The steps used to emulate the 1700 LOAD Q instruction are as follows.

Step 1. Form the address of the next macroinstruction to be executed.

This first step is executed prior to reading any macroinstructions. The microinstruction used to accomplish this is contained in micromemory address 058_{16}. This address is read as part of the previous emulation or initial start-up. When read, the instruction at address 058_{16} will increment the P register by 1 to form the macromemory address from which the macroinstruction to be emulated will be read. During this increment, the address will also be entered into AB register in macromemory. Additionally, this same microinstruction will point to the next micromemory address to be read, as shown in figure 3-15. The M-field contents cause the next sequential address, or the lower thirty-two bits of address 058_{16} to be read.

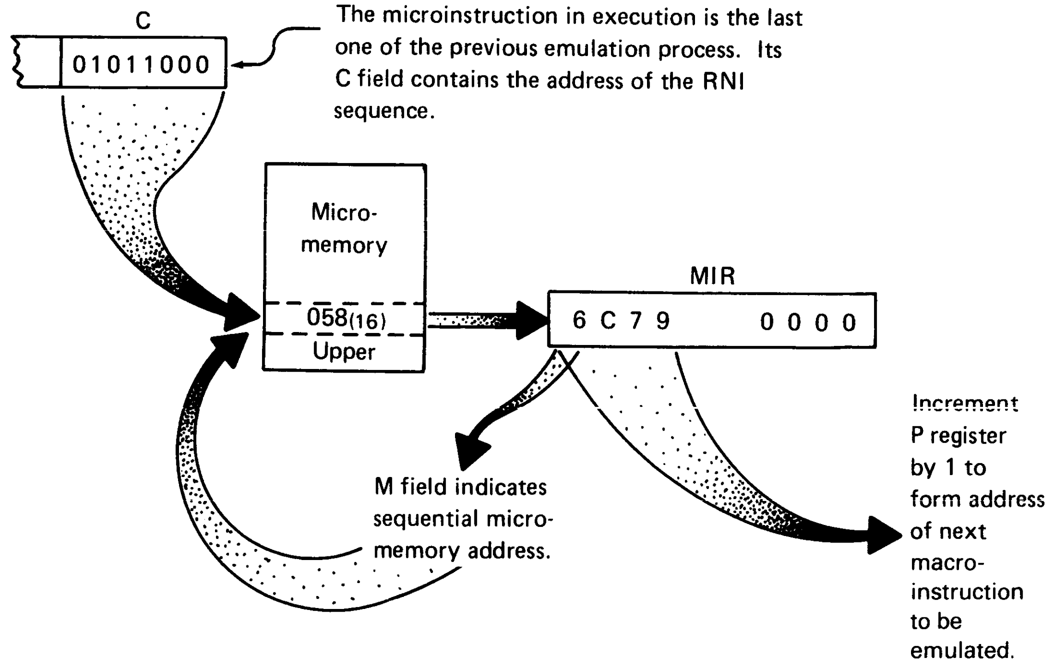

C
01011000
The microinstruction in execution is the last one of the previous emulation process. Its C field contains the address of the RNI sequence.
Micro-memory
058(16)
Upper
MIR
6 C 7 9
0 0 0 0
M field indicates sequential micro-memory address.
Increment P register by 1 to form address of next macro-instruction to be emulated.

Figure 3-15. Increment to Address of Next Macroinstruction (P = P + 1)

Step 2. Read the macroinstruction to be emulated.

Step two of the emulation process shown in figure 3-16 begins when the sequential address of micromemory is read and the microinstruction contained there is gated into the MIR. The ALU control field of this instruction performs no operation, although the M field does indicate that the next instruction read from micromemory will be sequential.

The T, S, and C fields generate control signals that initiate the reading of the macroinstruction to be emulated and the testing (T field) used to detect the presence of interrupts. The presence of an interrupt is the only reason why this instruction would not read memory at this time; the T field of the microinstruction checks for this condition. If an interrupt does exist, the microprogram sequences immediately to a microaddress that will process the interrupt condition before performing the transform command. Assuming that no interrupt condition is present, sequencing is done to read in the transform microinstruction.

The C field of this instruction contains the transform command (GETMAK/XT) that both causes memory data (1700 code) to be entered into IXT and initiates the transform process. (Within the transform hardware, the decoded level GATEIXT is used to enter registers and start processes.)

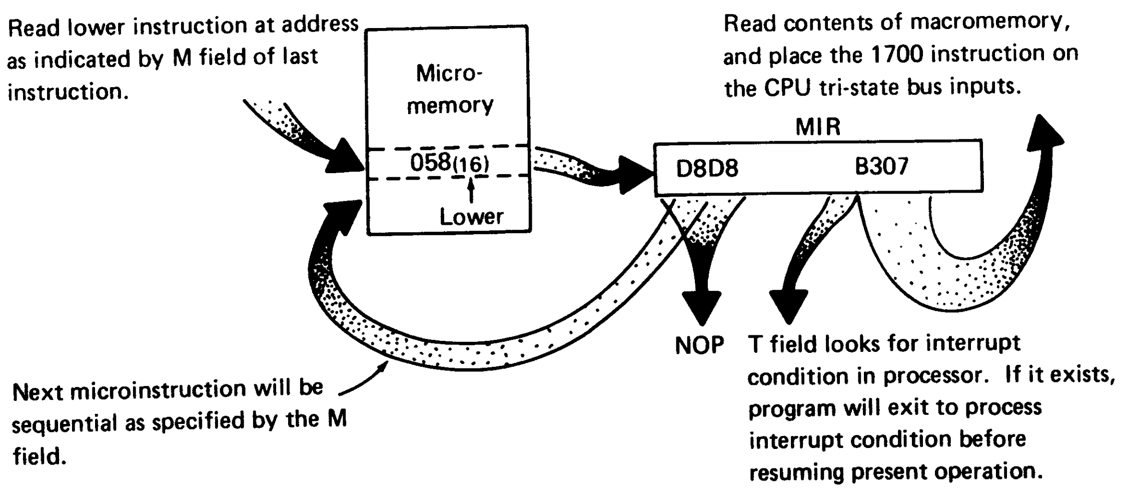

Read lower instruction at address as indicated by M field of last instruction.
Micro-memory
058(16)
Lower
Read contents of macromemory, and place the 1700 instruction on the CPU tri-state bus inputs.
MIR
D8D8
B307
NOP
T field looks for interrupt condition in processor. If it exists, program will exit to process interrupt condition before resuming present operation.
Next microinstruction will be sequential as specified by the M field.

Figure 3-16. Initiate a Read 1700 Instruction from Main Memory and Test for Interrupt

Step 3. Select transform options.

Once the 1700 LOAD Q instruction is contained in IXT, the next microinstruction of the RNI sequence will be read from micromemory. This instruction contains in its C field the transform operation information required for the next operation. A GETMAK/XT transform indicates that the address of the next microinstruction pair to be read from micromemory will be formed as a result of an MA transform. An MA transform will control selector S5 in forming the microaddress. The upper sixteen bits of the instruction shown in figure 3-17 will be replaced by the results of an MIR-encoding operation performed on the macroinstruction being emulated. This operation is demonstrated in step four.

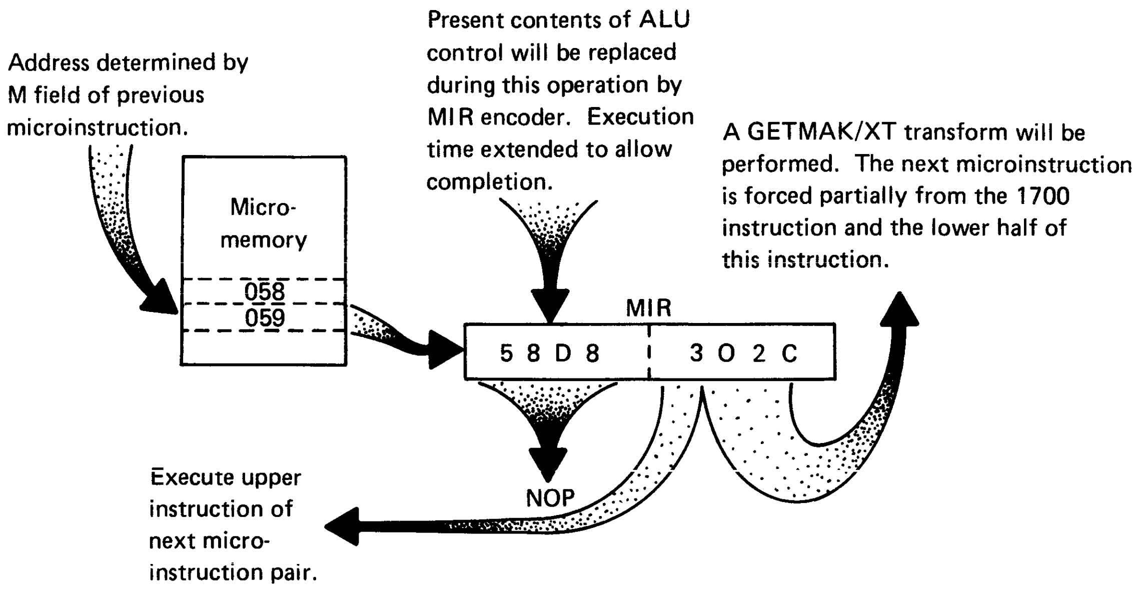

Address determined by M field of previous microinstruction.
Micro-memory
058
059
Present contents of ALU control will be replaced during this operation by MIR encoder. Execution time extended to allow completion.
A GETMAK/XT transform will be performed. The next microinstruction is forced partially from the 1700 instruction and the lower half of this instruction.
MIR
5 8 D 8
3 0 2 C
NOP
Execute upper instruction of next micro-instruction pair.

Figure 3-17. Select Transform Options (per LDQ)

Step 4. Perform the transform of the macroinstruction.

Many activities occur at the same time while the transform operation takes place. First, the micromemory address that contains the LOAD Q emulation program is formed. The formation of this address depends upon what function code bits of the 1700 instruction are being emulated and whether or not that instruction specifies indirect addressing. The transform specified here is a GETMAK/XT transform.

A second operation performed in this step is the encoding of the macroinstruction into the most significant sixteen bits of the MIR. This control information, when decoded, will begin the formation of an effective address for macromemory referencing. From this effective addressed location will come the operand destined for storage in the Q register. The MIR encoder output is determined, as MA was, by the function code of the macroinstruction and its F1 field.

Another operation performed in step 4 is delta field translation. The 1700 instruction being emulated in this example specifies that the effective address of memory is the delta field number. During transform, the delta field is monitored by the delta translator. This translator transfers the lower eight bits of IXT and supplies an additional eight bits depending on the sign bit of the 1700 delta field. These sixteen bits are sent to the ALU module to form the macromemory address. Observing figure 3-18, then, it can be seen that the ALU control field of MIR supplies the control to ALU that uses the delta field contents of the 1700 instruction as the next macromemory address.

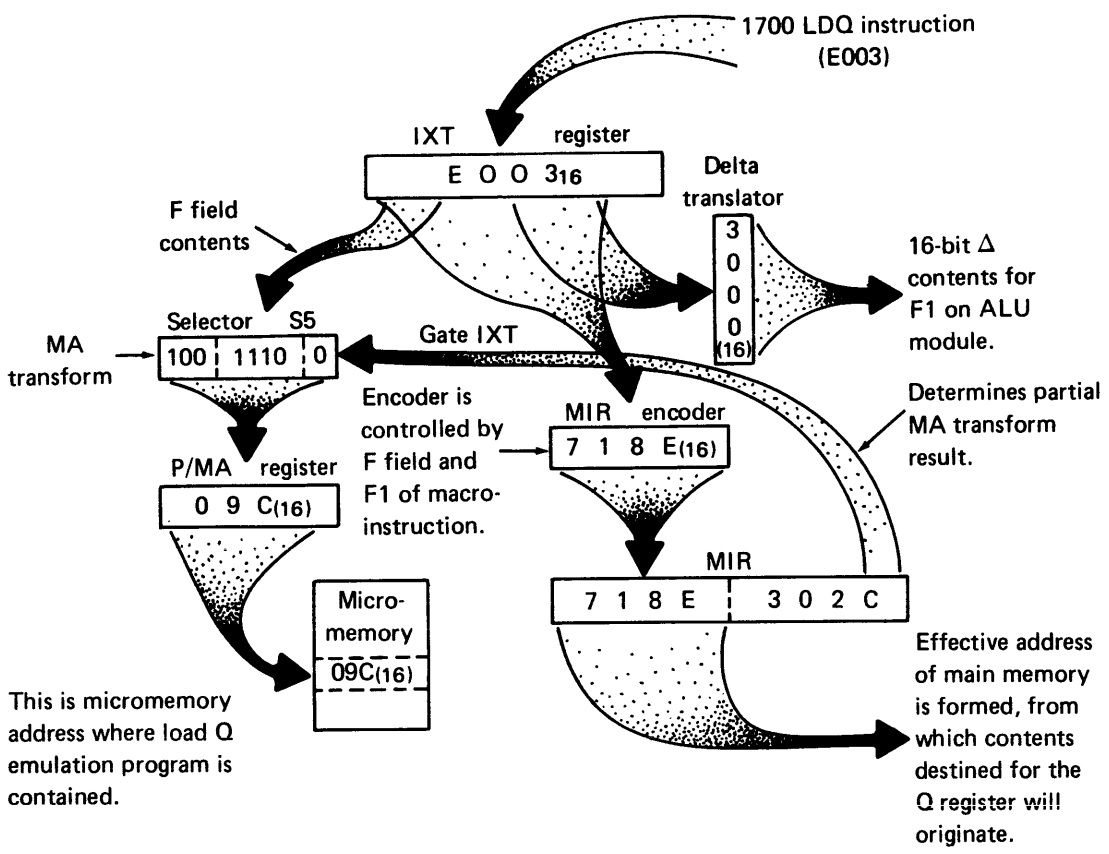

1700 LDQ instruction (E003)
IXT register
E 0 0 3₁₆
Delta translator
3 0 0 0 (16)
16-bit Δ contents for F1 on ALU module.
F field contents
Selector S5
MA transform
100 1110 0
Gate IXT
Encoder is controlled by F field and F1 of macro-instruction.
MIR encoder
7 1 8 E(16)
Determines partial MA transform result.
P/MA register
0 9 C(16)
Micro-memory
09C(16)
MIR
7 1 8 E 3 0 2 C
This is micromemory address where load Q emulation program is contained.
Effective address of main memory is formed, from which contents destined for the Q register will originate.

Figure 3-18. Transform of a 1700 LOAD Q Instruction

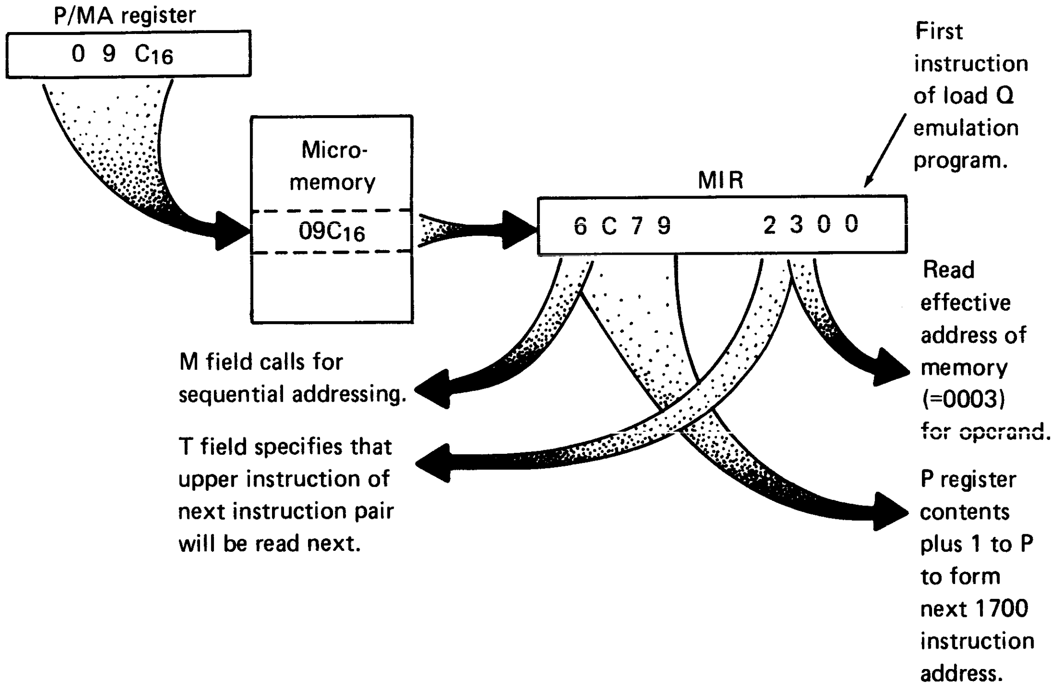

P/MA register
0 9 C16
Micro-memory
09C16
MIR
6 C 7 9 2 3 0 0
First instruction of load Q emulation program.
Read effective address of memory (=0003) for operand.
M field calls for sequential addressing.
T field specifies that upper instruction of next instruction pair will be read next.
P register contents plus 1 to P to form next 1700 instruction address.

Figure 3-19. Execution of Emulation Program from Micromemory

Step 5. Execute the emulation program in micromemory.

It is now possible to use the LOAD Q emulation program stored in micromemory. The contents of microaddress 09C are gated into MIR by the C field of the previous instruction (see figure 3-19). The ALU control field of this new instruction forms the incremented address (P = P + 1) that selects the next macroinstruction for emulation. The LDQ operand read command is contained in the S field of the same instruction. Information contained at the operand's macroaddress will be placed onto the tri-state bus. It will be gated into the Q register when the next microinstruction is executed. It should be noted that the microinstruction presently in execution also determines the next microinstruction to be read. For this (LDQ) routine, the microinstruction will be the upper 32-bit instruction at the next sequential address, that is, 09D.

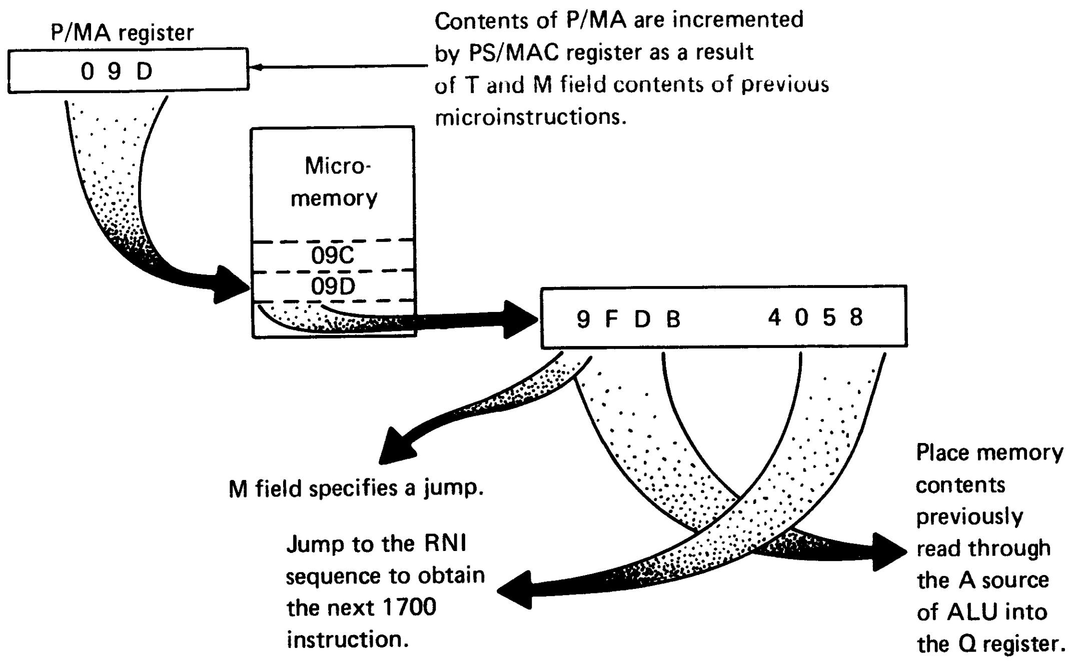

P/MA register
0 9 D
Contents of P/MA are incremented by PS/MAC register as a result of T and M field contents of previous microinstructions.
Micro-memory
09C
09D
9 F D B 4 0 5 8
M field specifies a jump.
Jump to the RNI sequence to obtain the next 1700 instruction.
Place memory contents previously read through the A source of ALU into the Q register.

Figure 3-20. Complete LOAD Q Emulation and Jump to RNI Sequence

Step 6. Complete emulation; go to RNI sequence.

The final operation required to complete a LOAD Q operation is placing the operand obtained from memory into the Q register. Figure 3-20 demonstrates this operation, performed by the upper instruction contained in the address 09D. The ALU control field of the instruction gates memory data from the tri-state bus to the A source of the ALU. The D field of ALU control places the information into the Q register.

The second function of the instruction is to specify the address of micromemory that will be read next. The content of the C field (058) is that address. 058 is the beginning address of the RNI sequence that began our emulation process. The emulation of the 1700 LOAD Q instruction is complete. The RNI sequence will now provide the processor with a new macroinstruction to be emulated, and the process just analyzed will be repeated.

Review of Sequence

The basic sequence of events you have just followed is demonstrated in figure 3-21. When the RUN switch is pressed, the first operation is the formation of a macromemory address that contains the first macroinstruction. Once that instruction is read, the emulation process occurs. This process varies depending upon the type of macroinstruction being emulated, for example, storage reference, register reference, interregister, or skip. Once the emulation process has been completed, the RNI sequence again occurs in order to locate the next macroinstruction to be executed.

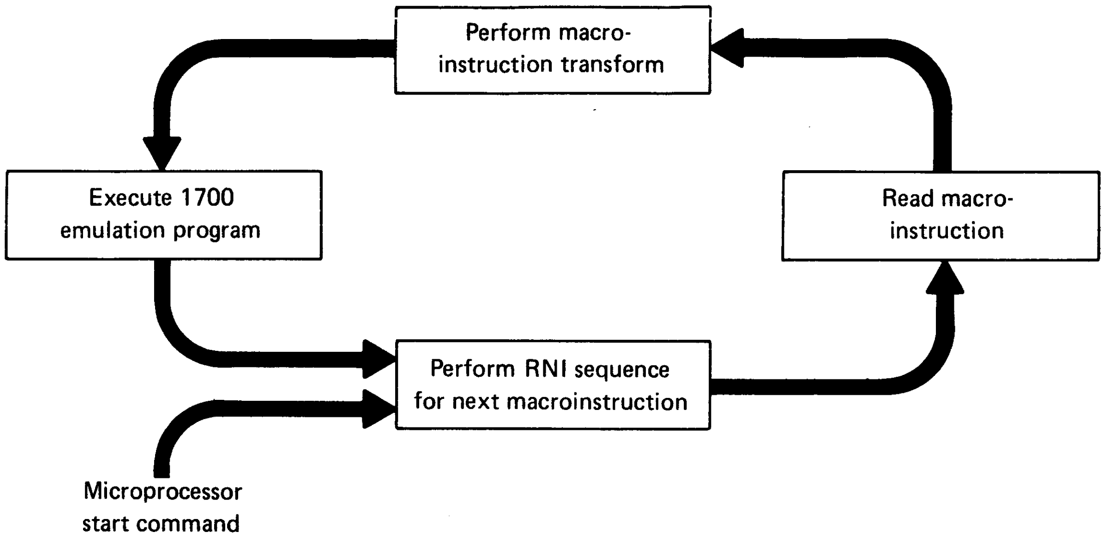

Perform macro-instruction transform
Execute 1700 emulation program
Read macro-instruction
Perform RNI sequence for next macroinstruction
Microprocessor start command

Figure 3-21. Sequence of Events in Emulation Process

Because an emulation process is somewhat complex, it is extremely difficult to demonstrate the complete operation. A close approximation of an entire emulation is shown in figure 3-22, which demonstrates an ADD to A operation. The sequence of operations occurs from top to bottom.

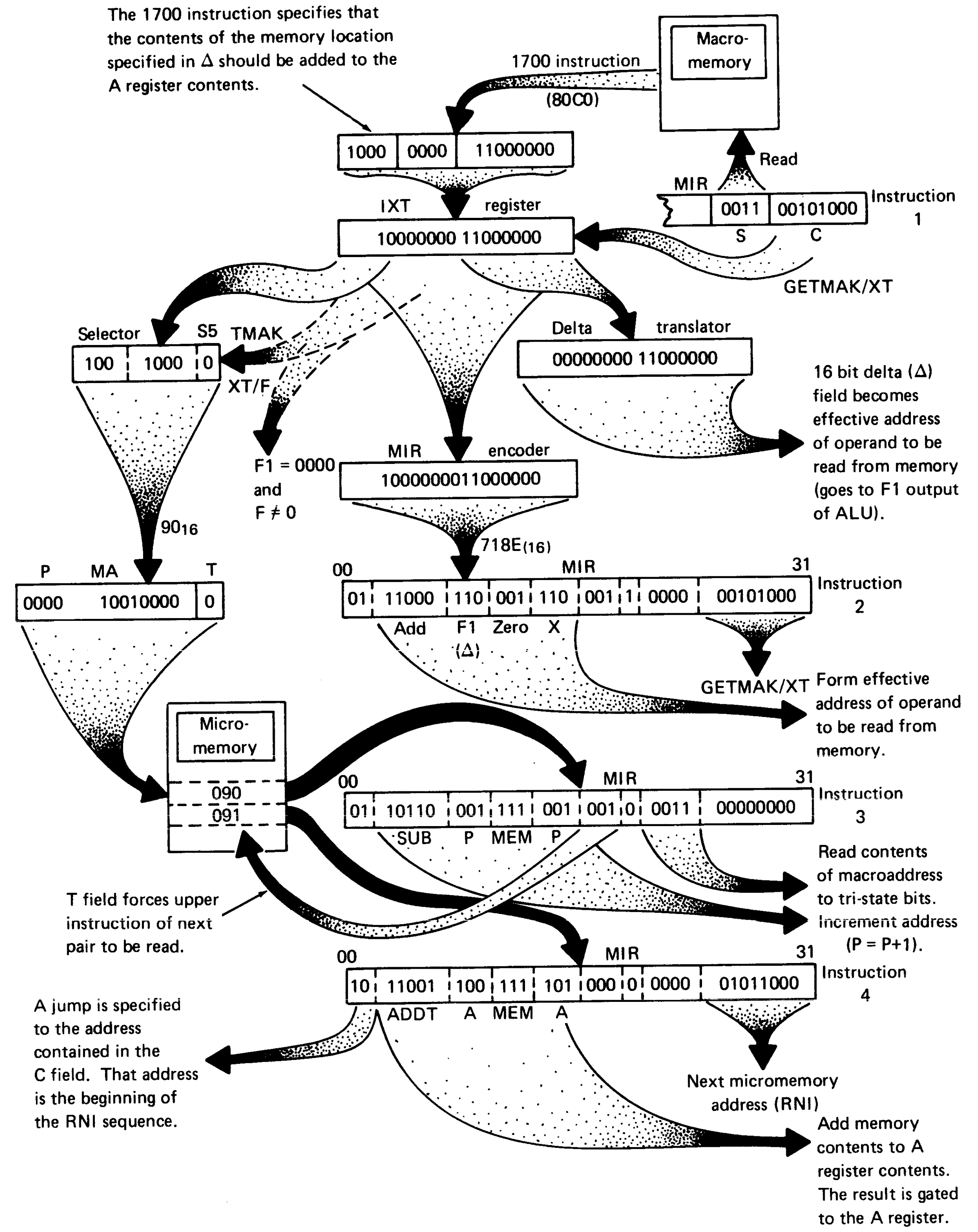

The 1700 instruction specifies that the contents of the memory location specified in Δ should be added to the A register contents.
1700 instruction
(80C0)
Macro-memory
1000 | 0000 | 11000000
Read
MIR
0011 | 00101000
Instruction 1
S
C
GETMAK/XT
IXT register
10000000 11000000
Selector
S5
TMAK
100 | 1000 | 0
XT/F
Delta translator
00000000 11000000
16 bit delta (Δ) field becomes effective address of operand to be read from memory (goes to F1 output of ALU).
F1 = 0000 and F ≠ 0
MIR encoder
1000000011000000
90₁₆
718E(16)
P
MA
T
0000 10010000 | 0
00
MIR
31
01 | 11000 | 110 | 001 | 110 | 001 | 1 | 0000 | 00101000
Instruction 2
Add
F1 (Δ)
Zero
X
GETMAK/XT
Form effective address of operand to be read from memory.
Micro-memory
090
091
00
MIR
31
01 | 10110 | 001 | 111 | 001 | 001 | 0 | 0011 | 00000000
Instruction 3
SUB
P
MEM
P
Read contents of macroaddress to tri-state bits.
Increment address (P = P+1).
T field forces upper instruction of next pair to be read.
00
MIR
31
10 | 11001 | 100 | 111 | 101 | 000 | 0 | 0000 | 01011000
Instruction 4
ADDT
A
MEM
A
A jump is specified to the address contained in the C field. That address is the beginning of the RNI sequence.
Next micromemory address (RNI)
Add memory contents to A register contents. The result is gated to the A register.

Figure 3-22. ADD to A Register Emulation Process

The emulation begins with the reading of the macroinstruction from memory and gating it into the IXT and IXT' registers. This is done during the RNI microroutine. A second microinstruction containing the transform command is read during RNI. The upper sixteen bits of this instruction are not gated into MIR; instead, the output of the MIR encoder is used to provide the arithmetic control information.

At the same time that this occurs, a microaddress is formed for the P/MA register that will select the emulator program. The delta translator also is active at this time. It interprets the lower eight bits of the macroinstruction and generates a resultant sixteen-bit output to the F1 (file 1) output on ALU. The microinstruction decodes the ALU control information obtained from the MIR encoder to form or begin to form (depending on address modifiers r, ind, Q, and I) and an effective address for main memory, since the macroinstruction being emulated (ADD) is a memory reference instruction.

The next step is to gate the first micromemory instruction of the ADD emulator program into MIR. The first instruction (number 3 on figure 3-22) reads the effective address of main memory and determines the next microinstruction to be executed. In the example demonstrated it is the instruction at 091_{16}. The second microinstruction read from micromemory (number 4 on figure 3-22) will add the contents of memory to the A register contents, placing the result of the ADD back into the A register. This same microinstruction also directs the microprogram to exit to the RNI routine at address 058_{16} of micromemory.

It is possible to follow the emulation of any 1700 instruction if the operation of transform is understood. Included in this activity is a theory of transform operation. Given in this section are a series of tables and figures which demonstrate the various transform operations available for the various 1700 instruction types. By analyzing the type of 1700 instruction and determining the type of transform taking place, it is possible to find the micromemory address, the output of the MIR encoder and delta conversion selected. Once these are known, it is only necessary to locate a program listing that shows the emulation programs and their micromemory addresses. The decoding of the emulation program is accomplished by determining the contents of each microinstruction field contained in this program.

Theory of Operation

The 1700 transform with read-only micromemory is used to emulate the Control Data 1700 computer instruction repertoire when it is combined with the basic microprocessor (MP) to form the 1700 enhanced processor. The emulation process includes both hardware and firmware for more efficient operation.

Arithmetic and Logic Operations

The firmware consists mainly of many microcode subroutines that emulate 1700 macroinstructions; therefore, it is also called the 1700 emulator. For each 1700 macroinstruction, there exists a corresponding subroutine required to emulate it. To start the emulation, the macroinstruction is read out from macromemory by a portion of the microprogram. The macroinstruction is then decoded by hardware; this hardware decoder is called the transform. The transform provides the microprogram with the capability to select patterns of bits from the registers and the data transmission path of the MP to form the micromemory address. This micromemory address selects the appropriate microcode subroutine to emulate the macroinstruction. More than one transform operation may be required to completely emulate a macroinstruction. The transform also sets the parameters, generates the microcode needed for the arithmetic and logical operation (refer to MIR Encode) during the emulation process, and sets the contents of the N and K registers.

There are three types of transforms:

- MA transform
- K or N transform
- Combined MA and K transform

The transform commands are coded in the C field of the microinstruction as TMA/j, TK/j, TN/j, GETMAK/j, and GETMAK/XT. The letter j is decoded from the lower four bits (MIR28 through MIR31) of the microinstruction register for the MA transform and the lower three bits (MIR29 through MIR31) for the K and N transform. These bits specify the selector position of selector S5 (MA transform) and of selector S8 (K and N transform). Table 3-2 lists the operations that result when the above transform commands are executed. Figure 3-23 is the block diagram of the 1700 transform module.

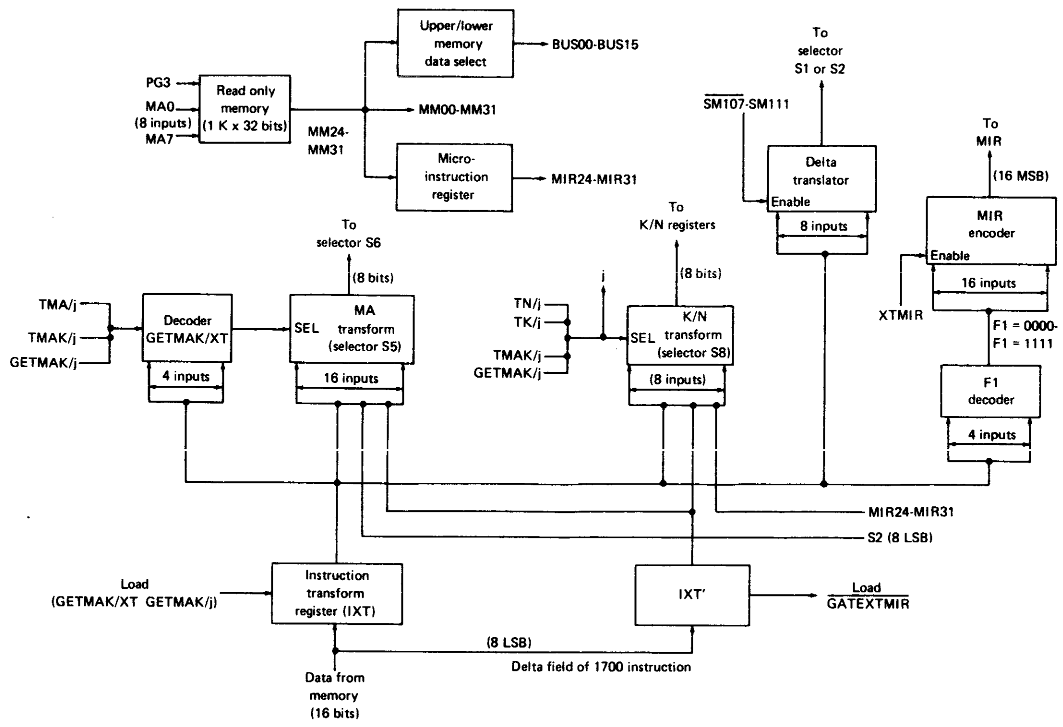


Figure 3-23. 1700 Transform Module Block Diagram

TABLE 3-2
Transform Operations

Mnemonics	Operation
TMA/j	Obtain next microinstruction pair from the address specified by MA transform (selector S5), setting j.
TK/j	Set K register to value specified by K transform (selector S8), setting j.
TN/j	Set N register to value specified by N transform (selector S8), setting j.
TMAK/j	An MA and K transform is executed based on the value of j.
GETMAK/j †	1. Output data from macromemory is gated into the instruction transform (IXT) register. 2. An MA and K transform is executed based on the value of j.
GETMAK/XT †	1. Output data from macromemory is gated into the IXT and IXT' registers. 2. One of eight MA transforms is executed based on the macroinstruction loaded into the IXT register (selected from selector S5, positions 8 through 15). The K register is always transformed from S8, position 7. 3. The most significant 16 bits of the microinstruction register (MIR) are loaded with a microcommand encoded from the macroinstruction residing in the IXT register. This operation is referred to as MIR transform (XT/MIR). The least significant 16 bits of MIR are loaded from micro-memory.

† These commands must be executed in the microinstruction following a read command.

IXT Register

The IXT register is a sixteen-bit register that holds the macroinstruction currently being emulated. The IXT register consists of D-type flip-flops A3, J2, C5, and B5. It receives its input directly from macromemory DFM01 through DFM16 (DFM01 is the least significant bit and DFM16 is the most significant bit). The output of the IXT register, I00 through I15 (I00 is the most significant bit and I15 is the least significant bit), is sent to selector S5, selector S8, the delta translator, and MIR encode to be transformed. The IXT register output is also sent to GETMAK/XT decoder to generate the proper control signals for selector S5 during a GETMAK/XT operation. The IXT register is loaded by executing a macromemory read microinstruction followed by a microinstruction with a GETMAK/j or GETMAK/XT in the C" field. Refer to the $\overline{\text{GATEIXT}}$ signal in the control 1 module. Otherwise, IXT can be loaded by executing a microinstruction with C' code equal to 011xxxx to generate a general-purpose strobe at time T4. Refer to the $\overline{\text{GATEBKP}}$ signal in control 2.

IXT' Register

The IXT' register is an eight-bit register that holds the least significant eight bits (delta field) of the 1700 macroinstruction. The IXT' register is utilized mainly for emulation of 1700 enhanced instructions that have doubleword format. The IXT' register consists of D-type flip-flops C4 and B4, which receive their inputs directly from macromemory DFM01 through DFM08. The data from macromemory is gated into the IXT' register by $\overline{\text{GATE XTMIR}}$, which is generated only during the GETMAK/XT command. The output of the IXT' register is sent to selectors S5 and S8 to be transformed.

Selector S5

Selector S5 is an eight-bit wide selector that is used to form micromemory addresses. It consists of 16-to-1 multiplexers L2, L3, L4, L5, L7, L9, L10, and L12, which allow sixteen different micromemory address (MA) transforms to be specified. The eight-bit transformed MA specifies one of the 256 64-bit micromemory words within a page. The selector position is determined by the letter j of TMA/j, TMAK/j, and GETMAK/j, or, depending upon the macroinstruction, via transform hardware (GETMAK/XT command).

Figure 3-24 shows sixteen different MA transforms. MA transforms 0 through 7 and 9 through 12 are used to emulate 1700 enhanced instructions. MA transforms 8 through 15 are used to emulate the basic 1700 instructions. The 1's and 0's are generated by handwiring to +5v or ground, respectively. Other patterns of bits are derived from registers IXT, IXT', the lower eight bits of selector S2, and special conditions such as protect violation or indirect address mode, which are decoded from the macroinstruction. Table 3-3 lists the different MA transforms applied for different types of macroinstructions and for different addressing modes. Whenever the GETMAK/XT command is executed, one of the eight MA transforms (8 through 15) is selected, based on the macroinstruction being emulated. Tables 3-4, 3-5, and 3-6 show the MA transforms for the basic 1700 storage reference instructions ($F \neq 0$), basic 1700 register reference instructions, and interregister reference instructions, respectively. The MA transforms for the enhanced instructions are selected via XT/F1 MA transform 13. The position-select signals, S5-S0 through S5-S3, of selector S5 are described in the following section.

Instruction	j Value	Transform output 0	1	2	3	4	5	6	7
XT/INT	0	1	0	1	1	1	1	S2 (11)	A†
XT/IR2	1	1	0	1	1	1	IXT' (11-12)		0
XT/F3A	2	1	1	0	1	1	IXT' (13-15)		
XT/DEST	3	0	0	1	1	1	IXT' (13-15)		
XT/F3*	4	1	1	1	IXT' (11-15)				
	5	0	0	0	0	0	0	0	0
XT/F2	6	1	0	1	0	IXT' (8-9)		Δ'= 0	0
XT/S2	7	S2, Lower 8 bits (08 through 15)							

Instruction	j Value	Transform output 0	1	2	3			6	7
XT/SK	8	1	1	0	IXT' (08-11)				0
XT/SH or XT/DRP	9	1	0	1	1	0	IXT' (08-10)		
XT/IR or XT/F3	A	1	1	1	0	IXT' (12-15)			
XT/F or XT/F4	B	1	0	0	IXT' (00-03)				0
XT/IM or XT/SKIP 2	C	1	1	1	1	IXT' (08-11)			
XT/F1	D	0	1	F= 0	IXT (04-07)				Δ'= 0
XT/F1*	E	1	0	1	0	0	B†	IXT (06-07)	
XT/FM	F	1	0	1	1	1	0	C†	D†

† A = (IXT = 0500) +S209
= IIN instruction or false attempt
B = (F1-00xx) + MULTILEVEL INDIRECT MODE
C = Protect violation
D = (Δ' = 0) + B

Figure 3-24. MA Transforms

TABLE 3-3
MA Transform Applications

Instruction	MIR28–MIR31 = j	Application
XT/INT	0	Micro/macro interrupt
XT/IR2	1	Interregister type 2 instruction
XT/F3A	2	Field instruction
XT/DEST	3	Register destination
XT/F3	4	Miscellaneous instruction
	5	Not used
XT/F2	6	F2 (address mode) for enhanced instruction
XT/S2	7	Selector S2 (lower eight bits); normally used for the breakpoint panel
XT/SK	8	Skip instruction
XT/SH or XT/DRP	9	Shift instruction, or decrement and repeat instruction
XT/IR or XT/F3	A	Interregister instruction with M not the origin, or miscellaneous instruction
XT/F or XT/F4	B	F (OP CODE) field, or OP CODE for storage reference type 2 and field instruction
XT/IM or XT/SKIP2	C	Interregister with M origin, or skip instruction type 2
XT/F1	D	F1 (address mode) field
XT/F1	E	Alternate F1 field
XT/FM	F	Miscellaneous F1 field

TABLE 3-4
1700 Storage Reference Transforms During GETMAK/XT Operation

Mode	F1 (Binary)	Hexadecimal	Delta	Instruction	MIR Transform
Absolute	0000	0	≠0	XT/F	Δ → X,AB
Constant			=0	XT/F1	P + 1 → P, AB
Absolute	0001	1	≠0	XT/F	Δ + (00FF) → X, AB
Constant			=0	XT/F1*	P + 1 → P, AB
Absolute	0010	2	≠0	XT/F	Δ + (Q) → X, AB
Constant			=0	XT/F1*	P + 1 → P, AB
Absolute	0011	3	≠0	XT/FM	Δ + (00FF) → X, AB
Constant			=0	XT/F1*	P + 1 → P, AB
Indirect	0100	4	≠0	XT/F1*	Δ → X, AB
Storage			=0	XT/F1*	P + 1 → P, AB
Indirect	0101	5	≠0	XT/F1*	Δ → X, AB
Storage			=0	XT/F1*	P + 1 → P, AB
Indirect	0110	6	≠0	XT/F1*	Δ → X, AB
Storage			=0	XT/F1*	P + 1 → P, AB
Indirect	0111	7	≠0	XT/F1*	Δ → X, AB
Storage			=0	XT/F1*	P + 1 → P, AB
Relative	1000	8	≠0	XT/F	P + Δ(SE)* → X, AB
16-bit relative			=0	XT/F1	P + 1 → P, AB
Relative	1001	9	≠0	XT/F1	P + Δ(SE) → X, AB
16-bit relative			=0	XT/F1	P + 1 → P, AB
Relative	1010	A	≠0	XT/F1	P + Δ(SE) → X, AB
16-bit relative			=0	XT/F1	P + 1 → P, AB
Relative	1011	B	≠0	XT/F1	P + Δ(SE) → X, AB
16-bit relative			=0	XT/F1	P + 1 → P, AB
Relative indirect	1100	C	≠0	XT/F1*	P + Δ(SE) → X, AB
Relative indirect			=0	XT/FM	P + 1 → P, AB
Relative indirect	1101	D	≠0	XT/F1*	P + Δ(SE) → X, AB
Relative indirect			=0	XT/FM	P + 1 → P, AB
Relative indirect	1110	E	≠0	XT/F1*	P + Δ(SE) → X, AB
Relative indirect			=0	XT/FM	P + 1 → P, AB
Relative indirect	1111	F	≠0	XT/F1*	P + Δ(SE) → X, AB
Relative indirect			=0	XT/FM	P + 1 → P, AB

* SE = Sign Extended

TABLE 3-5
1700 Register Reference Transforms During GETMAK/XT Operation

F1 (Binary)	Instruction	MIR Transform	Comment
0000	XT/F1	NOP	Selective stop (Δ=0) Instruction enhanced ($\Delta \neq 0$)
0001	XT/SK	$P + 1 \rightarrow P$, AB	Skip
0010	XT/F1	$P + \Delta(SE) \rightarrow F$, AB	Input to A
0011	XT/F1	$P + \Delta(SE) \rightarrow F$, AB	Output from A
0100	XT/F1	NOP	Enable interrupt (Δ=0) Instruction enhanced ($\Delta \neq 0$)
0101	XT/F1	NOP	Inhibit interrupt (Δ=0) Instruction enhanced ($\Delta \neq 0$)
0110	XT/F1	$Q \rightarrow X$, AB	Set program protect (Δ=0) Instruction enhanced ($\Delta \neq 0$)
0111	XT/F1	$Q \rightarrow X$, AB	Clear program protect (Δ=0) Instruction enhanced ($\Delta \neq 0$)
1000	†	†	Interregister
1001	XT/F1	$P + 1 \rightarrow P$, AB	Increase A
1010	XT/F1	$P + 1 \rightarrow P$, AB	Enter A
1011	XT/F1	NOP	Pass (Δ=0) Instruction enhanced ($\Delta \neq 0$)
1100	XT/F1	$P + 1 \rightarrow P$, AB	Enter Q
1101	XT/F1	$P + 1 \rightarrow P$, AB	Increase Q
1110	XT/F1	$\Delta(INT) \rightarrow X$, AB	Exit interrupt
1111	XT/SH	$A \rightarrow F$	Shift

† See 1700 Interregister Transforms.

Select Signals S5-S0 through S5-S3

During TMA/j, TMAK/j, and GETMAK/j transform operations, position select signals S5-S0 through S5-S3 directly correspond to MIR28 through MIR31. During the GETMAK/XT transform operation, select signals S5-S0 through S5-S3 are generated based on the microinstruction being emulated. Multiplexer K12 selects one of the above cases depending upon the state of the GETMAK/XT signal at K12-1. The output of the IXT register is first decoded to select the MA transform according to figure 3-25 for the storage reference instructions (F ≠ 0) and according to figure 3-26 and table 3-5 for the register reference instructions and interregister reference instructions (F = 0) during the GETMAK/XT operation. The F = 0 signal at pin 1 of multiplexer D10 selects the MA transforms as follows:

F = 0 low	S5-S0 through S5-S3 are generated from the storage reference instruction
F = 0 high	S5-S0 through S5-S3 are generated from the register reference and interregister reference instructions

The MA transform selection shown in figure 3-26 is performed simultaneously by the combination circuit. These flow charts are used to show the selection conditions rather than the sequential steps in selecting the MA transforms.

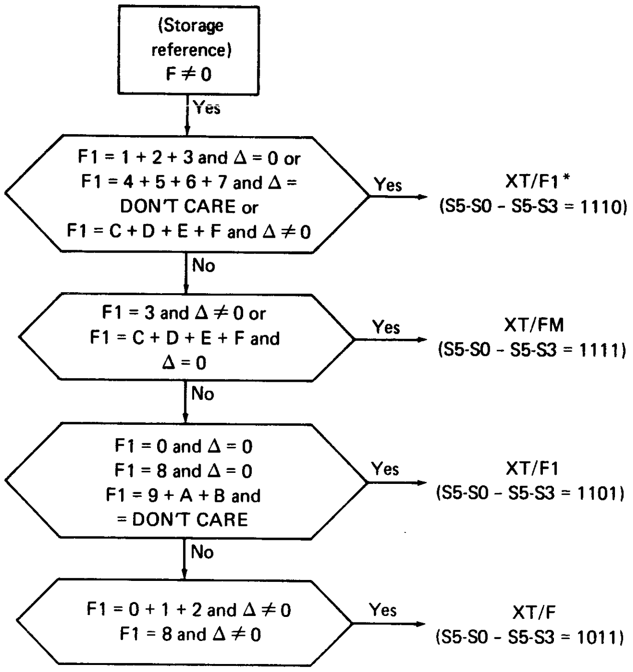

(Storage reference) F≠0
Yes
F1 = 1 + 2 + 3 and Δ = 0 or F1 = 4 + 5 + 6 + 7 and Δ = DON'T CARE or F1 = C + D + E + F and Δ ≠ 0
Yes
XT/F1* (S5-S0 - S5-S3 = 1110)
No
F1 = 3 and Δ ≠ 0 or F1 = C + D + E + F and Δ = 0
Yes
XT/FM (S5-S0 - S5-S3 = 1111)
No
F1 = 0 and Δ = 0 F1 = 8 and Δ = 0 F1 = 9 + A + B and = DON'T CARE
Yes
XT/F1 (S5-S0 - S5-S3 = 1101)
No
F1 = 0 + 1 + 2 and Δ ≠ 0 F1 = 8 and Δ ≠ 0
Yes
XT/F (S5-S0 - S5-S3 = 1011)

Figure 3-25. MA Transform Selection For 1700 Storage Reference Instructions

TABLE 3-6
1700 Interregister Transforms During GETMAK/XT Operation

Instruction				Condition							
XT/IR XT/IM				M not origin register (I12 ≠ 0) M is origin register (I12 = 0)							
MIR Transform (MIR Fields)				Conditions							
						Origin			Destination		
F Bits 2-6	A Bits 7-9	B Bits 10-12	D Bits 13-15	LP 18	XR 19	A I10	Q I11	M I12	A I13	Q I14	M I15
ADDT 11001	–	–	–	0	0	X	X	0	X	X	X
A B 01110	–	–	–	1	0	X	X	0	X	X	X
A + B 01001	–	–	–	0	1	X	X	0	X	X	X
(-A) + (-B) 00001	–	–	–	1	1	X	X	0	X	X	X
ADD+ 11010	P regis- ter 001	Zeros 001	P reg- ister † 001	X	X	X	X	1	X	X	X
–	Ones 110	–	–	X	X	0	X	0	X	X	X
–	A regis- ter 100	–	–	X	X	1	X	0	X	X	X
–	–	Ones 110	–	X	X	X	0	0	X	X	X
–	–	Q regis- ter 100	–	X	X	X	1	0	X	X	X
–	–	–	NOP 000	X	X	X	X	0	0	0	0
–	–	–	A reg- ister † 101	X	X	X	X	0	1	X	X
–	–	–	Q reg- ister † 011	X	X	X	X	0	0	1	X
–	–	–	F regis- ter 111	X	X	X	X	0	0	0	X

† NOP if protect violation detected.

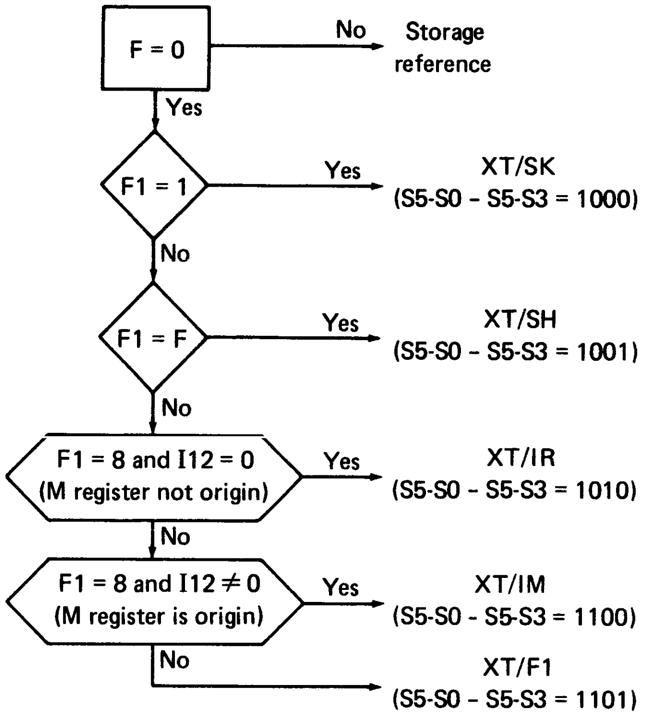


Figure 3-26. MA Transform Selection for 1700 Register Reference and Interregister Reference Instructions

Selector S8

Selector S8 is an eight-bit wide selector that is used to choose between a maximum of eight different sources for loading the N and K registers. Figure 3-27 shows eight K/N transform assignments. The 0's are generated by directly connecting to ground. Other bit patterns are derived from the lowest eight bits (S208 through S215) of selector S2, the IXT and IXT' registers, and the lowest eight bits (MIR24 through MIR31) of the microinstruction register.

Selector S8 consists of eight 8-to-1 multiplexers, K1, K2, K3, K4, K5, K6, K7, and K9, which are enabled only when the $\overline{\text{S8ENABLE}}$ signal at pin 7 is low. The $\overline{\text{S8ENABLE}}$ signal is high to disable S8 during the clear N register (CLRN), clear K register (CLRK), and clear N and page register (CLRNP) commands, allowing all 0's to be loaded into the N or K register. The position select signals S8-S0 through S8-S2 are generated by multiplexer H9.

Instruction	j Value	Transform output (0 … 7)
XT/S2	0	S2, lowest 8 bits (08-15)
XT/SHCNT	1	0 | 0 | 0 | IXT' (11-15)
XT/FLDLTH	2	0 | 0 | 0 | 0 | IXT (04-07)
XT/RA	3	0 | 0 | 0 | 0 | 0 | IXT' (10-12)
XT/RA*	4	0 | 0 | 0 | 0 | 0 | IXT' (08-10)
XT/RB	5	0 | 0 | 0 | 0 | 0 | IXT' (13-15)
XT/MIR	6	MIR, lowest 8 bits (24-31)
XT/FLDSTR	7	0 | 0 | 0 | 0 | IXT (00-03)

Figure 3-27. K/N Transforms

Table 3-7 indicates that if $\overline{\text{MODE11}}$ is high (sequential address mode, MIR00 and MIR01 = 11), selector S8 will be at position 6 to allow MIR24 through MIR31 to be loaded directly into the K or N register. If MODE11 is low and MIR28 is low, S5-S2 through S5-S0 correspond directly to MIR29 through MIR31. However, if MODE11 is low and MIR28 is high (GETMAK/XT operation), K/N transform 7 is selected for all combinations of MIR29 through MIR31.

TABLE 3-7
Position Select Signal Generation

$\overline{\text{MODE 11}}$	MIR28	S5S2	S5S1	S5S0
0	0	MIR29	MIR30	MIR31
0	1	1	1	1
1	X	1	1	0

Selector S7

Selector S7 is a one-bit wide selector that allows up to sixteen different external and/or internal conditions to be tested to determine which upper or lower microinstruction to execute from the next microinstruction pair.

Table 3-8 shows the conditions to be tested by the 1700 emulator during the emulation process.

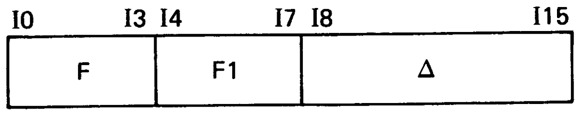
The test bit is selected by the lowest four bits of the microinstruction register (MIR28 through MIR31). The output $\overline{\text{BTU}}$ is sent to the T-field test multiplexer in the control 2 module and is tested if the T field of the microinstruction contains a BTU command.

TABLE 3-8
Emulation Test Conditions

Test Bit	Operation	Selector S7	
		Pin	Position
BTU00	Not assigned	8	0
I02	Execute upper microinstruction if I02 is a 1.	7	1
I07	Execute upper microinstruction if I07 is a 1.	6	2
I06	Execute upper microinstruction if I06 is a 1.	5	3
IND00FF	Execute upper microinstruction if STORE 00FF (index 1) status is true.	4	4
SM105/ $\overline{\text{(PROTECT FAULT)}}$	Execute lower microinstruction if storage protect fault is detected.	3	5
SELSTOP	Execute lower microinstruction if selective stop switch is set.	2	6
SELSKIP/	Execute lower microinstruction if selective skip switch is set.	1	7
SM108 (PARITY ERROR)	Execute lower microinstruction if storage parity error is detected.	23	8
BTU00	Not assigned	22	9
DELTA'=0	Execute upper microinstruction if delta equals 0 (LXT8 through IXT15=0).	21	10
$\overline{\text{EA=OPER}}$	Execute lower microinstruction if the effective address equals the operand.	20	11
EVENPAR	Execute upper microinstruction if memory parity line is true (even parity).	19	12
I00	Execute upper microinstruction if 100 is a 1.	18	13
MULTIND	Execute upper microinstruction if multilevel indirect address mode is selected.	17	14
SM101+SM108	Execute upper microinstruction if previous macromemory write cycle was aborted (caused either by parity error or protect fault).	16	15

MIR Encode

The 1700 instruction format is repeated here to help in understanding the signal mnemonics:



During the GETMAK/XT command the macroinstruction is encoded to form the upper sixteen bits, MM00 through MM15, which can be loaded directly to the micro-instruction register. These upper sixteen bits of MIR control the arithmetic functions for read next instruction (RNI) cycles and other required operations to provide more efficient execution. The types of MIR transform based on the macroinstruction being emulated are shown in tables 3-4, 3-5, and 3-6 for storage reference instructions, register reference instructions, and interregister instructions, respectively. Figures 3-28 and 3-29 show the MIR transform selection for storage reference instructions and for the register reference instructions as flow charts. Table 3-6 shows the MIR transform for interregister reference instructions.

The MIR transform selection is performed simultaneously by the combination circuit. Figures 3-28 and 3-29 show the selection conditions rather than the sequential steps in selecting the MIR transform.

The 2-to-1 multiplexers G6, E6, E5, and G5 select the MIR transform either for storage reference instructions (F = 0 at multiplexer input pin 1 is low) or for register reference and interregister reference instructions ($\overline{\text{F = 0}}$ at multiplexer input pin 1 is high). These multiplexers are enabled by the $\overline{\text{XTMIR}}$ signal, which is only generated during GETMAK/XT operation. If the protect violation is detected, the D field of MIR encode is set to 0000 (NOP).

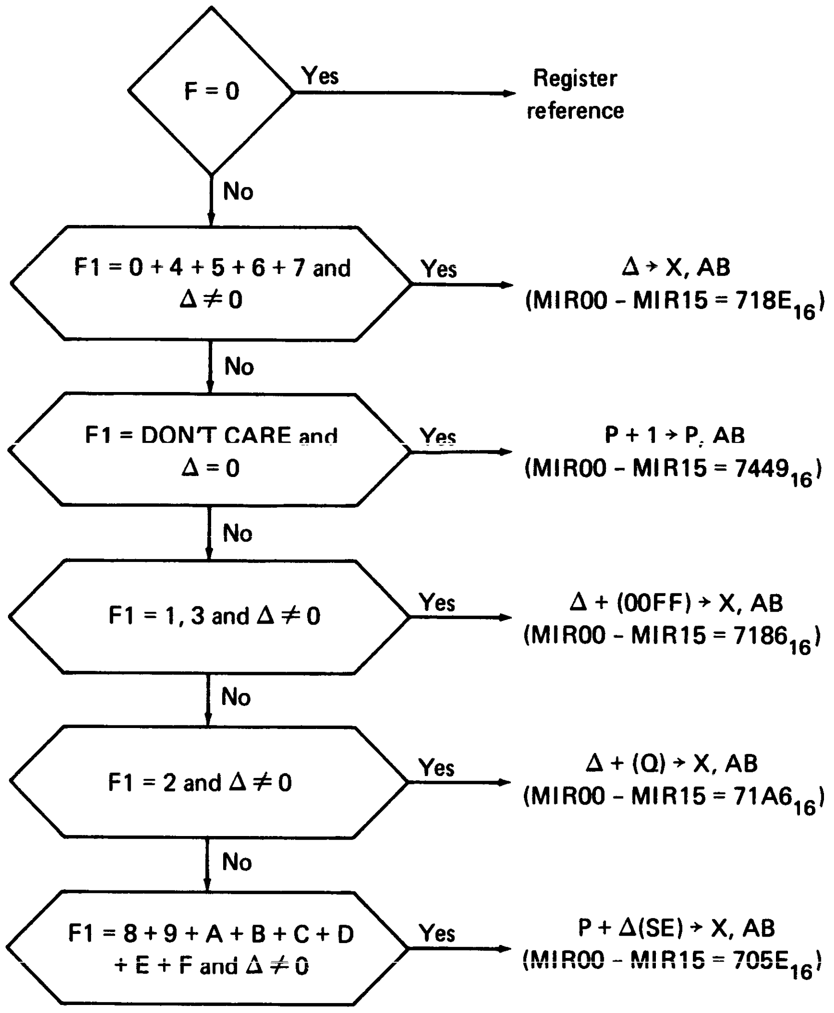
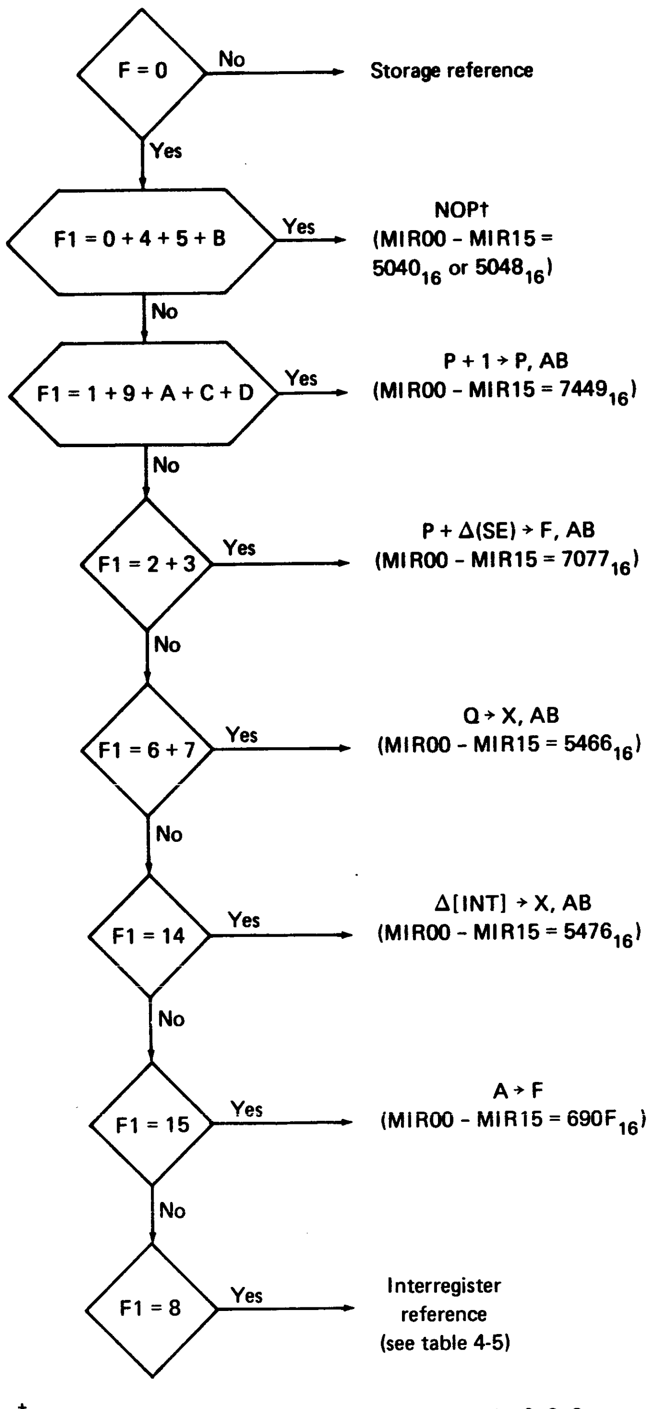

F = 0
Yes
Register reference
No
F1 = 0 + 4 + 5 + 6 + 7 and Δ ≠ 0
Yes
Δ → X, AB (MIR00 - MIR15 = 718E16)
No
F1 = DON'T CARE and Δ = 0
Yes
P + 1 → P, AB (MIR00 - MIR15 = 744916)
No
F1 = 1, 3 and Δ ≠ 0
Yes
Δ + (00FF) → X, AB (MIR00 - MIR15 = 718616)
No
F1 = 2 and Δ ≠ 0
Yes
Δ + (Q) → X, AB (MIR00 - MIR15 = 71A616)
No
F1 = 8 + 9 + A + B + C + D + E + F and Δ ≠ 0
Yes
P + Δ(SE) → X, AB (MIR00 - MIR15 = 705E16)

Figure 3-28. MIR Transform
of 1700 Storage Reference Instructions



† NOP = The D field of the microinstruction must be 0 0 0.

Figure 3-29. MIR Transform of 1700 Register Reference Instructions

Delta Translator

To emulate certain basic and enhanced 1700 instructions, the delta field of the macroinstruction must be modified before it is used in operations indicated by MIR transform. Table 3-9 shows the conditions and modified delta fields.

The delta translator consists mainly of a combination of circuits that translates the delta field according to the types of macroinstructions being emulated, as in table 3-5. The 2-to-1 multiplexers, A9, A10, C9, and C10, are enabled only when:

- SM111 is not set (disable the F1 output to selector S1 or S2 and enable the output of the delta translator to S1 or S2).
- SM107 is not set (disable decimal arithmetic correction logic).
- F = 0 or F1 = 1000 are low (not an interregister reference instruction).

When an interregister reference instruction is emulated, the above multiplexers are disabled. This causes the delta ($FFFF_{16}$) to be sent to selector S1 or S2. The select signal at input pin 1 is generated as follows:

$$SM213 \cdot F = 0 \cdot (F1 = 0 + 1 + 6 + E)$$

If the above select signal is high, it allows Δ(SK) and Δ(INT) to be selected from position 1 of the multiplexers. Δ and Δ(SE) are selected from position 0 when the select signal is low. Status mode bit SM213 is set by the emulator only during emulation of enhanced instructions; i.e., type 2 storage reference instructions and field reference instructions.

TABLE 3-9
Delta Translations

Conditions	Delta (Δ)
a. (F=0) (F1 = 0xxx) b. Enhanced instructions: (F=0) (F1 = 4 + 5) (r = 0)[†]	Δ = 0 0 0 0 0 0 0 0 I08 I09 I10 I11 I12 I13 I14 I15
a. (F=0) (F1 = 1xxx) b. (F=0) (F1= 2 + 3) c. Enhanced instructions: (F=0) (F1 = 4 + 5) (r = 0)[†]	Δ(SE) (with sign extend) = $\underbrace{\text{C C C C C C C C}}_{\text{Constant = I08}}$ I08 I09 I10 I11 I12 I13 I14 I15
a. (F=0) (F1=1) b. (F=0) (F1= 0 + 6)	Δ(SK) (for skip instruction) = 0 0 0 0 0 0 0 0 0 0 0 0 I12 I13 I14 I15
a. (F=0) (F1=E)	Δ(INT) (for interrupt instruction = 0 0 0 0 0 0 0 1 I08 I09 I10 I11 I12 I13 I14 I15
a. (F=0) (F1=8)	Δ(FFFF) = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[†]Refer to the CYBER 18 processor reference manual for type 2 storage reference instructions, enhanced instructions, and field reference isntructions. Flag r is the relative address flag represented by IXT'08.

Read Only Micromemory

The micromemory of the 1700 transform module is a read only memory that has been preprogrammed with the 1700 instruction emulator. This micromemory consists of 512 64-bit words (two pages). Each word consists of two microinstructions that are referred to as upper (32 bit) and lower (32 bit). Each word in the micromemory is addressed by the memory address bits (MA0 through MA7, PG3). The MA0 through MA7 specify one of 256 words (microinstruction pairs, 32-bit upper and 32-bit lower instruction within a page. The page (PG3) selects the page (page 0 or 1) in which the instruction resides. The output of the read only memory is coupled to the upper/lower memory data select where the sequence of selection (upper or lower) is determined in accordance with the emulation required. The selected instruction is transferred via the CPU three-state bus to the X register.

Macroarithmetic Instruction Execution (Exercise)

DIRECTIONS: Mark the following questions T for true or F for false.

______ 1. The function code of the 1700 instruction being emulated forms part of the micromemory address containing the emulation program.

______ 2. The first operation normally performed during an emulation process is an RNI sequence.

______ 3. During the emulation of a 1700 storage reference instruction, the formation of an effective address is controlled by the output of the delta field decoder.

______ 4. The sixteen-bit output of the MIR encoder is determined by the 1700 instruction's function code and address modifiers.

______ 5. When a 1700 storage reference instruction is emulated, the MA portion of the P/MA register will be equal to the F field of the 1700 instruction.

DIRECTIONS: Complete the following statement.

6. To be emulated, a 1700 instruction must first be gated into the IXT register. The microinstruction field that controls this gating is____________________.

ANSWERS

1. T 2. T 3. F 4. T 5. F 6. C FIELD

PROGRESS CHECK

QUESTIONS

1. Addressing information needed to access memory locations in file 2 originates from the ______ register.

   a. K
   b. N
   c. X
   d. P/MA

2. D' decoding of the D field will occur if the S field of the microinstruction equals ______.

   a. 0011
   b. 1000
   c. 1001
   d. 1111

3. Which transform operation results in the formation of a new ALU control field in the MIR?

   a. RNI encoding
   b. Microaddress formation
   c. MIR encoding
   d. Delta translation

4. A 1700 instruction is gated into the IXT register by a transform command contained in the ______ field of a microinstruction.

   a. A
   b. S
   c. C
   d. T

5. Information for "N" and "K" is selected from one of eight possible sources by selector ______.

   a. S2
   b. S5
   c. S7
   d. S8

6. The read only micromemory containing the 1700 instruction emulator consists of ______ pages.

   a. one
   b. two
   c. three
   d. four

7. The B field of the microinstruction controls ______.

   a. selector S1
   b. selector S2
   c. selector S3
   d. the adder

8. When the RUN switch is pressed, the first operation performed is ______.

   a. a transform operation
   b. execution of a micromemory emulation program
   c. decoding the ALU control field of the microinstruction
   d. reading a macromemory from memory

9. Shift functions in the A register are controlled by ______ enabling the signal(s).

   a. gate X
   b. gate P
   c. AMODESO 0 and AMODESO 1
   d. AP/+TO

10. Which statement concerning field decoding is true?

   a. D field decoding determines the select inputs of selector S3 multiplexes
   b. The B source to the ALU is designated by the T field contents
   c. A' decoding of the A field enables data sources on the tri-state bus as A source data
   d. S field decoding normally controls micromemory address sequencing

ANSWERS

1. Correct Answer: b
   Resource: Text: CYBER 18-20 Arithmetic Logic SRM, page 3-2.

2. Correct Answer: c
   Resource: Text: CYBER 18-20 Arithmetic Logic SRM, page 3-18.

3. Correct Answer: c
   Resource: Text: CYBER 18-20 Arithmetic Logic SRM, page 3-22.

4. Correct Answer: b
   Resource: Text: CYBER 18-20 Arithmetic Logic SRM, page 3-25.

5. Correct Answer: d
   Resource: Text: CYBER 18-20 Arithmetic Logic SRM, page 3-43.

6. Correct Answer: b
   Resource: Text: CYBER 18-20 Arithmetic Logic SRM, page 3-50.

7. Correct Answer: b
   Resource: Text: CYBER 18-20 Arithmetic Logic SRM, page 3-15.

8. Correct Answer: d
   Resource: Text: CYBER 18-20 Arithmetic Logic SRM, page 3-31.

9. Correct Answer: c
   Resource: Text: CYBER 18-20 Arithmetic Logic SRM, page 3-1.

10. Correct Answer: c
    Resource: Text: CYBER 18-20 Arithmetic Logic SRM, page 3-14.

CYBER 18-20 ARITHMETIC LOGIC

LEARNING GUIDE

CONTROL
DATA

Pub. No. 76770502

1/85

CONTENTS

INTRODUCTION

The arithmetic logic unit does the actual work of the computer: adding, L-subtracting, and comparing quantities. In this unit, you study the operating details of the CYBER 18-20 ALU, its working registers, and the path taken by data as it goes through the ALU.

You examine how a microinstruction's arithmetic code field is decoded, as well as the sequence of events in the ALU during instruction execution.

Whenever you are asked to look at logic diagrams in this unit, those diagrams can be found in the CYBER 18-20 Logic Circuits Reference Manual.

Resources

- CDC 110 Terminal with disk drive.
- PLATO course disk ct-cpu2,
  Control Data Corporation, pub. no. 76773086.
- Audiotape: "ALU Logic Diagram Data Flow,"
  Control Data Corporation, pub. no. 76362465.

BLOCK 1: ALU REGISTERS

The main parts of the arithmetic logic unit (ALU) are described in this block. In addition, each working register and its function is described and explained. As you examine the ALU, you will follow the flow of data through it.

___1-A ALU FUNCTIONAL AREAS

This activity describes the basic operation, components, and data flow of the ALU.

Objective

- Identify the functional areas of the arithmetic logic unit (ALU), and name the ALU working registers.

Resource

Text/ Reading	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "ALU Functional Areas," pages 1-1 through 1-9.

___1-B ALU WORKING REGISTERS

This activity describes the registers that are used in ALU operations.

Objective

- Identify the functional areas of the arithmetic logic unit (ALU), and name the ALU working registers.

Resource

Text/ Reading	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "ALU Working Registers," pages 1-10 through 1-12.

___1-C ALU ORGANIZATION

This exercise checks your understanding of each functional block on the ALU module.

Objective

- Identify the functional areas of the arithmetic logic unit (ALU), and name the ALU working registers.

Resource

Text/ Exercise	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "ALU Organization," page 1-13.

__1-D ALU OPERATION

This activity examines the basic construction and organization of the ALU and the look-ahead carry generator.

Objective

- Define the function of each ALU register.

Resource

Text/ Reading	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "ALU Operation," pages 1-14 through 1-20.

__1-E ALU OUTPUT SELECTOR AND REGISTER DATA PATHS

This activity describes selector S3 and the data paths it controls, and reviews the working areas of the ALU.

Objective

- Follow the ALU register input and output data paths.

Resource

Text/ Reading	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "ALU Output Selector and Register Data Paths," pages 1-21 through 1-26.

__1-F ALU DATA FLOW

This activity explains the following types of data flow within the arithmetic logic module: data transfer, arithmetic operations, logic operations, and shift operations. It also explains the ALU control field of the microinstruction.

Objective

- Follow the ALU register input and output data paths.

Resource

CBE	"ALU Data Flow" (PLATO course disk ct-cpu2)

__1-G ALU REGISTER DATA PATHS

This exercise checks your understanding of ALU register data paths.

Objective

- Follow the ALU register input and output data paths.

Resource

Text/ Exercise	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "ALU Register Data Paths," page 1-27.

__1-H PROGRESS CHECK

At this point you should check your understanding of the material in this block by answering the progress check questions.

Resource

Text/ Exercise	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "Progress Check," pages 1-29 and 1-30; answers, page 1-31.

BLOCK 2: MICROINSTRUCTION DECODING

This block examines microinstructions in detail from a hardware viewpoint. The microinstruction's control field and the functions of each bit within the ALU control field are described. You study the ALU data flow and the various operating modes controlled by microinstruction control field bits.

_2-A ALU CONTROL FIELD FUNCTIONS

This activity lists the meaning of the bits that appear in the ALU control fields of a microinstruction.

Objective

- Identify the function of each bit in the ALU control field.

Resource

Text/ Reading	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "ALU Control Field Functions," pages 2-1 through 2-19.

_2-B ALU CONTROL FIELDS

In this exercise, you are shown several examples of ALU control fields and are asked questions concerning the effects they would initiate.

Objective

- Identify the function of each bit in the ALU control field.

Resource

Text/ Exercise	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "ALU Control Fields," pages 2-20 through 2-22.

_2-C ALU CONTROL FIELD OPERATIONS

This activity explains how each subfield determines sources of input data, destination of output data, and control of shift operations.

Objective

- Identify the ALU input and output data paths that are active by examining the microinstruction control field.

Resource

Text/ Reading	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "ALU Control Field Operations," pages 2-23 through 2-30.

__2-D ALU DATA FLOW AND OPERATING MODES

This exercise checks your understanding of control field operations.

Objective

- Identify the ALU input and output data paths that are active by examining the microinstruction control field.

Resource

Text/ Exercise	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "ALU Data Flow and Operating Modes," page 2-31.

__2-E ARITHMETIC OPERATIONS

This activity analyzes the microinstruction program listing, showing how to interpret the steps performed by a microinstruction.

Objective

- Identify which arithmetic operation is being commanded, by examining microinstruction control field bits.

Resource

Text/ Reading	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "Arithmetic Operations," pages 2-32 through 2-40.

__2-F PROGRESS CHECK

At this point you should check your understanding of the material in this block by answering the progress check questions.

Resource

Text/ Exercise	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "Progress Check," pages 2-41 and 2-42; answers, page 2-43.

BLOCK 3: ARITHMETIC AND LOGIC OPERATIONS

In this block, you follow the sequence of events in the ALU logic diagrams during instruction execution. By the end of the block, you should be able to list the sequence of events that occurs during the execution of a macroarithmetic instruction.

_3-A ALU OPERATIONS

This activity examines six functional areas of the arithmetic logic unit: the P register, the A register, file 2, selector S1, ALU with look-ahead carry, and selector S3 with shift control.

Objective

- Describe the purpose of the ALU.

Resource

Text/ Reading	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "ALU Operations," pages 3-1 through 3-12.

_3-B ALU INSTRUCTION EXAMPLE

In this activity, you look at the decoding process for the ALU control field and the signals that result from this decoding.

Objective

- Follow an ALU instruction sequence through the logic diagrams.

Resource

Text/ Reading	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "ALU Instruction Example," pages 3-13 through 3-19.

_3-C ALU LOGIC DIAGRAM DATA FLOW

This tape helps you locate some of the more important functional areas that are part of the ALU and explains how these areas operate. To complete this activity, you need the logic diagrams for the ALU and a functional block diagram with logic page references. As you listen to this audiotape, answer the questions found in the exercise activity.

Objective

- Follow an ALU instruction sequence through the logic diagrams.

Resources

Audio	"ALU Logic Diagram Data Flow"
Text/ Exercise	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "ALU Logic Diagram Data Flow," pages 3-20 and 3-21.

__3-D MACROARITHMETIC INSTRUCTION EXECUTION (TEXT)

This activity describes the sequence of events occurring during the emulation of a 1700 macroinstruction.

Objective

- List the sequence of events that occurs during the execution of a macroarithmetic instruction.

Resource

Text/ Reading	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "Macroarithmetic Instruction Execution (Text)," pages 3-22 through 3-50.

__3-E MACROARITHMETIC INSTRUCTION EXECUTION (EXERCISE)

This exercise checks your understanding of macroarithmetic instruction execution.

Objective

- List the sequence of events that occurs during the execution of a macroarithmetic instruction.

Resource

Text/ Exercise	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "Macroarithmetic Instruction Execution (Exercise)," page 3-51.

__3-F PROGRESS CHECK

At this point you should check your understanding of the material in this block by answering the progress check questions.

Resource

Text/ Exercise	CYBER 18-20 Arithmetic Logic Supplementary Reference Manual, "Progress Check," pages 3-53 and 3-54; answers, page 3-55.