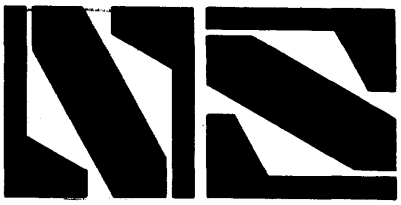
Pub. No. 4200005A



PRODUCT DESCRIPTION

GENERAL PURPOSE

CONTROLLER/PROCESSOR (GPC/P)

MOS/LSI SYSTEM KIT

National Semiconductor
2900 Semiconductor Drive,
Santa Clara, Ca. 95051

Pub. No. 4200005A

PRODUCT DESCRIPTION

General Purpose Controller/Processor (GPC/P)
MOS/LSI System Kit

March 1972

CONTENTS

		Page No.
1.0	INTRODUCTION	1-1
2.0	EXAMPLE OF SYSTEM APPLICATION	2-1
2.1	Functional Description of Example System	2-1
2.2	System Logic Partitioning	2-4
2.3	System Timing and Data Flow	2-4
3.0	MOS/LSI DEVICE DESCRIPTIONS	3-1
3.1	RALU Functional Description	3-1
3.2	CROM Functional Description	3-8
4.0	GENERAL PURPOSE MACROINSTRUCTION SET	4-1
4.1	Introduction	4-1
4.2	Notation	4-1
4.3	Memory Addressing	4-3
4.4	Instruction Set Synopsis	4-4
4.5	Execution Time	4-9
4.6	Initialization, Interrupt Processing	4-10

ILLUSTRATIONS

Figure No.		Page No.
2-1	Functional Block Diagram of "Example" System	2-2
2-2	"Example" System Logic Partitioning	2-5
2-3	GPC/P Timing	2-6
3-1	RALU Simplified Functional Block Diagram	3-2
3-2	CROM Simplified Functional Block Diagram	3-9
3-3	Loading ROM Address Register (RAR)	3-10

TABLES

Table No.		Page No.
3-1	Definition of Abbreviations Used on Diagrams	3-3
3-2	ALU Function Bits	3-4
3-3	Control-Function Bits	3-5
3-4	ROM Bit Assignments	3-12
4-1	Notation Used in Descriptions of IMP-16 Instructions . .	4-1
4-2	Instruction Execution Time	4-9

1.0 INTRODUCTION

The past several years have seen a dramatic increase in the use of small-scale digital computers in dedicated control applications. The advantages of adaptability to a variety of applications, ease of change (by program modification), proven design, and simplified interface have given the system manufacturer an attractive alternative to special-purpose designs. Unfortunately, costs have limited the application of this approach to systems whose end-user sales price is the order of $30,000 or more. In the future, however, the economies that are offered by MOS/LSI will allow the system manufacturer to apply the "computer controlled" concept to a whole new class of systems - whose sales price may be as much as an order of magnitude below that of present applications.

The GPC/P (General Purpose Controller/Processor) System Kit offers the system designer a versatile digital processor that he may customize to provide features for his particular application. Using this approach, system manufacturing costs are reduced due to lower cost of packaging (fewer connectors, printed circuit boards, cables, and so forth), smaller power supply, lower assembly costs, more modest cooling requirements, and greater standardization of hardware. Product variations may be achieved by program change rather than by logic design modifications. This results in a reduction of the number of different circuit board types, and thus yields significant indirect cost savings and lower inventory requirements.

Development costs for the customer who uses the GPC/P are modest because the MOS/LSI circuits are standard products, in contrast to custom LSI devices that require time-consuming development effort. Another cost-reduction benefit realized by using the GPC/P large-scale system building blocks is the minimization of the need for "gate-level" detailed logic design. The designer is thus free to apply more of his efforts toward a higher level of system design.

2.0 EXAMPLE OF SYSTEM APPLICATION

Although the GPC/P System Kit consists of general-purpose system building blocks that may be adapted to a wide variety of uses, they are described here in terms of a specific system implementation. This allows the reader to gain a better appreciation of the utility of the devices. The system described may be applied without modification by many users: the design is proven, and software aids are available to ease program development. However, it should be kept in mind that many variations to this system are possible for special application needs.

2.1 Functional Description of Example System

The "example" system is functionally equivalent to a small-scale general-purpose digital computer. The system has a 16-bit word and uses parallel, binary arithmetic. Figure 2-1 is a simplified functional block diagram of the example system. Seven conventional 16-bit general-purpose registers are provided. The application described here uses the registers in the following fashion:

- Program Counter (PC)
- Memory Data Register (MDR)
- Memory Address Register (MAR)
- 4 General-purpose working registers (AC0, AC1, AC2, and AC3) - accessible to the macroprogrammer

In one basic machine cycle (hereinafter called a microcycle), any two of these registers may be operated upon by the Arithmetic and Logic Unit (ALU), and the result then stored into one of the registers. In addition, a 16-word Last-In-First-Out (LIFO) stack is available. The top entry of the stack may be "pulled" (i.e., "popped") and gated into one of the inputs (A) of the ALU, and/or the result (R) of the ALU may be "pushed" onto the stack.

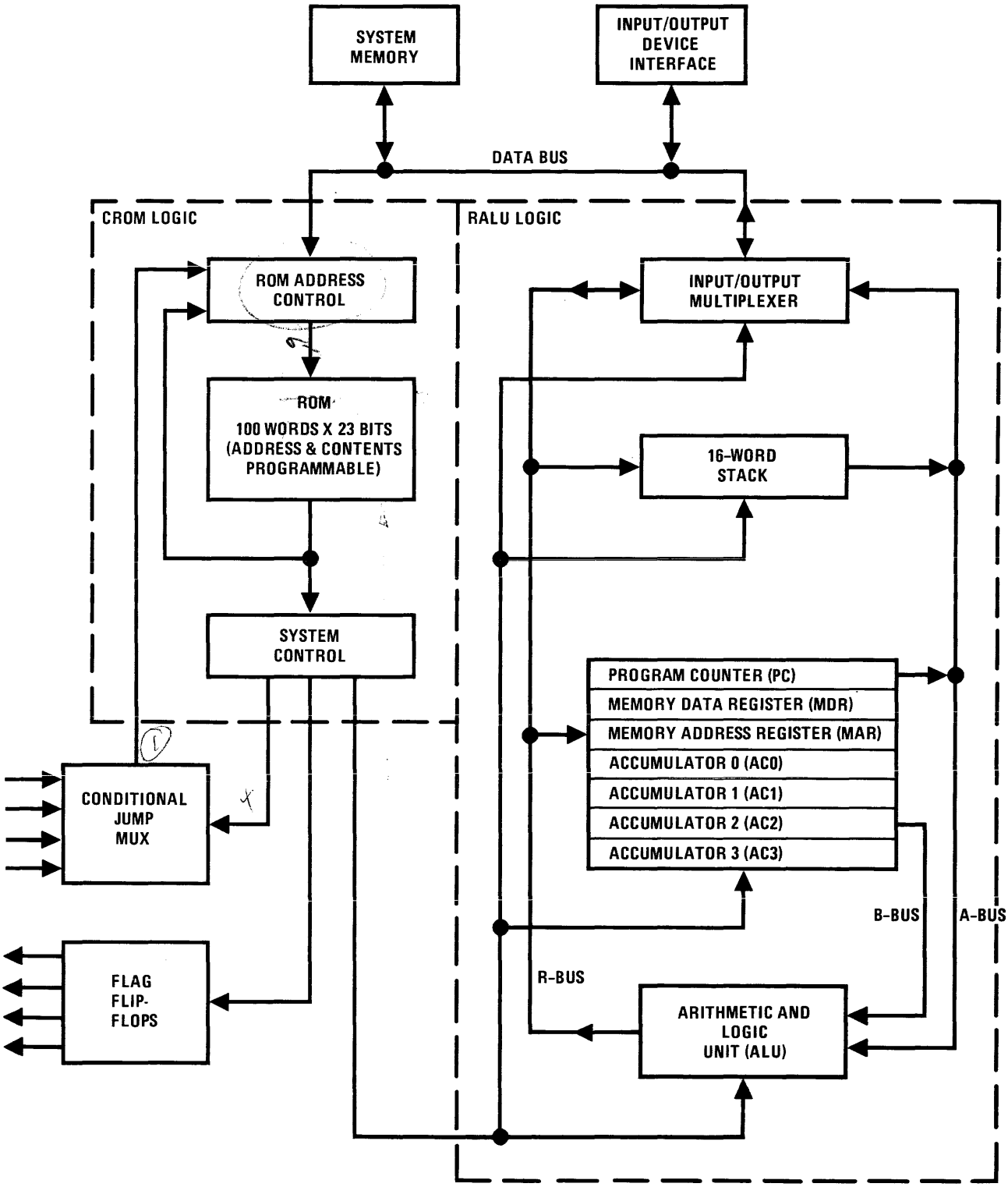


Figure 2-1. Functional Block Diagram of "Example" System

The System Memory shown in figure 2-1 may be either a (read/write) Random Access Memory (RAM), a Read Only Memory (ROM), an Electrically Programmable Read Only Memory (EPROM), or perhaps a combination of these (i.e., the control program might be stored in nonvolatile ROM.) In some applications, the same System Memory may be shared by a number of processors. An example of this sort of application is a case where one processor is used as the controller for a peripheral device that inputs data, and the other processor is used as a conventional central processor. The System Memory is composed of standard product devices, structured to meet the application needs of the end-user. Maximum memory size (address limitation) is 65,536 words.

The control function of the processor is provided by a microprogram contained in a ROM. Associated with the ROM are circuits that provide ROM address control (that is, branching within the microprogram) and distribution of the control signals to the system. Conditional branching is accomplished by selecting 1 of the 16 possible inputs to the Conditional Jump Multiplexer (CJ MUX) and gating it to the ROM Address Control Logic. Inputs to the Multiplexer consist of (1) signals generated by the processor (e.g., R = 0, sign of R) and (2) external signals (e.g., Interrupt Request). A group of 16 Flag Flip-flops are available; these may be set or reset under control of the microprogram. Some Flags are dedicated to requirements of the processor (e.g., Read Memory Flag, Write Memory Flag, et cetera); other flags may be used as required by the system application (e.g., as status flags).

The microprogram contained in the ROM executes macroprogram instructions, which are stored in the System Memory. The microprogram contains an Instruction Fetch Routine, which reads the next macroinstruction (from the System Memory) and gates the 9 most-significant bits of the 16-bit instruction word into the ROM address control. These 9 bits define a starting location in the ROM for the microinstructions that implement the macroinstruction. The last step of the Instruction Fetch Routine causes a microprogram branch to this starting location in the ROM.

2.2 System Logic Partitioning

The physical partitioning of the logic in the GPC/P is shown in figure 2-2. Five MOS/LSI devices are shown in heavy outline. The two MOS/LSI device types are the Register and ALU (RALU) and the Control ROM (CROM). The remaining devices shown in figure 2-2 are standard bipolar circuits. The circuitry shown in figure 2-2 (except the memory and the I/O Device Interface) may be readily packaged in less than 48 square inches of area on a printed circuit board (e.g., 6-inch-by-8-inch board). A 100-pin card edge connector would be adequate to make the connections to the System Memory, I/O Device Interface, and Power Supply.

The RALU logic consists of a "4-bit slice" of the logic shown on the right-hand side of figure 2-1: I/O Multiplexer, Stack, registers and ALU blocks. Four RALU devices are needed for a 16-bit processor. Machines with other word lengths that are multiples of four bits can be constructed using the appropriate number of RALUs.

The CROM of figure 2-2 contains the logic shown on the left-hand side of figure 2-1: the ROM Address Control, ROM, and System Control. A single CROM is adequate to implement a macroinstruction set similar to that of a typical minicomputer. The CROM logic design permits system operation with multiple CROMs if the microprogram requires more words of ROM than are available in a single CROM.

2.3 System Timing and Data Flow

The basic machine cycle of the GPC/P consists of the execution of a single microprogram step. This cyclic time period comprises eight time intervals: T1, T2, ... T8. As indicated in the timing diagram of figure 2-3, clock pulses occur on the four clock lines at the respective odd-time periods T1, T3, T5, and T7.

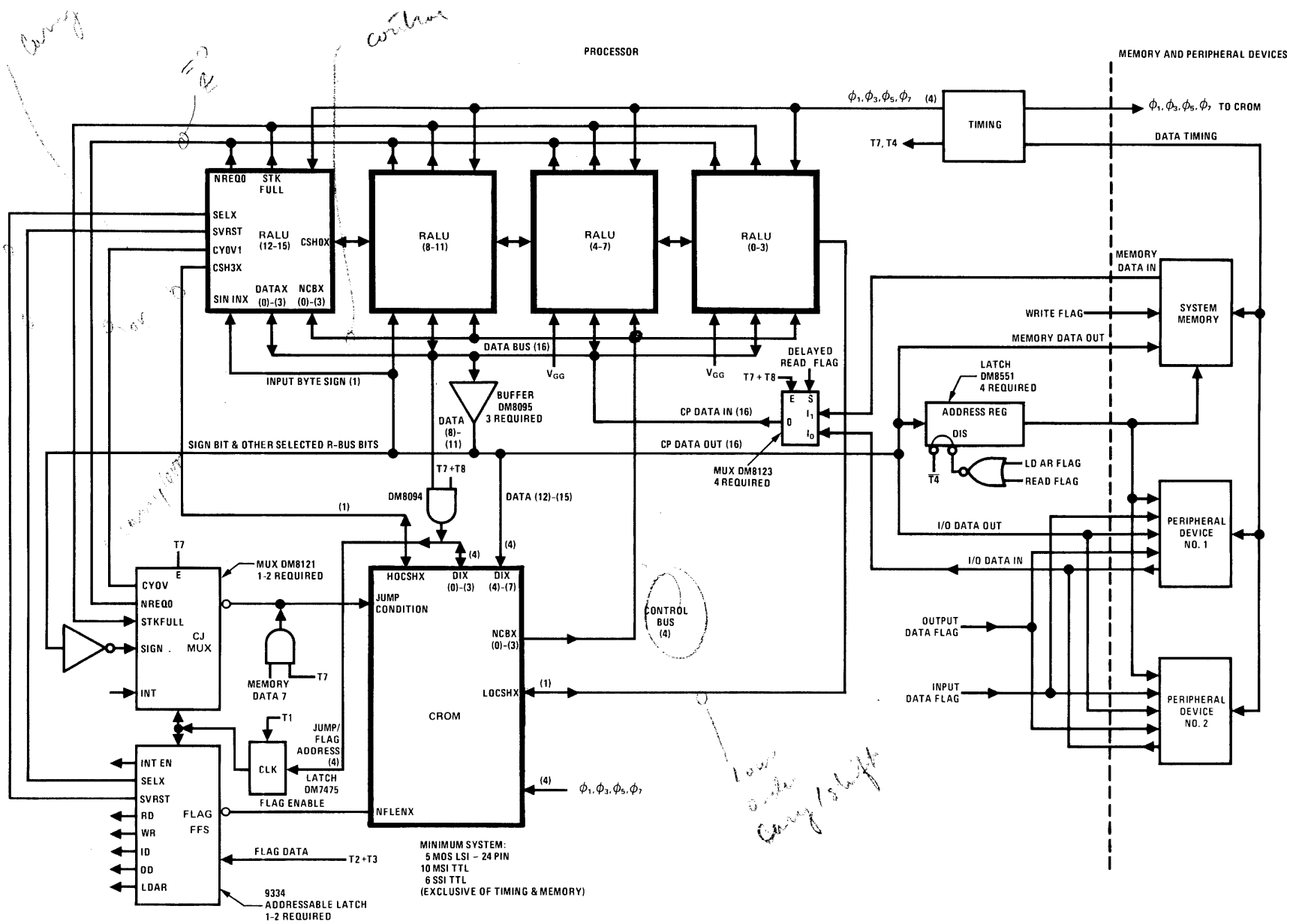


Figure 2-2. "Example" System Logic Partitioning, Simplified Schematic Diagram

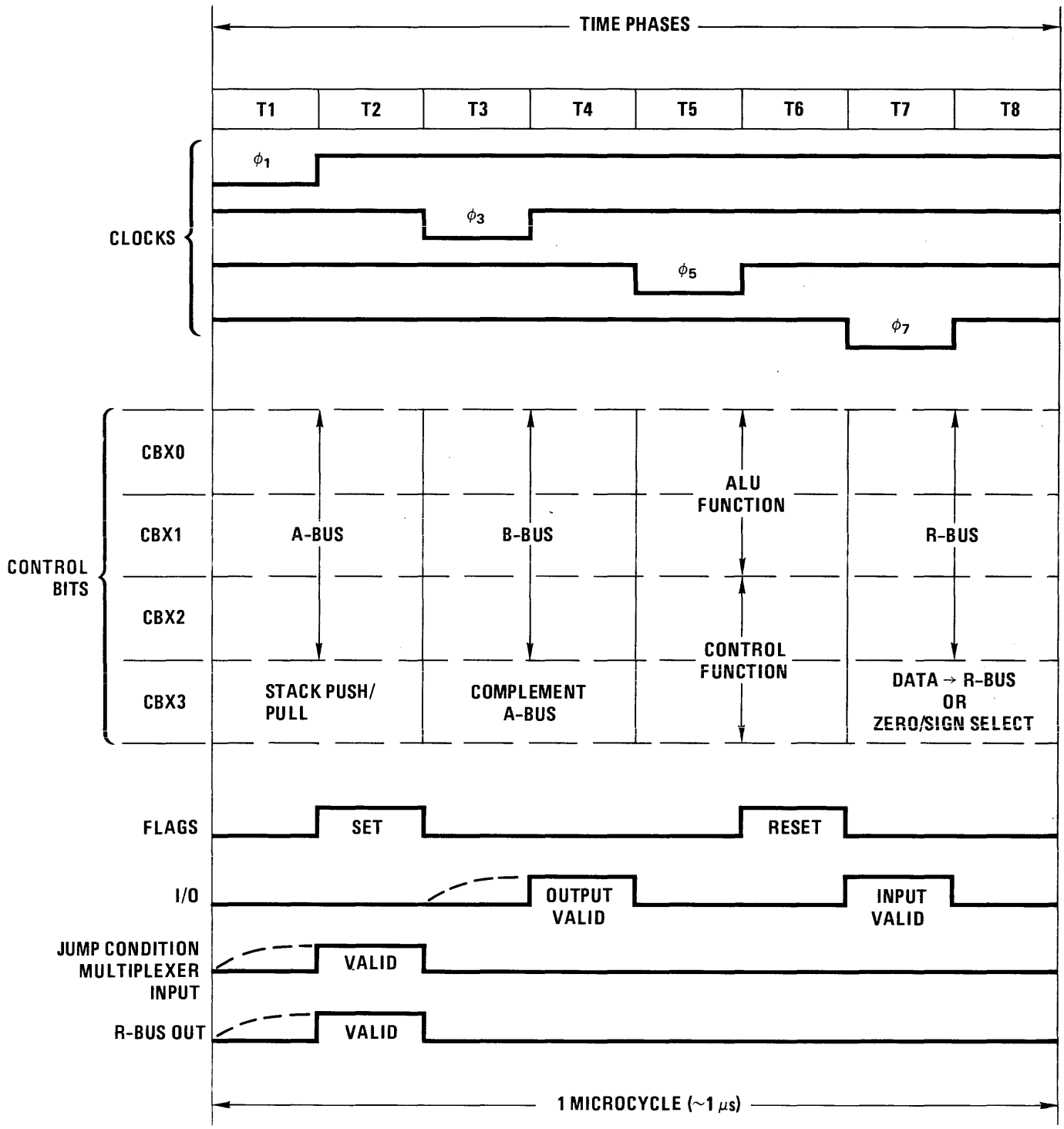


Figure 2-3. GPC/P Timing

The primary control of the RALU devices by the CROM is accomplished over the 4-line-wide Control Bus. This Control Bus is time-multiplexed to yield four 4-bit words of control information per machine cycle. The functions effected by the control bits during the four time periods are indicated on the timing diagram and are discussed further under 3.1.

The control of the Flag Flip-flops is indicated on the timing diagram. A unique flag address is established during each machine cycle; at T2, the flag may be set, and/or, at T6, the flag may be reset. It is thus possible to set, reset, or pulse a flag during a single machine cycle.

The Conditional Jump Multiplexer shares the same address lines as the Flag Flip-flops. If a conditional jump is being performed, the condition is tested during T2.

Data transfer between the RALU and the Data Bus may occur at three times during each microcycle:

- During T4, the data presently on the A-Bus is gated onto the Data Bus. The information that appears on the Data Bus at this time is either output data destined for memory or an external device, or is an address value used for loading the Address Register latch (shared by memory and external devices).

- At T2, the contents of the R-Bus (result of preceeding microcycle) are gated onto the Data Bus. Primarily, this data provides inputs to the Conditional Jump Multiplexer to permit testing of the result of the operation performed on the previous microcycle.

- During T7, data to be transferred to the RALU appears on the Data Bus. This data may then be gated onto the R-bus by the RALU's I/O Multiplexer and, subsequently, may be stored in one of the working registers or the stack.

3.0 MOS/LSI DEVICE DESCRIPTIONS

3.1 RALU Functional Description

Figure 3-1 is a simplified functional block diagram of the 4-bit RALU. The names of the 24 pins of the 4-bit RALU device package are shown around the periphery of figure 3-1. The acronyms of figure 3-1 are defined in table 3-1. The registers of the RALU may be gated onto the A-Bus or the B-Bus and, thus, may be inputs to the ALU. The output of the ALU may be shifted left or right one position and, then, transferred onto the R-Bus, from which gating into a register may occur. Data transfers between the A-Bus or the R-Bus and the external data bus may occur via the Data Multiplexer. The following text contains a more-detailed explanation of the function of the RALU.

3.1.1 Effect of Control Bits

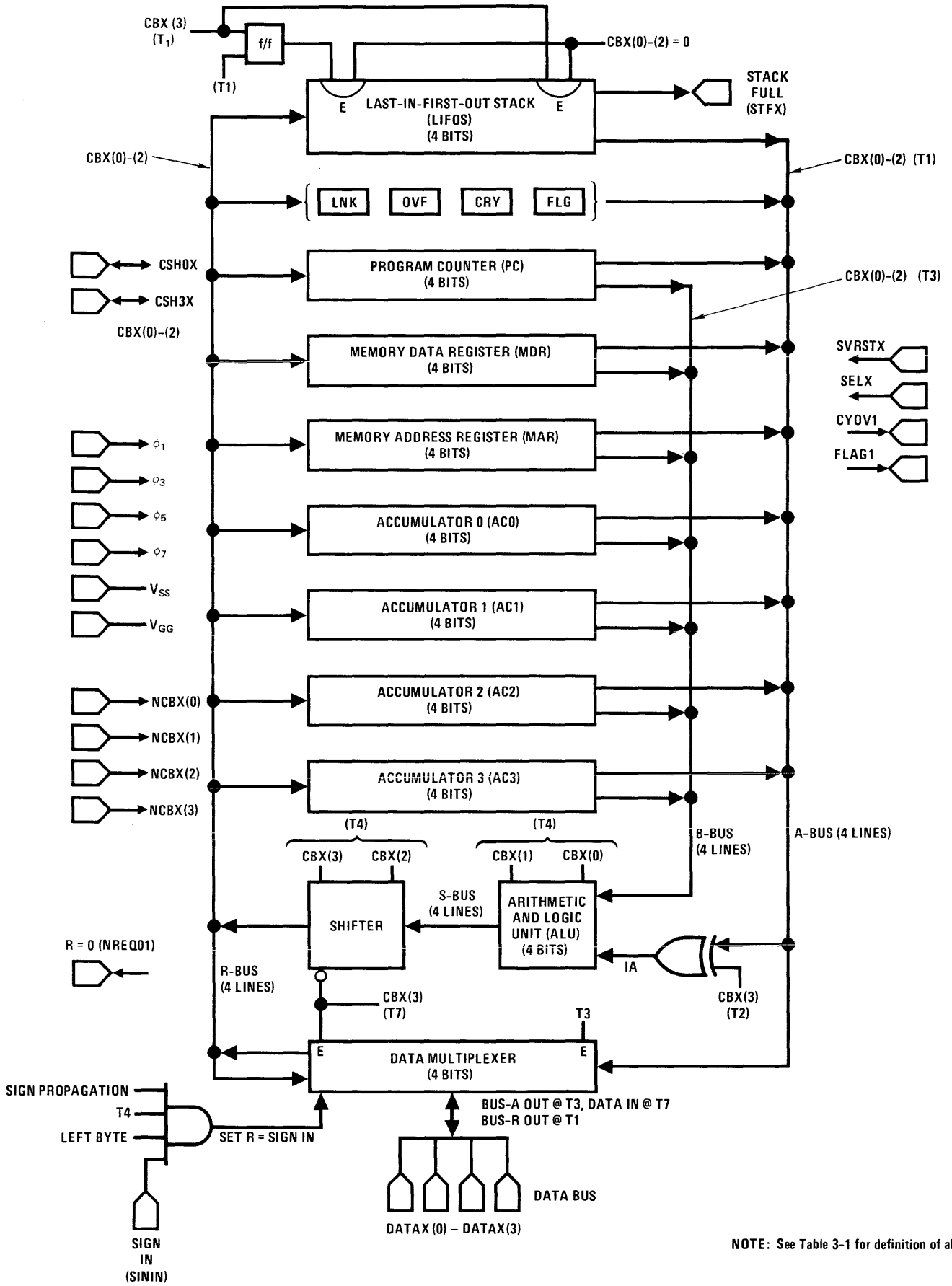
The encoding of the control bits shown on figure 2-3 for the time-multiplexed control bus is described below for each of the four time phases. It is during these time phases that control information is transferred from the CROM to the RALU.

Phase 1 Control Bits.

- If the 3-bit A-Bus address field is nonzero, the contents of the designated register are gated onto the A-Bus.

- If the A-Bus address field is zero and the push-pull control bit is "1," the LIFO Stack is pulled and the contents of the top of the stack is gated onto the A-Bus.

- If the A-Bus address field is zero and the push-pull control bit is "0," then, a value of zero is gated onto the A-Bus.



NOTE: See Table 3-1 for definition of abbreviations.

Figure 3-1. RALU Simplified Functional Block Diagram

Table 3-1. Definition of Abbreviations Used on Diagrams

Abbreviation	Definition
Ø1, Ø3, Ø5, and Ø7	Phase times 1, 3, 5, and 7. (Each of these phase times corresponds to a clock pulse.)
CBX (0), (1), (2), and (3)	Control bit signal lines 0, 1, 2, and 3
CJMUX	Conditional jump multiplexer
CSH0X, CSH3X	Carry/shift signal lines
CY0V1	Carry or overflow (flip-flop) signal
DATAX (0), (1), (2), and (3)	Data bus lines
DIX (0), (1), (2), and (3)	Data input lines
FLAG1	RALU general purpose flag signal
NFLENX	Flag enable (flip-flop) signal line (complement)
HOCSHX	High-order carry/shift signal line
INT	Interrupt request signal
INT EN	Interrupt enable (flip-flop) signal
LDAR	Load address register signal
LOCSHX	Low-order carry/shift signal line
MUX	Multiplexer
NCBX (0), (1), (2), and (3)	Control bit signal lines 0, 1, 2, and 3 (complement)
NREQ 01	Result equals zero (complement) signal
LIFOS	Last-in-first-out stack
SELX	Select (flip-flop) signal line
SININX	Sign-in (flip-flop) signal line
RD	Read memory (flip-flop) signal line
STFX	Stack full (flip-flop) signal line
SVRSTX	Save/restore (flip-flop) signal line
V_{GG}, V_{SS}	Supply voltages
WR	Write memory (flip-flop) signal line

NOTE

Pushing data onto the stack is also contingent on the push/pull control bit but is described under Phase 7 Control Bits, where the operation occurs.

Phase 3 Control Bits

- If the 3-bit B-Bus address field is nonzero, then, the contents of the register designated are gated onto the B-Bus.

- If the B-Bus address field is zero, a value of zero is gated onto the B-Bus.

- The Complement A-Bus bit causes the A input to the ALU to be complemented.

Phase 5 Control Bits

- The ALU bits designate a function according to table 3-2.

Table 3-2. ALU Function Bits

Code	Function
00	AND
01	XOR
10	OR
11	ADD

- The Control Function bits designate a function according to table 3-3. The no-op function applies only to the Control Function field. The expression R← 0/SIGN means that either zeros or the sign of the less-significant byte of the word being transferred to the R-Bus is propagated throughout the more-significant byte. (If the fourth control bit CBX(3) is "1" during phase 7, the sign is propagated; otherwise, zero is propagated.)

Table 3-3. Control-Function Bits

Code	Function
00	No Op (no operation for control bits only)
01	R← 0/SIGN (zero or sign propagation)
10	LSH (Left SHift)
11	RSH (Right SHift)

Phase 7 Control Bits

- If the 3-bit R-Bus field is nonzero, the contents of the R-Bus are gated into the register addressed by this field.

- If the Control Function bits transferred during phase 5 do not specify R← 0/SIGN (see table 3-3), then, CBX (3) specifies the source of the data gated onto the R-Bus:
  (1) If CBX(3) is "0," the source is the output of the ALU.
  (2) If CBX(3) is "1," the data comes from an external source via the Input/Output Multiplexer.

- If R← 0/SIGN is specified, then, CBX(3) specifies whether zero or the sign is propagated throughout the more-significant byte.

- If the R-Bus address is zero and the Push/Pull Control bit was active (during phase 1), data is pushed onto the LIFO Stack from the R-Bus.

The output of the ALU (S) may be shifted either left or right one place. The CSH0X and CSH3X lines are used for transferring the carry and shift data between RALUs. During T5 these lines are used for carry propogation: CSH3X has the carry-out of the most-significant bit, and CSH0X has the carry-in for the least-significant bit of the RALU.

3.1.2 STFX and NREQ0 Signals

The STFX signal indicates a "stack full" condition. When the bottom entry of the stack is filled with nonzero data, this STFX line is a "1." The STFX lines of all RALUs may be tied together and connected to the Conditional Jump Multiplexer to allow testing for the stack-full condition by the microprogram. A similar scheme is used to detect a zero-result condition with the NREQ0 signal. The NREQ0 lines are tied together for all the RALUs; the NREQ0 signal is a "0" if the R-Bus is zero as a result of the preceding machine cycle.

3.1.3 RALU Status Flags

The RALU has four status flags, which are interfaced to the A- and R-Buses. This provides a convenient means of saving status after an interrupt, and for setting the status flags. For all except the most-significant RALU, the status flags may be used for a variety of functions depending upon the application requirements. For the most-significant RALU, the status flags have the following functions:

LINK flip flop. When the SELX input to the RALU is "1," LINK is included in shift operations.

OVERFLOW flag. When enabled (under control of the CROM), this flag is set if an arithmetic overflow occurs during an add operation.

CARRY flag. When enabled (under control of the CROM), this flag is set to the value of the carry bit out of the most-significant ALU bit after an add operation.

FLAG flip flop. This flag is available for general-purpose use.

The flags may be loaded from the R-Bus or stored onto the A-Bus under control of the SVRSTX input. The output of the general-purpose Flag is available at the FLAG1 output pin; Carry and Overflow are available at CYOV1. The SELX input is used to select the Carry or Overflow for output on CYOV1 and to determine whether the Link is included in shift operations.

3.1.4 Shift Operations

During T7 and T8, CSH3X and CSH0X are used to transfer shift data: for a left shift, the most-significant bit is shifted out over CSH3X and the least-significant bit is shifted in by CSH0X; the converse is true for a right shift.

3.1.5 Byte Operations

During a macroinstruction fetch, the least-significant (LS) 8 bits of the macroinstruction are loaded into the MDR, but the most-significant (MS) 8 MDR bits are set to the value of the MS bit of the LS 8 bits; this value is the sign of the LS 8-bit byte. This permits these 8 bits to be used as a signed displacement in memory-address-formation arithmetic and for immediate instructions. The SININ signal accomplishes this function. The SININ pin is permanently connected to a logic 1 for the two low-order RALUs. The SININ pins of the two high-order RALUs are connected to the MS bit of the DATAX line of the second low-order RALU (the sign bit of the low-order 8-bit byte). During T5 and T6, the DATAX lines of all RALUs are driven to a logic 1. This permits the two high-order RALUs to

"know" that they are part of the 8 MS bits. Then, if so directed by the control bits, the R-Bus for the MS RALUs is set to the value that appears on their SININ line during T2 (i.e., the "sign" of the 8 LS bits).

3.2 CROM Functional Description

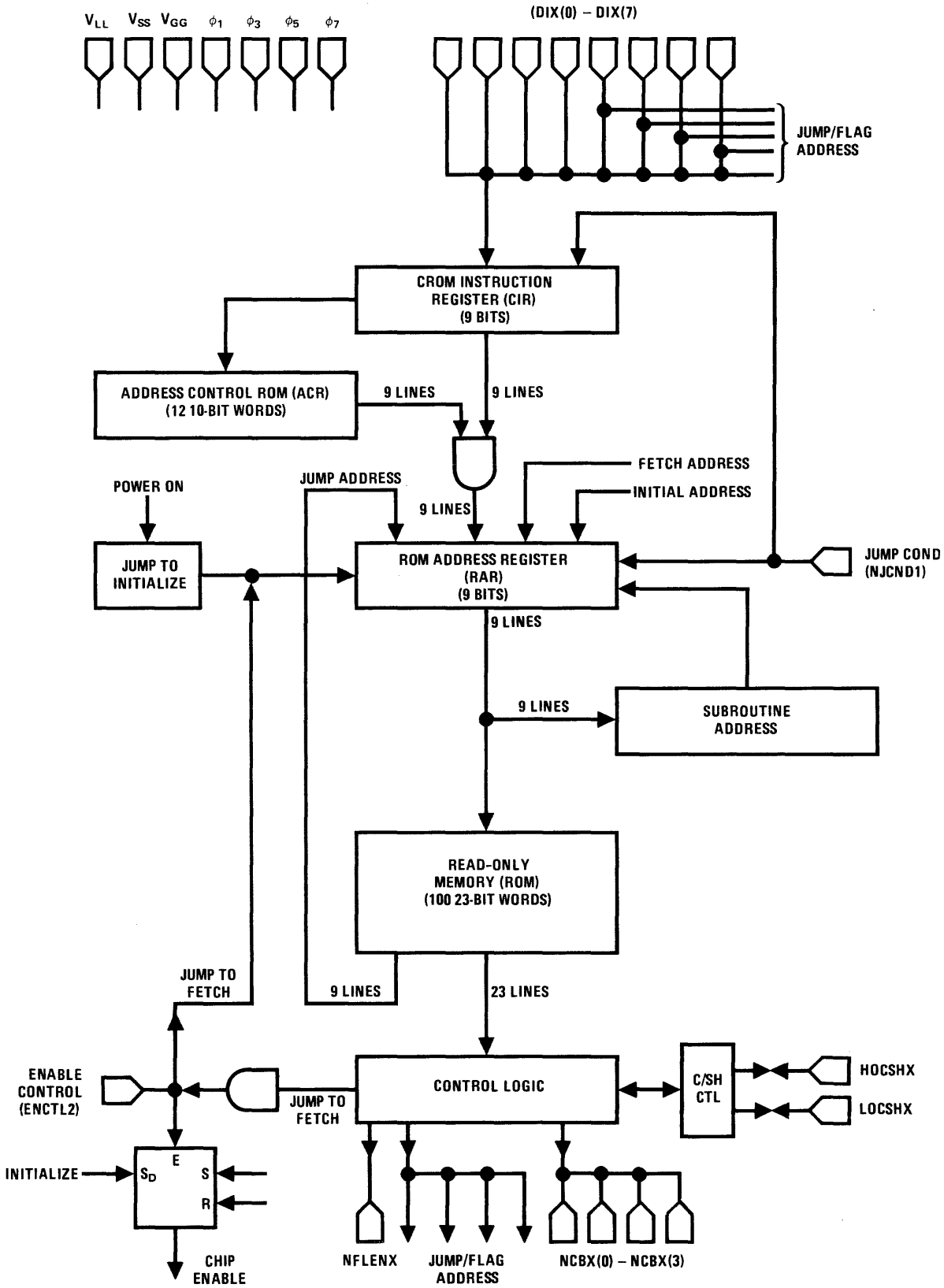
Figure 3-2 is a simplified functional block diagram of the CROM. The contents of the ROM Address Register (RAR) specify the word that is accessed in the ROM. The output of the ROM is encoded by the Control Logic to generate control signals for the CROM and the Control Bus that governs the RALUs. The ROM Address Register is loaded from the CROM Instruction Register, which in turn was loaded from memory over the data bus. A more-detailed explanation of the workings of the CROM is contained in the description that follows.

3.2.1 CROM Instruction Register (CIR)

When a macroinstruction is fetched from the system memory, the 9 most-significant bits of the macroinstruction are loaded into the CROM Instruction Register (CIR) over the eight DIX lines and the JCOND line during T7. At the conclusion of the instruction fetch routine, selected bits of the CIR are transferred to the ROM Address Register (RAR). The Address Control ROM is programmed to generate a mask that determines the CIR bits that are transferred to the RAR. This "masking" capability is used to allow some fields of the CIR to be used for purposes other than specifying the starting location in ROM for the microinstructions for a given macroinstruction. For example, bits 0 and 1 of the CIR may specify the destination (D) accumulator or Index Register (XR) for a macroinstruction. Similarly, bits 2 and 3 may specify the source (S) accumulator. For some macroinstructions, bit 8 of the CIR is masked since it is part of the displacement field of the macroinstruction; for other instructions, this bit is used to augment the remaining instruction field and, thus, is transferred to the RAR. See figure 3-3.

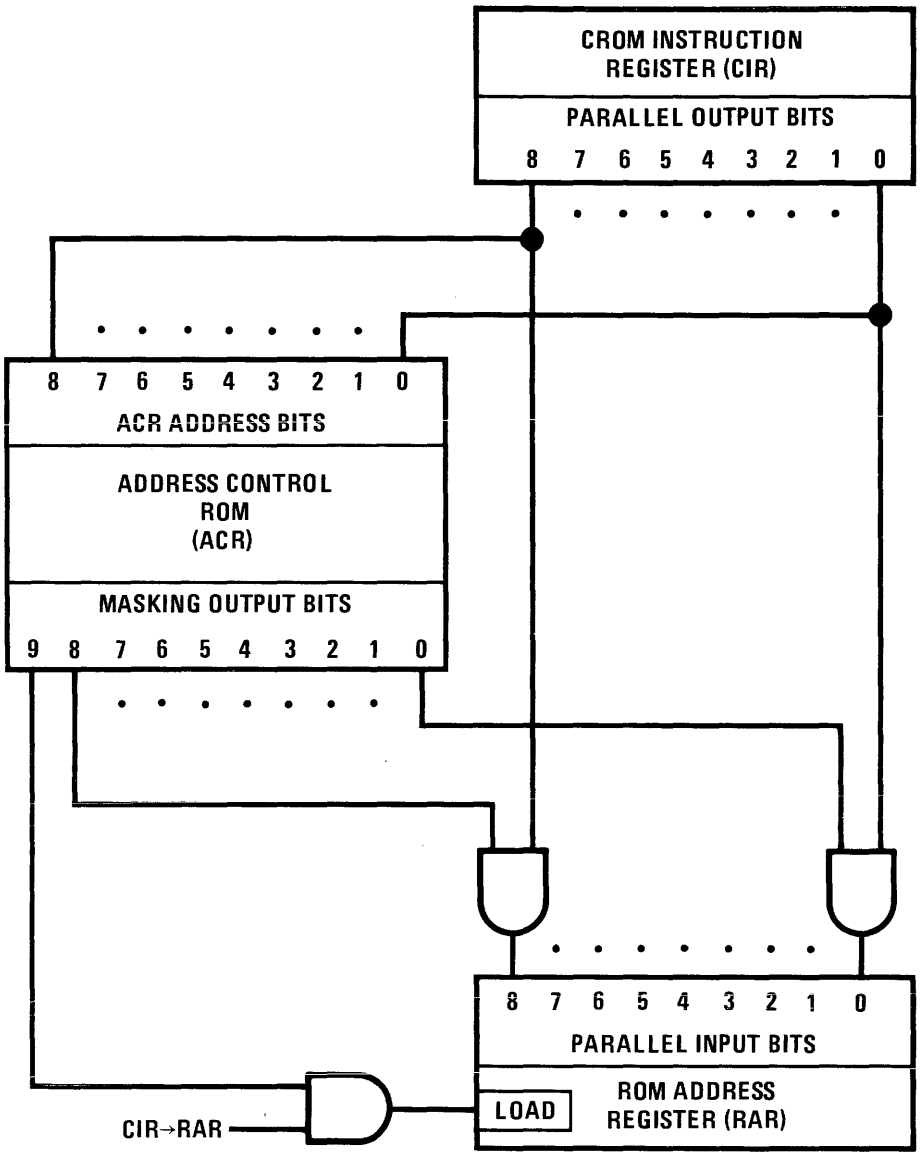
3.2.2 Address Control ROM

The Address Control ROM (ACR) consists of twelve programmable 10-bit words, each word having 9 programmable address bits (programmable as 1, 0, or don't care). The

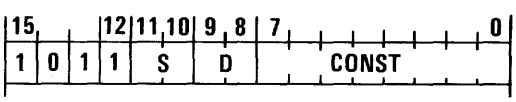


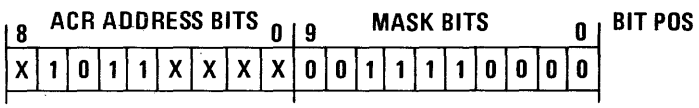
NOTE: See Table 3-1 for definition of abbreviations.

Figure 3-2. CROM Simplified Functional Block Diagram



EXAMPLE:





IF CIR CONTAINS 110110110 THEN CIR→RAR WOULD CAUSE THE RAR TO BE SET TO 010110000.

*X = DON'T CARE TERM

Figure 3-3. Loading ROM Address Register (RAR)

address lines are driven by the CIR. Output bits 0 through 8 mask the CIR bits 0 through 8, respectively when loading the RAR. Bit 9 of the ACR enables or inhibits the command to load the CIR into the RAR , as appropriate. The ACR sets instruction bits that are not part of the op code to zero when branching to an instruction address. Thus, for the example at the bottom of Figure 3-3, the S, D and CONST fields are not part of the op code and so are set to zero when the CIR loads the RAR.

3.2.3 ROM Address Register (RAR)

The contents of the RAR may be established by a number of ways in addition to transfers from the CIR. The contents of the RAR are set to a value determined by the output of the ROM as a result of a microprogram jump. The RAR is forced to specified values as the result of a "jump to initialize" command (caused by power turn on) or a "jump to fetch" (i.e., macroinstruction fetch) command. The RAR may also be modified by a return from a microprogram subroutine when the RAR is loaded with the contents of the Subroutine Address Register (SRA). In the event that the contents of the RAR are not modified by any of the means indicated above, it functions as a binary counter, and is incremented once each microcycle.

3.2.4 Read Only Memory (ROM)

All operations of the General Purpose Controller/Processor (GPC/P) are controlled by the microprogram stored in read only memory (ROM). The ROM has 100 words of 23 bits each. A new ROM word is executed each microcycle; each microprogram word specifies data flow paths and operations to the Register and Arithmetic and Logic Unit (RALU).

The function of each bit in the ROM word is shown in abbreviated form in Table 3-4. Several classes of microinstruction may be derived from the listed functions, and these are described in the Microcoding Manual. The descriptions given hereinafter serve only as an aid to understanding the function of the microprogram in the ROM.

Table 3-4. ROM Bit Assignments

FIELDS		COLUMN A	COLUMN B	COLUMNS ACTIVE
S = CIR 3,2	0	CIR →RAR (1)	JSR/RET (2)	A or B only
D = CIR 1,0	1	ENABLE Col B FNS		
XR = CIR 1,0	2	ALT A FNS	SHIFT EN	A or A & B (11)
	3	ALT B FNS	JA8	
ALU (1,0)	4	B2/B ← S (3)	JA7	
00 AND	5	B1/R ← S (4)	JA6	
01 XOR	6	B0/P-P STKEN	JA5	
10 OR	7	A2/R ← D (4)	JA4	
11 ADD	8	A1/A ← D (3)	JA3	
	9	A0/A ← XR (5)	JA2	
CTL (3,2)	10	R2	JA1	
00 NO-OP	11	R1	JA0	
01 R ← SIGN/0	12	R0	JUMP UC	
10 LSH	13	ALU 1, FA3, JC3		
11 RSH	14	ALU 0, FA2, JC2		
	15	COMPA, FA1, JC1		
	16	CIN = 1, FA0, JC0	(I/0)	B only if SHIFT EN = 1, else A only
CBX BITS	17	SFL	SHIFT L/R	
T 3 2 1 0	18	RSFL	SHIFT O/C	
1 PP A	19	J to FETCH (6)		
3 CA B	20	DATAX →R BUS	EN JUMP	A or B only
5 CTL ALU	21	BLANK L.B./SIGN PR/ENCOV (7)	READI (8)	
7 ED R	22	ROM EN (9)	GATE FA, JC (10)	A or A & B

STD'D CROM
INIT ADDR:
110001000
FETCH ADDR:
110000000

REGISTERS
000-NONE/STK
001-R1 (PC)
010-R2 (MDR)
011-R3 (MAR)
100-R4 (AC0)
101-R5 (AC1)
110-R6 (AC2)
111-R7 (AC3)

1. ROM MUST BE ENABLED AFTER CIR → RAR.
2. JUMP TO SUBROUTINE IF EN JUMP & JCOND=1, RETURN IF EN JUMP = 0.
3. HARDWARE SETS MSB=1.
4. "OR'ED" WITH R1, R0, HARDWARE SETS R2 = 1.
5. HARDWARE SETS MSB = CIR (1).
6. MULTIPLE CROM'S MAY NOT DO ANY OTHER TYPE OF JUMP TO FETCH.
7. BLB=$\overline{ADD} \cdot \overline{T} \cdot \overline{20} \cdot 21$, ENCOV= ADD $\cdot \overline{T} \cdot \overline{20} \cdot 21$, SIGN PR = $\overline{T} \cdot 20 \cdot 21$.
8. 1st μCYCLE: DATA → CIR & MDR, SIGN PROP (MUST COUNT FOR 2nd & 3rd μCYCLE TO OCCUR).
   2nd μCYCLE: INHIBIT CIR →RAR IF ACR(9) = 1.
   3rd μCYCLE: BLANK LEFT BYTE IF ACR (9) = 1 & XR = 00.
9. GATE FA, JC ALSO CAUSES ROM EN.
10. CIR (0) - (3) "OR'ED" WITH BITS 13-16.
11. IF EN JUMP = 1 THE R BUS AND P-P STKEN CBX BITS ARE SET TO ZERO INDEPENDENT OF ROM BITS 2-12.

Many of the ROM bits have multiple functions. Consider first the case where bit 1 is a "0." Then, only the functions of column A of table 3-4 are active. Bit 0 causes the transfer of the CIR contents (which are masked by the Address Control ROM) into the RAR. If bits 2 and 3 are both "0," then bits 7, 8, and 9 and bits 4, 5, and 6 specify the A-Bus and B-Bus addresses that are transferred over the Control Bus. If bit 2 or 3 is a "1," then the Alternate A or Alternate B functions are active. As indicated in table 3-4, this permits the A-, B-, or R-Bus address to be established from the S or D fields of the CIR, thus allowing the macroinstruction to specify one of the four general registers (AC0, AC1, AC2, or AC3) that takes part in an operation. Bits 13 through 16 specify the ALU function, whether the A-Bus is to be complemented, and the carry-in for the least-significant RALU. These bits also specify the address of a Flag Flip-flop that is to be set or reset (as determined by bits 17 and 18), and the select address for the Conditional Jump Multiplexer. Note that when a Flag or conditional jump operation is being performed and the contents of the registers are not to be disturbed, an R-Bus address of zero should be specified and the Push-Pull Stack bit not activated in order to inhibit loading of the R-Bus. Bit 19 causes a jump to the Instruction Fetch Routine location in the ROM. Bit 2 being set causes the R-Bus to be gated from the Data Bus rather than from the ALU, thus permitting data to be entered into the RALU from memory or an external device.

When bit 1 is set, the functions shown in column B are active. As indicated in table 3-4, this may affect whether or not the functions of column A are active. With bits 2, 17, and 18, it is possible to shift left or right (one position) either open (zero fills the end bit) or circular. Bits 3 through 11 are used to specify a ROM address for jumps if Enable Jump (bit 20) is active. Bit 12 can cause an unconditional jump; otherwise, the jump is conditional upon the response of the Conditional Jump Multiplexer. If bit 0 and bit 20 are set, then the current contents of the RAR are saved in the SRA register (i.e., a microprogram subroutine jump occurs). If bit 0 is set but bit 20 is not, then the contents

of the SRA register are stored into the RAR (i.e., a return from the subroutine is effected). Bit 21 (READI) performs multiple functions in order for the Instruction Fetch routine to be as efficient as possible. Bit 22 is used to permit the macroinstruction to specify the Flag Address or CJ MUX address (i.e., bits 0 through 3 of the CIR are OR'ed with bits 13 through 16 of the ROM output).

3.2.5 External CROM Control Signals

The HOCSHX and LOCSHX lines (High Order Carry/Shift and Low Order Carry/Shift) are connected to the CSH3X line of the most-significant RALU and the CSH0X line of the least-significant RALU, respectively. During T7 and T8, these lines are tied together within the CROM if a circular shift is being performed. LOCSHX may be controlled by the ROM output during T4 and T5 to generate carry-in for the least significant RALU.

4.0 GENERAL PURPOSE MACROINSTRUCTION SET

4.1 Introduction

Much of the control logic of the GPC/P is implemented with a microprogram stored in ROM; a wide variety of macroinstruction sets may be designed to meet custom requirements for a relatively low development cost. However, many user's requirements may be satisfied by the general purpose IMP-16 Macroinstruction Set that is described in this section. This saves instruction set development and provides off-the-shelf delivery of standard product MOS/LSI devices. An assembler program is available for this instruction set. This assembler is written in ASA FORTRAN to allow the use of an in-house general purpose computer or time-sharing system for software development. The IMP-16 Assembler Manual should be consulted for a more-detailed description of the assembler program and the macroinstruction set.

Seven classes of instructions comprise the IMP-16 macroinstruction set, totaling 43 instructions. The instruction set has unused op codes that allow expansion of the basic set to meet custom requirements.

4.2 Notation

Table 4-1 defines the notation used in the instruction descriptions.

Table 4-1. Notation Used in Descriptions of IMP-16 Instructions

ACi	denotes a specific working register (AC1, AC2, AC3, or AC4) where i is the number of the register specified as part of the function code in the instruction word.
R	denotes a working register that is specified in the instruction word field designated R. The working register is limited to one of four: AC0, AC1, AC2, or AC3.

Table 4-1. (cont.)

SR	denotes a source working register that is specified in the instruction-word field designated SR. The working register is limited to one of four: AC0, AC1, AC2, or AC3.
DR	denotes a destination working register that is specified in the instruction-word field designated DR. The working register is limited to one of four: AC1, AC2, AC3, or AC4.
PC	denotes the program counter, which contains the location of the next instruction.
EA	denotes the effective address specified by the instruction directly, indirectly, or by indexing. The contents of the effective address are used during implementation of the instruction.
STK	denotes the register at the top of the stack.
IOREG	denotes an input/output register in a peripheral device.
CC	denotes the external condition code to be test for a branch-on-condition instruction.
FC	denotes the flag code
FN	denotes the function code of instruction. The function (operation) that is to be performed by the instruction and is expressed as a binary number.
D	stands for displacement value and is an 8-bit signed twos-complemented number. This number represents either an operand or an address field in an instruction word.
XR	when not zero, designates the index register to be used in index addressing of memory.

Table 4-1. (cont.)

CV	indicates that the overflow flag is set if there is an overflow due to the instruction (either an addition or a subtraction).
CY	indicates that the carry flag is set if there is a carry due to the instruction (either an addition or a subtraction).
CTL	denotes the 7-bit control field for flag, input/output, and miscellaneous instructions.
( )	denotes the contents of the item within the parentheses (R) is read as "the contents of R."
((EA))	denotes the effective address obtained by indirection.
~	indicates the logical complement of the value on the right-hand side of ~ .
←	means "is replaced by."
@	appearing after a mnemonic of an instruction, denotes indirect addressing.
XOR	denotes the exclusive OR operation.

4.3 Memory Addressing

Memory addressing is specified by indexing (or base relative addressing). In this mode of addressing, the instruction specifies an 8-bit displacement value, and the register contains a 16-bit index value. The effective address of the instruction is formed by

adding the displacement and the contents of the specified register. Four base or index registers may be specified: ZERO, PC, AC2, AC3. The ZERO register always has a value of zero; the displacement value is treated as an unsigned value, thereby allowing the first 256 words of memory to be referenced directly. The remaining three registers treat the displacement as a twos-complement number; these blocks begin 128 words before and extend 127 words beyond the address contained in the specified register.

4.4 Instruction Set Synopsis

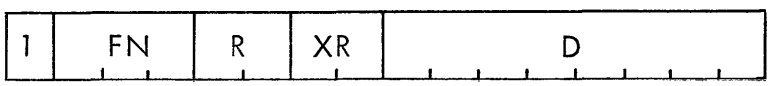
The instruction set is outlined below. Each instruction's description contains the following:

The instruction's op code

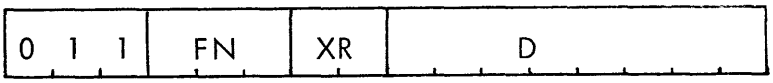
The instruction's mnemonic code

A description of the instruction's operation

4.4.1 Register/Memory (Indexed) Instructions

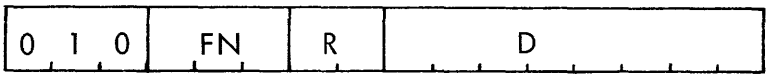


Mnemonic	FN	Description
LD	000	(R) ← (EA); LoaD
LD@	001	(R) ← ((EA)); LoaD Indirect
ST	010	(EA) ← (R); STore
ST@	011	((EA)) ← (R); STore Indirect
ADD	100	(R) ← (R) + (EA), OV, CY;ADD
SUB	101	(R) ← (R) + ~ (EA) + 1, OV, CY; SUBtract
SKG	110	IF (R) > (EA), (PC) ← (PC) + 1; SKip if Greater
SKNE	111	IF (R) ≠ (EA), (PC) ← (PC) + 1; SKip if Not Equal



Mnemonic	FN	Description
ANDi*	00i	(ACi) ← (ACi) AND (EA); AND
ORi*	01i	(ACi) ← (ACi) OR (EA); inclusive OR
SKAZi*	10i	IF [(ACi) AND (EA)] = 0, (PC) ← (PC) + 1; SKip if AND is Zero
ISZ	110	(EA) ← (EA) + 1; IF (EA) = 0, (PC) ← (PC) + 1; Increment and Skip if Zero
DSZ	111	(EA) ← (EA) - 1; IF (EA) = 0, (PC) ← (PC) + 1; Decrement and Skip if Zero

4.4.2 Single Register Instructions



Mnemonic	FN	Description
PUSH	000	(STK) ← (R); PUSH onto stack
PULL	001	(R) ← (STK); PULL from stack
AISZ**	010	(R) ← (R) + D; IF (R) = 0, (PC) ← (PC) + 1, OV, CY; Add Immediate, Skip if Zero
LI**	011	(R) ← D; Load Immediate
CAI**	100	(R) ← ~ (R) + D; Complement and Add Immediate
XCHRS	101	(STK) ← (R), (R) ← (STK); eXCHange Register and Stack
ROL***	110	ROtate (R) D places to the Left. (For D ≥ 0)
ROR***	110	ROtate (R) D places to Right. (For D < 0)
SHL***	111	SHift (R) D places to the Left. Zeros fill the vacated bit positions. (For D ≥ 0)
SHR***	111	SHift (R) -D places to the Right. Zeros fill the bit positions. (For D < 0)

* These operations are used with AC0 or AC1 (i = 0, 1).

** D is a twos-complement signed number; i.e., the high-order bit D is extended left eight positions to provide a 16-bit value.

*** The link bit will be included in the shift if the SELX flag is set. Selecting the link makes the register appear as if it were 17 bits (with the link in the most-significant bit position).

4.4.3 Register/Register Instructions

FN - designated by cross-hatching

Mnemonic	FN	Description
RADD	000	(DR) ← (SR) + (DR), OV, CY; Register ADD
RXCH	100	(DR) ← (SR), (SR) ← (DR); Register eXCHange
RCPY	101	(DR) ← (SR); Register CoPY
RXOR	110	(DR) ← (SR) XOR (DR); Register eXclusive OR
RAND	111	(DR) ← (SR) AND (DR); Register AND

4.4.4 Jump Instructions

Mnemonic	FN	Description
JMP	00	(PC) ← EA; JuMP
JMP@	01	(PC) ← (EA) ; JuMP indirect
JSR	10	(STK) ← (PC) ← EA; Jump to SubRoutine, PUSH (PC) onto stack
JSR@	11	(STK) ← (PC), (PC) ← (EA) ; Jump indirect to SubRoutine, PUSH (PC) onto stack

4.4.5 Flag Instructions

Mnemonic	FN	Description
SFLG	0	Set FLaG FC = Flag Code, Addr Reg ← CTL
PFLG	1	Pulse FLaG FC = Flag Code, Addr Reg ← CTL

The interrupt enable Flag Code is 1, and the SELX Flag Code is 2. The remaining Flag Codes may be assigned to those functions needed by the system application. The Flag Address is 8 (binary 1000) greater than FC (i.e., only Flags 8-15 are accessible with this instruction).

4.4.6 Conditional Branch Instructions

Mnemonic	CC	Description
BOC	XXXX	Branch On Condition

The four bits of CC specify an external condition that is to be tested.

If true (PC) ← (PC) + D; D is treated as a signed, twos complement number. AC0 is used for tests on the contents of a register. The following signals typically would be supplied to the CJ Mux for this instruction set. The remaining inputs to the CJ Mux may be connected as desired.

JC MUX		Condition Tested (Branch occurs if condition is true)
Address (FN)	Input Line	
0	INTRPT	Interrupt
1	REQ0	(AC0) = 0
2	NDATA (15)	(AC0) positive
3	DATA (0)	(AC0) an odd number
4	DATA (1)	Bit 1 of (AC0) a "1"
5	NREQ0	AC0 $\neq$ 0
6	CPINT	Control panel interrupt. Used by microprogram to transfer control of system to the control panel

Continued on next page.

JC MUX		Condition Tested (Branch occurs if condition is true)
Address (FN)	Input Line	
7	CONT	CONTINUE switch
8	STKFUL	Stack full
9	INEN	Interrupt Enable
10	CYOV	Carry or overflow flag: If SELX is set, overflow is tested; otherwise, carry is tested.
11	NRGT0	AC0 ≤ 0

4.4.7 Input/Output, Misc.

Mnemonic	FN	Description
HALT	0000	HALT until CONTINUE switch is pushed and released
RTI	0010	Set Interrupt Enable Flag, (PC) ← (STK) + CTL: ReTurn from Interrupt
RTS	0100	(PC) ← (STK) + CTL; ReTurn from Subroutine
----	0110	Reserved code for control panel service
RIN*	1000	(AC0) ← (IOREG), Addr Reg ← CTL + (AC3); Register IN
ROUT*	1100	(IOREG) ← (AC0), Addr Reg ← CTL + (AC3); Register OUT
PUSHF**	0001	(STK) ← RALU Flags; PUSH RALU flags onto stack
PULLF**	0101	RALU Flags ← (STK); PULL stack, store into RALU Flags

* The external device receives control signals and a 16-bit function code and device address. These 16 bits are the sum of (AC3) and CTL.

** The RALU flag bit assignments are: LINK - Bit 15, OVF (overflow) - Bit 14, CRY (Carry) - Bit 13.

4.5 Execution Time

The following table defines the execution time in terms of the number of microprogram cycles (N), the number of main memory write cycles (W), and the number of main memory read cycles (R) required. The actual execution time (in microseconds) depends upon the specific system implementation; typical values are: 1 microsecond for microprogram cycle time, 0.25 microsecond of delay per main memory read cycle, and no delay for each main memory write cycle. For this typical example, total execution time would equal 1.00N + 0.00W + 0.25R.

Table 4-2. Instruction Execution Time

Instructions	R	W	N	Comments
LD, ADD, SUB, ANDi, ORi, JMP @	2	--	5	
LD@	3	--	5	
ST	1	1	6	
SKG, SKNE, SKAZi	2	--	6,7	If skip occurs, N=7
ISZ	2	1	7,8	If skip occurs, N=8
DSZ	2	1	8,9	If skip occurs, N=9
PUSH, PULL, LI, CAI, RADD, JMP	1	--	3	
AISZ	1	--	4,5	If skip occurs N=5
XCHRS, RTI	1	--	5	
ROL, ROR, SHL, SHR	1	--	4+3K	K=number of positions shifted or rotated
RXCH	1	--	8	
RCPY, RXOR, RAND	1	--	6	
JSR @	2	--	6	
SFLG, PFLG, RTS, PUSHF, PULLF, JSR	1	--	4	
BOC	1	--	4,5	If branch occurs N=5
RIN, ROUT	1	--	7	
ST@	2	1	8	

4.6 Initialization, Interrupt Processing

The first instruction executed after power is applied is located at FFFE (hexadecimal). The choice of this address was made in order to permit initialization routines (stored in ROM) to be located in the upper part of main memory.

When the processor is interrupted, the interrupt enable flag (INEN) is cleared and the program counter (PC) is pushed onto the stack. The instruction in location 1 of main memory is then executed.

DOCUMENT REVIEW FORM

Your comments concerning this document help us produce better documentation for you.

GENERAL COMMENTS

	Yes	No		Yes	No
easy to read?	☐	☐	complete?	☐	☐
well organized?	☐	☐	well illustrated?	☐	☐
accurate?	☐	☐	suitable for your needs?	☐	☐

How do you use this document?

As an introduction to the subject ☐ For continual reference ☐

For additional knowledge ☐ Other ☐

SPECIFIC CLARIFICATIONS AND/OR CORRECTIONS

Reference	Page No.

This form should not be used as an order blank. Requests for copies of publications should be directed to the National Semiconductor sales office serving your locality.

Send comments to: National Semiconductor Corporation, 2900 Semiconductor Drive, Santa Clara, California 95051 - Attention: Systems Publications.