Data
Systems

PHILIPS

Service Manual

P856M/P857M CPU

A PUBLICATION OF
PHILIPS DATA SYSTEMS
APELDOORN, THE NETHERLANDS.

PUB. NO. 5122 991 26953

DATE October 1978

Great care has been taken to ensure that the information contained in this handbook is accurate and complete. Should any errors or omissions be discovered, however, or should any user wish to make a suggestion for improving this handbook, he is invited to send the relevant details to:

PHILIPS DATA SYSTEMS
SERV. DOC. AND TRAINING DEPT.
P.O. BOX 245, APELDOORN,
THE NETHERLANDS.

TABLE OF CONTENTS

Paragraph | Page

I GENERAL DESCRIPTION

IV MECHANICAL

V POWER SUPPLIES

Paragraph | Page

LIST OF ILLUSTRATIONS

Figure Page

Figure Page

Figure Page

LIST OF TABLES

Table Page

SECTION I

GENERAL DESCRIPTION

1.1 SYSTEM ORGANIZATION

The P856M/P857M systems consist of a basic central processor unit (CPU) and various independent elements interconnected via a General Purpose Bus. The CPU is contained on a single printed-circuit card. The same card includes a Serial Control unit for connecting an operator's device. Some of the other system elements are: memory modules, control units, input/output processor (IOP), the P857 options floating-point processor (FPP) and memory management unit (MMU).

1.2 A block diagram of the P856M/P857M systems is shown in Figure 1-1. All system elements are interconnected via the GP Bus. The P852M system elements are plug compatible with the P856M/P857M systems and may be connected via the same GP Bus. P850/P855 control units may also be used with the 856/857 system; the 850/855 cards are mounted in their own chassis and connected to the GP Bus via a bus converter unit. Chassis information and wiring is included in Section IV.

1.3 The P857M and P856M systems use the same basic logic design and both systems are based on a single-card CPU and operator's control unit. The two systems use different microprogram control, and the P857M performs an expanded number of functions, including operations with the FPP and MMU options. The specific logic differences are noted throughout the Logic description, Section II. Also, the P857M and P856M systems have different standard control panels as shown in Section III.

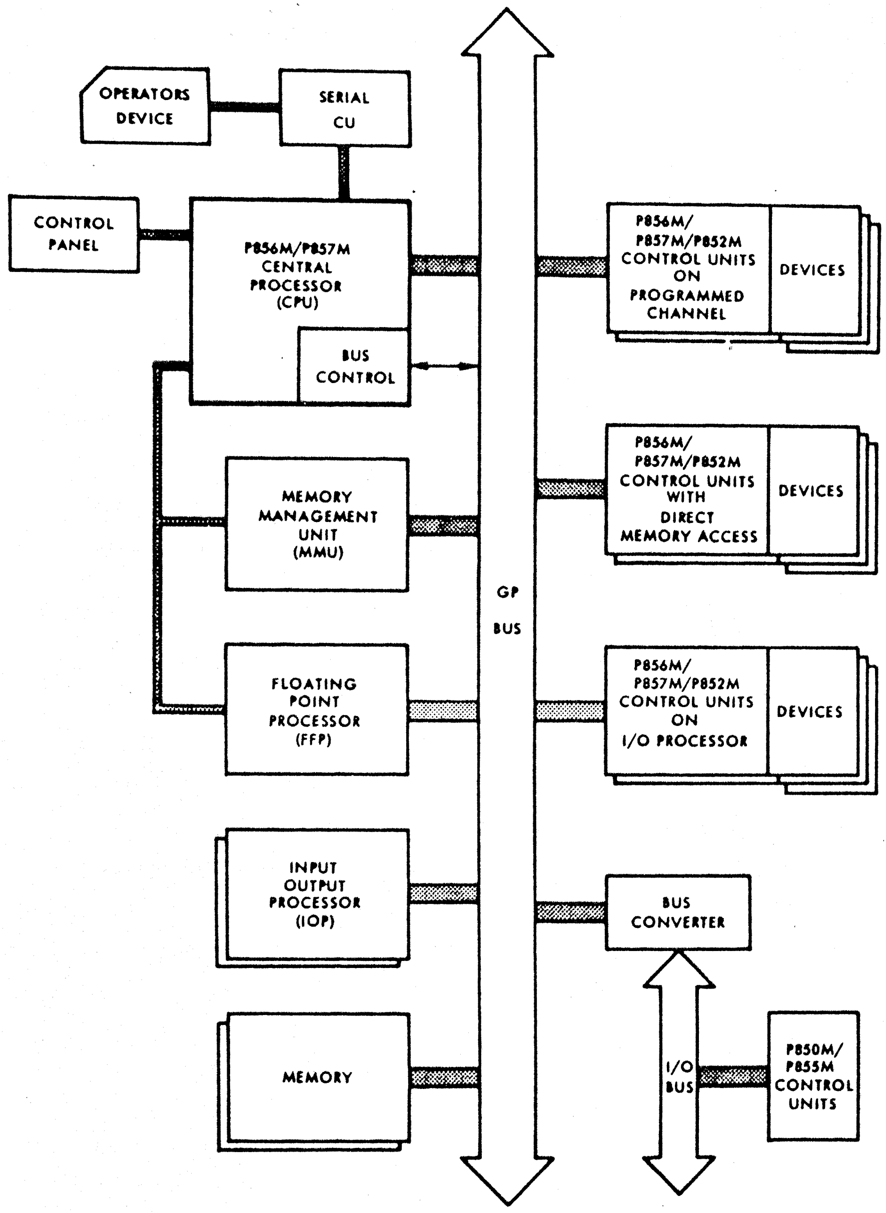

OPERATORS DEVICE
SERIAL CU
CONTROL PANEL
P856M/P857M CENTRAL PROCESSOR (CPU)
BUS CONTROL
MEMORY MANAGEMENT UNIT (MMU)
FLOATING POINT PROCESSOR (FFP)
INPUT OUTPUT PROCESSOR (IOP)
MEMORY
GP BUS
P856M/ P857M/P852M CONTROL UNITS ON PROGRAMMED CHANNEL
DEVICES
P856M/ P857M/P852M CONTROL UNITS WITH DIRECT MEMORY ACCESS
DEVICES
P856M/ P857M/P852M CONTROL UNITS ON I/O PROCESSOR
DEVICES
BUS CONVERTER
I/O BUS
P850M/ P855M CONTROL UNITS

Figure 1-1 P856M/P857M System Block Diagram

1.4 Memory

The P856M maximum memory size is 32k words. The P857M maximum memory size is 32k words with the basic system configuration, 64k words with the MMU option, or 128k words with the MMU and the M5 chassis options. Memories are provided in modular form with up to 16k words per card.

1.5 Memory addressing on the GP Bus lines is by character address, and usually shown in hexadecimal. CPU logic which deals with memory word addresses simply places the word code on the GP Bus lines shifted one bit to the left. This then accesses an even-numbered memory character address. Some of the first (low numbered) memory locations are reserved for hardware-addressed functions, as follows:

Address		Function
Decimal Word	Hexadecimal Character	
0	000	Interrupt list words
62	07C	
63	07E	Trap routine list word
64	080	
127	0FE	
128	100	

OVERFLOW OF STACK

1.6 Memory Management Unit (MMU)

The MMU is a P857 single-card hardware option which uses Virtual Addressing. This option extends the main memory from 32k to 128k words, while still using the 16-bit addressing. The Virtual Addressing system also allows software extension of main store to backing store, via the Direct Memory Access channel. The MMU card uses a dedicated slot adjacent to the CPU card. The MMU is described in Appendix C.

1.7 Floating Point Processor (FPP)

The FPP is a P857 single-card hardware option which performs floating-point arithmetic operations. The FPP uses a dedicated slot (beside the MMU position)

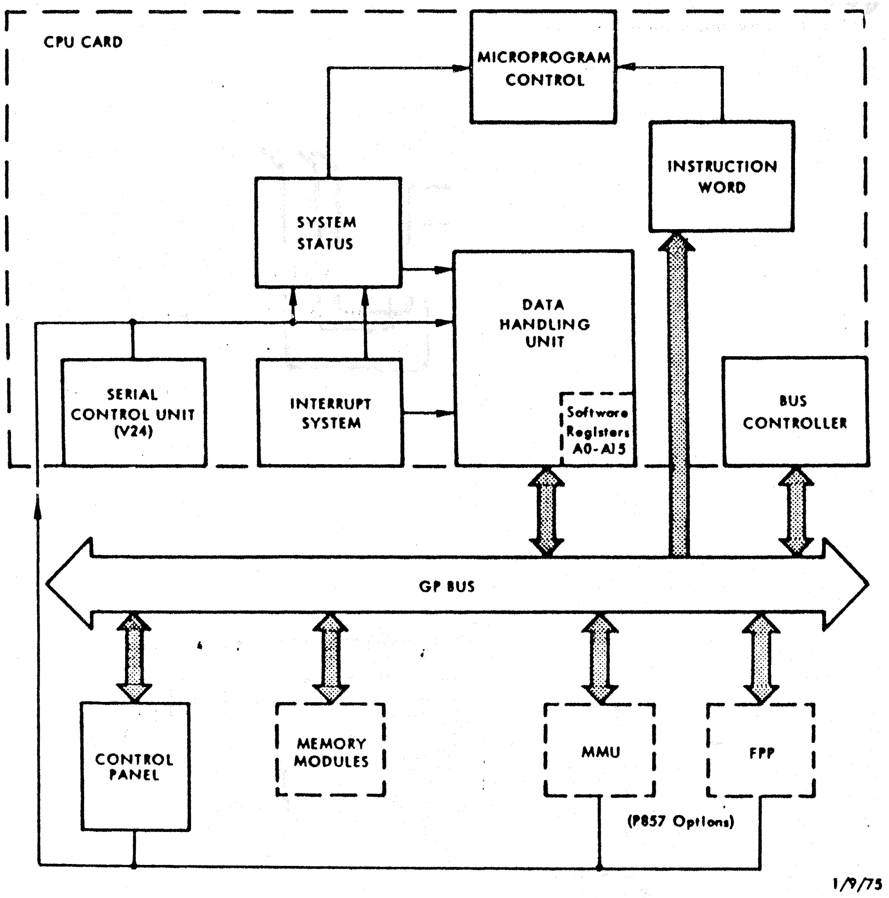


Figure 1-2 P856M/P857M CPU Block Diagram

in the cabinet. Dedicated wiring between the FPP and CPU is used to increase operating speed for the floating point calculations. The FPP is considered an extension of the CPU arithmetic section rather than an independent unit. The FPP is described in Appendix D.

1.8 I/O Processor (IOP)

The IOP is a hardware I/O channel that manages direct data transfers between control units and memory. The IOP multiplexes a number of control units for memory data transfer (eight CUs with the IOP type A). The IOP card can be inserted in any slot of the basic cabinet. The IOP is described in Appendix B. Additional information about I/O channels is given in paragraph 1.28.

1.9 GP BUS

The General Purpose Bus is a 57-line communicating link between all system elements, such as the CPU memory modules, I/O processors, and control units (Figure 1-1). System elements use the GP Bus on a master-slave basis. The CPU operates only as a master; the memory, external registers, and most device control units are slaves; the I/O processor may operate as master or slave (for CU with integrated DMA channel, the DMA can operate as master or slave).

1.10 The Bus Controller logic in the CPU regulates access of masters to the GP Bus. Whenever the Bus is free, the Bus Controller scans the masters in a specific sequence for a Bus-access request. The CPU has direct access to the memory at the completion of each instruction.

1.11 CENTRAL PROCESSOR UNIT

The P856/P857 CPU card contains the complete central processor, the GP-Bus control logic, and a serial control unit. The main CPU logic units and data paths are shown on the block diagram, Figure 1-2. A more detailed block diagram and complete logic diagrams are provided in Section II, CPU Logic description.

1.12 The Data Handling Unit does the processing of all data words accessed by the CPU. This unit also handles the addressing for both data transfers and instruction-word transfers. The Microprogram Control is a read-only memory and associated logic which controls all CPU operations.

1.13 CPU SOFTWARE REGISTERS

A scratchpad comprising sixteen 16-bit registers (A0 to A15) is directly accessible to software. The scratchpad contains 15 working registers (A0 to A14) and a stack pointer (A15). The working registers are used as an operand for some instructions. The scratchpad is located in the Data Handling section of the CPU logic and is connected to the operand-A input of the arithmetic logic unit.

1.14 A 2-bit condition register (CR) is provided for testing operation results. This register is also located in the Data Handling section of the CPU logic.

1.15 BUS CONTROLLER

The Bus Controller scans the GP Bus priority chain for selecting a master, provides control for the memory, and gates input/output data between the GP Bus and the CPU. The Bus Controller is included on the CPU card and is interconnected with the CPU logic.

1.16 INTERRUPT SYSTEM

The Interrupt System is a hardware feature which allows a running program to be interrupted by a higher-priority program. The Interrupt System is used for CPU logic functions and for control-unit I/O channel operations. There are 63 interrupt levels, divided into two groups: internal and external. There is a separate interrupt-request associated with each of the 63 levels.

1.17 Internal Interrupts

The internal interrupts (Figure 1-3A) use the eight highest-priority levels, 0-7. Four of the internal interrupts are pre-wired from CPU logic functions; the other four may be connected to other units in the basic chassis, but are not

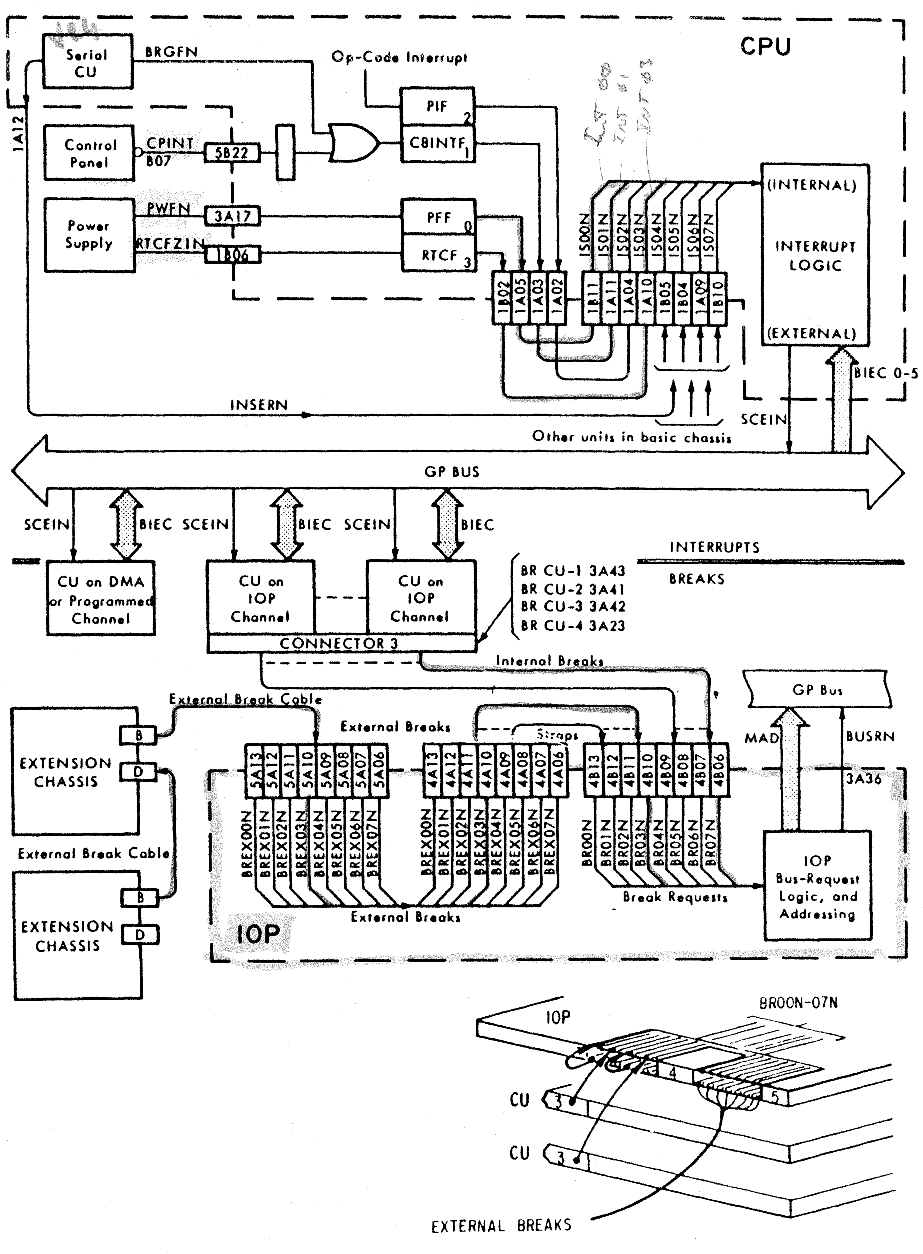
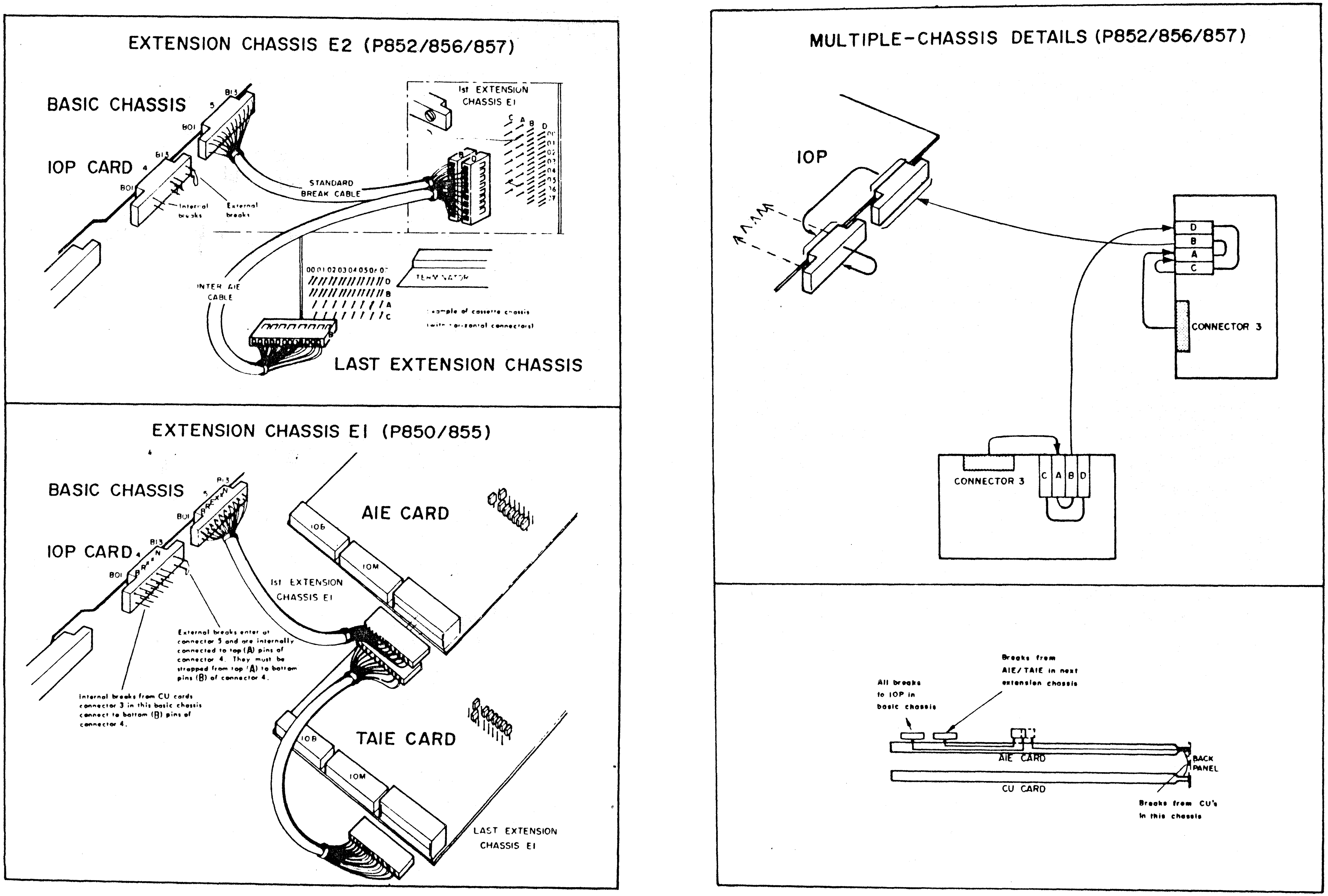


Figure 1-3A Interrupts and Breaks



Example of cassette chassis (with horizontal connectors)

External breaks enter at connector 5 and are internally connected to top (A) pins of connector 4. They must be strapped from top (A) to bottom pins (B) of connector 4.

Internal breaks from CU cards connector 3 in this basic chassis connect to bottom (B) pins of connector 4.

All breaks to IOP in basic chassis

Breaks from AIE/TAIE in next extension chassis

Breaks from CU's in this chassis

Figure 1-3B Break Cable Details

usec for CU I/O channel operations. If any of the levels four through seven are not used for internal interrupts, they may be used by the external interrupts on the BIEC lines. The internal interrupts are assigned as follows:

level 0 - Power failure, Automatic restart.
1 - Operator's interrupt (Control Panel or I/O Console).
2 - Operation Code interrupt (Link to Monitor and Stack Overflow).
3 - Real Time Clock interrupt.
4-7 - Available for other internal interrupts within the basic chassis (serial CU, FPP interrupts, etc.) or external interrupts on the BIEC lines.

The four CPU logic interrupts are set into CPU flip-flops as they occur. They are reset individually by the RIT instruction in the corresponding interrupt program.

1.18 External Interrupts

The external interrupts (Figure 1 -3) are assigned priorities 4 to 62, although 4 to 7 may be used by internal interrupts. All I/O control units (except the CPU-integral serial controller) use the external interrupts, including those control units mounted in the basic chassis. Control units on the programmed channel use an external interrupt to transfer each word. Control units on all three channels use the external interrupt to request a status transfer at the end of a data-block transfer. For control units on the IOP channel, word transfers are initiated by break requests (BR) to the IOP; the IOP then makes a Bus Request to obtain control of the GP Bus for the word transfer. A break request is part of the IOP channel (paragraph 1.33) and not part of the interrupt system.

1.19 External interrupt requests are connected to the CPU interrupt logic via a 6-bit code on the GP-Bus BIEC lines. The priority level of a control unit is established by a priority encoder (with a set of jumpers) on the CU itself. Any pending interrupts on the BIEC lines are sampled at the end of the instructions by the scan-interrupt signal SCEIN. The CU priority encoders sample the BIEC lines and only the highest priority external interrupt request is coded onto the lines.

1.20 Interrupt Control

An Enable Interrupt (ENB) instruction is used to enable the CPU interrupt system. The entire interrupt system can be blocked with the Inhibit Interrupt (INH) instruction. The hardware flip-flops generating the internal interrupts are reset individually by the RIT instruction. The hardware that is generating the external interrupts is reset by appropriate CIO instructions. At power-on time, and at every master clear from the control panel:

- the CPU is set to Enable Interrupt mode,
- the current program level is established at 63, and
- all internal and external interrupt requests are reset.

1.21 Interrupt Sequence

The internal/high-priority interrupts (levels 0-7) are sampled at the end of each instruction execution (except Move Table which is sampled early). If there is no internal interrupt, and there has been no external sampling within 2 microseconds, the highest-priority external interrupt (which is coded on the BIEC lines) is sampled. The highest-priority interrupt request is then compared with the priority level of the running program. If the running program is of higher or equal priority to the interrupt request, the program continues. If the interrupt request is of higher priority than the running program, the interrupt sequence is started:

- The current instruction (except Move Table) is completed. For Move Table, registers are updated to allow resuming the instruction at the point it was suspended.
- The program counter (P) is stored in the memory-stack location specified by stack pointer A15. P contains the address of the next instruction (except for Move Table, where P points to the instruction itself). A15 is decremented by 2.
- The program status word is stored in the memory location adjacent to (P), specified by stack pointer A15. A15 is decremented by 2.
- The Inhibit Interrupt state is set.
- The system User Mode flag is reset (unless already reset).
- The priority level register (PLR) is loaded with the new level number.

- An indirect branch is made to the corresponding memory location (paragraph 1.23).
- The interrupt routine is executed.

Note: A Return instruction with a pointer other than A15 can be used independently of the interrupt routine to switch from any program to another under a supervisory program control.

1.22 Interrupt Routine

The following operations must be performed by the interrupt-routine program:

- Some or all of the accumulators (A0 - 15) are saved in the interrupt memory stack.
- The interrupt itself is treated, including a RIT, SST, or other instruction to reset the interrupt signal.
- The accumulators are re-loaded from the stack at the end of the interrupt routine.
- A Return instruction, referring to stack pointer A15, is programmed. During this instruction, the contents of A15 are used (with incrementing) to restore the program counter into P and retrieve the program status word.

1.23 Interrupt Addresses

The first 63 word locations in memory are used for the interrupt-routine list words. The CPU interrupt logic generates a direct six-bit word address for the accepted interrupt. This address code is shifted left one position onto the address lines to produce the memory character address, as follows:

Interrupt Level	Address Code from Interrupt Logic	Bit 0 added, for mem. char. add.	
0	0 0 0 0 0 0	0	0
1	0 0 0 0 0 1	0	2
2	0 0 0 0 1 0	0	4
3	0 0 0 0 1 1	0	6
4	0 0 0 1 0 0	0	8
5	0 0 0 1 0 1	0	A
6	0 0 0 1 1 0	0	C
	etc. to		
62	1 1 1 1 1 0	0	7C

1.24 Memory Stack Operation

The interrupt system utilizes a memory stack with automatic handling. The hardware uses this system stack during the interrupt sequence to save the program status word and the instruction counter of the interrupted program. The software uses the stack: to save and later restore any other parameters of the interrupted program; to link a program to a subroutine; and to return to the main program. The system stack is also used by software for Traps and page faults.

1.25 The stack operates on a last-in first-out basis, controlled by the automatic updating of stack pointer A15. Load, Store, Multiple Load, and Multiple Store can be used as stack-handling instructions when their effective address refers to A15. The Call Function (CF) instruction is a branch with automatic saving of PSW and P into the stack. The Return (RTN) instruction is used at the end of an interrupt routine or a subroutine to restore PSW and P.

1.26 Stack Overflow is signalled by an Operation Code interrupt (level 2, internal interrupt). This signal is generated by hardware when the stack pointer decrements to less than 128_{10} (word address) to indicate that the stack is almost full and to prevent overwriting in the dedicated low address memory locations.

1.27 Additional memory stacks may be used by software, using the scratchpad accumulators A0 to A14 for stack pointers. These software stacks will not have the automatic handling and updating like the system stack which uses A15 as the pointer. All references here to the stack pertain to the automatic-handling stack. All memory stacks may have software limits to stack size established at system generation time.

1.28 INPUT/OUTPUT CHANNELS

The P852M/P856M/P857M systems have three different input/output channels:

- programmed channel,
- multiplex-type I/O-processor channel, and
- direct memory access channel.

All I/O data transfers are via the GP Bus and are timed by the Bus Controller logic on the CPU card. For all three types of I/O transfers, the program initiates control-unit operation with a CIO Start command. The CU must reply with an interrupt request or a break request (depending on the type of I/O channel) when it is ready for the first word (or character) transfer. The data are then transferred according to the channel type. When a data-block transfer is ended (either complete or early), the CU signals to the CPU with an interrupt request. The program must then issue a Send Status (SST) command to obtain the status word from the CU.

1.29 For transferring a block of data via any I/O channel, the program provides the memory address of the first word and the length of the block to be transferred. During the transfer, these two control words must be updated: the memory address is incremented to select sequential locations in the data block; the block length is decremented to determine when the complete block has been transferred (length = zero). The method of handling this pair of control words depends on the type of I/O channel.

1.30 Programmed Channel

This is an input/output exchange between a CU and memory via the CPU, under complete program control (Figure 1-4). The exchange is word-by-word or character-by character at up to 40,000 characters per second. On the programmed channel, the address/length control-word set is located in program registers. For each data word or character transferred, the program must access both of these registers to up-date the control words.

1.31 For an output transfer, a Load Register instruction loads the first word from memory into the CPU scratchpad register. The CU signals that it is ready with an Interrupt Request to the CPU. An OTR instruction then transfers the word from the CPU to the CU, while another Load instruction obtains the next word from memory. This procedure continues until the last word of the block is transferred. When the program loads the last data word, its block length is counted to zero. The CPU then sends CIO Halt to the CU along with the last OTR data transfer, and the transfer is ended with the status transfer.

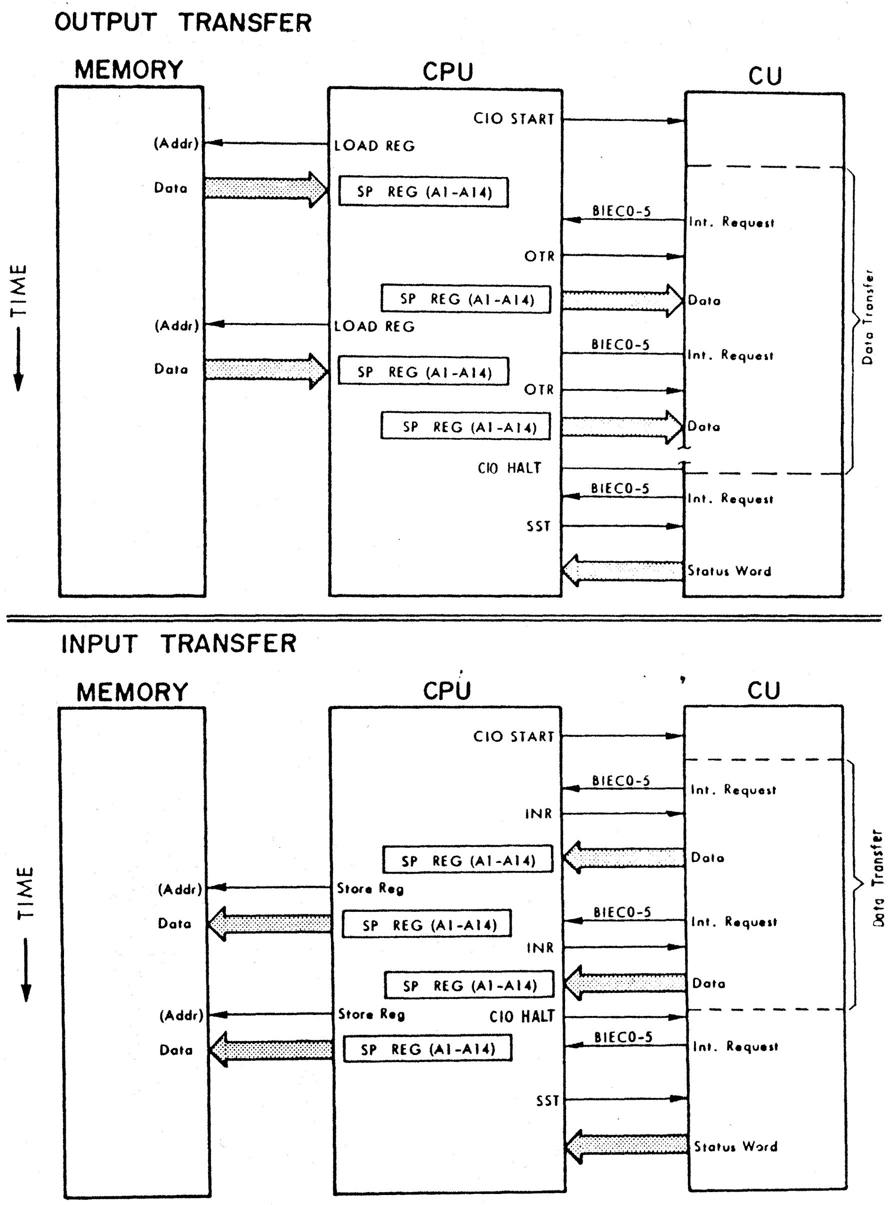


Figure 1-4 Programmed Channel Transfers

1.32 The input transfer sequence is essentially the same as the output transfer. The input sequence, however, begins with an INR command which transfers data from the CU into the CPU scratchpad register. A Store Register instruction then transfers the word from the scratchpad register into the memory.

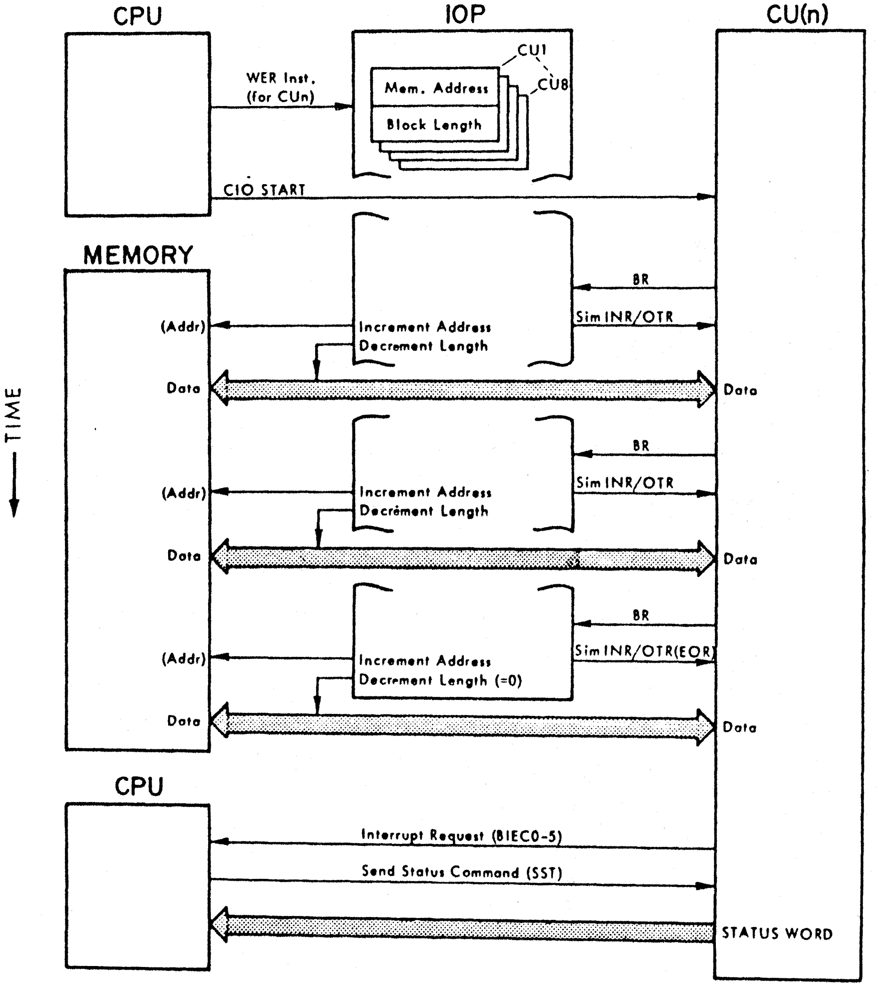
1.33 I/O Processor Channel

The Input/Output Processor (IOP) channel manages data transfers directly between memory and a number of multiplexed control units (Figure 1-5). CPU registers are not used, and there is no need for program control except for starting the exchange and testing status at the completion. The exchange is in blocks at up to one million words per second.

1.34 The IOP is a hardware option that contains a pair of address/length control-word registers for each of its CU channels. At the beginning of a transfer to one CU, the program uses two WER instructions to load this register pair with the starting address and the block length. The IOP logic then provides all GP Bus timing signals to control the data transfers directly between the memory and the CU.

1.35 For input or output transfer, the CU signals that it is ready with a Break Request (BR) to the IOP. The IOP makes a Bus Request to obtain control of the GP Bus (Figure 1-3). The IOP then sends a simulated INR or OTR command to the CU to initiate one word transfer. The INR/OTR command is simulated in that it is generated by the IOP and is not a programmed instruction. The IOP logic updates its control-word registers for each data word. When the block length is counted to zero, the IOP sends End Of Record (EOR) to the CU along with the last simulated INR/OTR data transfer. The data block transfer is ended with an SST command and status transfer between the CU and CPU.

1.36 The IOP can be loaded with the data-exchange control information for a number of control units (up to eight for the type-A IOP). The IOP then multiplexes the exchanges between the CUs and the memory.



Note -- This diagram shows the sequence for transferring a block of data between memory and a single CU, (n). The IOP can multiplex up to eight CUs; the operation shown is thus duplicated for each CU.

Figure 1-5 Multiplexed I/O Processor Channel Transfers

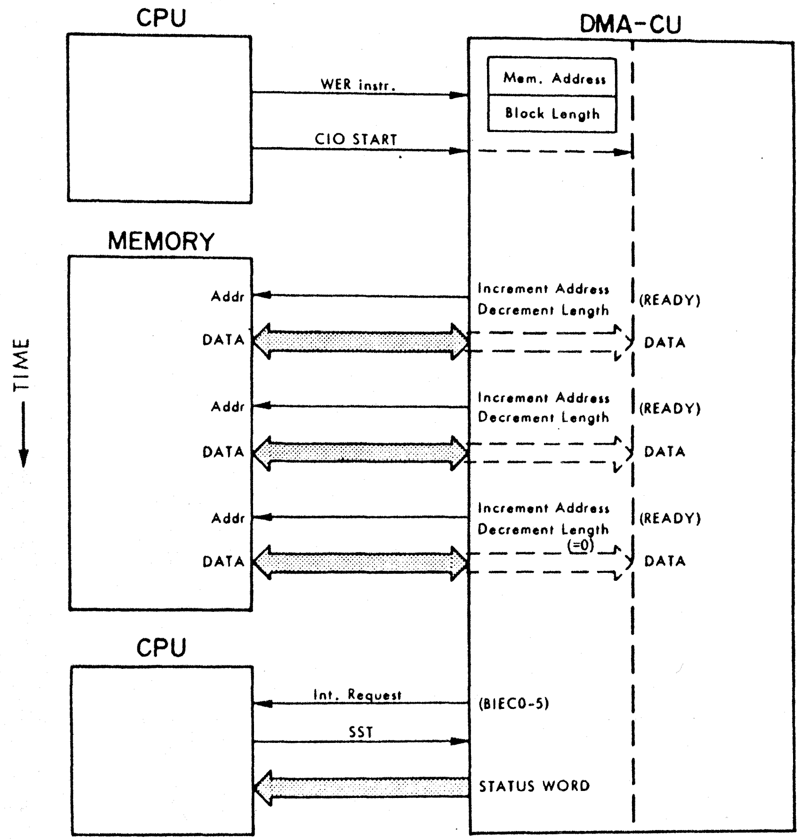


Figure 1-6 Direct Memory Access Channel Transfer

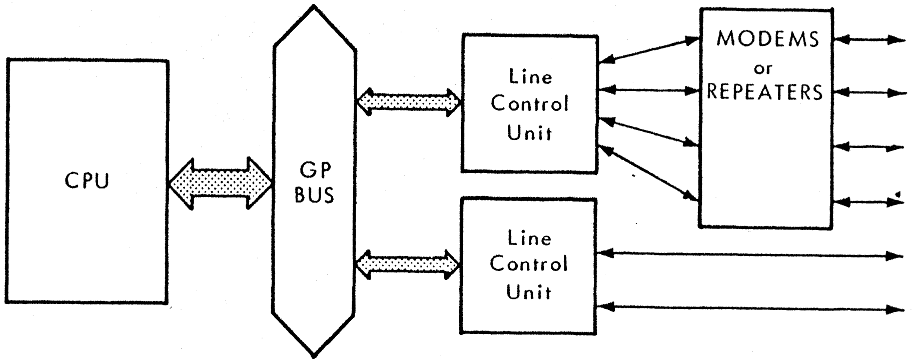
1.37 Direct Memory Access Channel

The DMA channel manages data transfers directly between memory and a single high-speed control unit (Figure 1-6). CPU registers are not used, and there is no need for program control except for starting the exchange and testing status at the completion. The DMA channel is a hardware channel included as a part of the high-speed control unit (DMA-CU card).

1.38 At the beginning of a transfer, the program uses two WER instructions to load the starting address and block length into the DMA control-word register. The DMA logic then provides all GP Bus timing signals to control the data transfers directly between the memory and the DMA CU. The DMA logic also updates the control-word register for each data word and detects when the complete block has been transferred. The data block transfer is ended with an SST command and status transfer between the CU and CPU.

1.39 Data Communications Channels

Data communication with the P856M/P857M system is accomplished with various Line Control Units operating via the programmed channel or the multiplexed IOP channel. The Line Control Units connect to the GP Bus in the same manner as any other control unit cards. The Line Control units are connected to the communications lines, either directly or via modem interface or repeater units (following diagram).



Some Line Control Units used with the system are:

- SLCU Synchronous line control units
- ALCU Asynchronous line control units
- ALM Asynchronous low speed multiplexers

1.40 CLOCKS

1.41 CPU Timing Clock

The CPU logic timing is controlled by the Sequensor which provides the clock signals AP, BP, and T1 through T10. The Sequensor is described in Section II.

1.42 Real Time Clock

A Real Time Clock produces an interrupt signal every 20ms. The real-time clock pulse is produced directly from the power supply and is described in Section V.

1.43 CONTROL PANEL

1.44 The standard control panel for the P856M system is the same as for the P852M. This panel is directly interchangeable between the two systems. The Extended control panel used by the P857M is available as an option for the P856M system. Description and operation of the control panel is provided in Section III.

1.45 The standard control panel for the P857M system is the Extended Control Panel. This panel has, in addition to the standard Data and Control facilities, complete Addressing facilities, including the ability to stop on preset addresses. The Data/Control half of the panel is the same as the P852M/P856M control panel, but with the addition of a TEST position on the key switch for performing automatic microdiagnostic tests. Description and operation is provided in Section III.

1.46 TESTING

The P856/7 contains an automatic testing feature in the form of a microprogrammed diagnostic built in to the CPU logic. This automatic microdiagnostic operates through the control panel to test approximately 70% of the CPU logic. Successful running of the microdiagnostic indicates that sufficient parts of the CPU function for loading of test programs. The automatic microdiagnostic tests and the test programs are described in Section III.

1.47 STATUS

System status is contained in a sixteen-bit Program Status Word (PSW). The use and operation of the status word is described in Section II (Paragraph 2.84). The status word contents are shown in the following diagram.

Program Status Word

PLR						CR		GF							
0	1	2	3	4	5	0	1	RUN	ENB	CPF	PFF	RTCF	PIF	░	FU

Interrupts

1 = User Mode
0 = System Mode

1.48 DATA FORMAT

The data are handled in 16-bit words with 15 magnitude bits and bit 0 as a sign bit. Bit 1 is the most-significant data bit. The data word is handled in the CPU and on the GP Bus with negative logic: a logic 1 is a low level (0V) and a logic 0 is a high level (+5V).

Data Word

Bit	0	1 15

Sign Bit
0 = positive
1 = negative

Magitude Bits

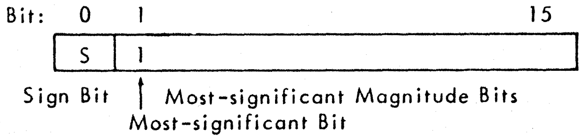
Least-significant bit

1.49 Double Precision

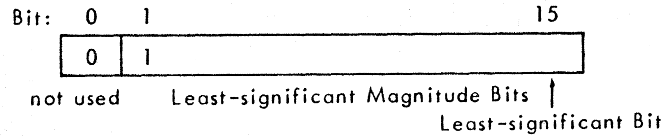
Double precision is obtained by utilising two successive words to obtain 30 magnitude bits. The sign bit and the 15 most-significant bits are in the first word; the 15 least-significant bits are in the second word. Bit 0 of the second word is not used, and is always zero.

1.50 Double precision is used for the product of the multiplication of two single-precision words, and for some other operations.

1st Word



2nd Word



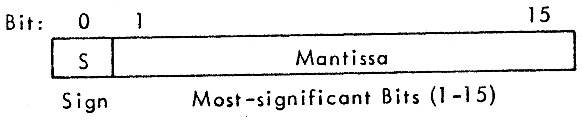
1.51 Logical Data

Logical data, such as the condition of sixteen binary indicators, can be stored in a single data word. This type of data is generally not treated arithmetically by the program but logically by means of boolean operations such as AND, OR, and Exclusive-OR. In this case, bit 0 of a word is not used as a sign bit.

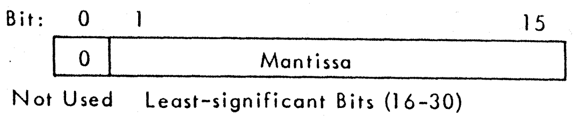
1.52 Floating-Point Data

Real, floating-point numbers are contained in three successive words. The mantissa is stored in the first two words as a double-precision integer. The third word contains the exponent, represented as a single-precision integer.

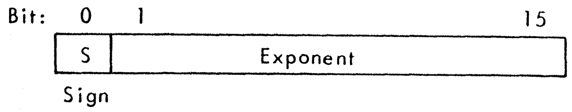
1st Word



2nd Word



3rd Word



1.53 Character Handling

Character handling is performed by some instructions. The right character of a word is the least significant (bits 8-15) and the left character is the most significant (bits 0-7).

1.54 OPERATING MODES

The CPU can operate in two basic modes: System Mode or User Mode. The mode is specified by bit 15 of the program status word (0 = system; 1 = user).

1.55 System Mode

The System Mode is reserved for the monitor program and system programs. In the System Mode, execution of the complete instruction set is allowed. This mode assures the system resource allocation, protected by the following privileged instructions:

- Control Instructions which modify the CPU state:
  HLT -- Halt
  INH -- Inhibit Interrupt
  RIT -- Reset Internal Interrupt
- All I/O Instructions:
  CIO, OTR, INR, SST, TST
- External Register Instructions:
  WER, RER
- (P857M) MMU-related instructions, Extended Load or Store, and Segment Table Load or Store:

ELR, EL, ESR, ES,
TLR, TL, TSR, TS

- Some other instructions are reserved for System Mode when they modify the contents of the stack pointer (A15). Refer to Table 1-1.

1.56 User Mode

This mode is reserved for the user program execution. If a program in User Mode attempts to execute a privileged instruction, a Trap routine is performed: the program parameters (P and PSW) are saved in the stack, and the program branches to the address contained in memory location 7E.

1.57 When a program running in User Mode needs system allocation, a Link to Monitor (LKM) instruction executes a call to the monitor which sets the CPU to System Mode.

1.58 When a program running in System Mode is complete, and ready to allow user programs to run, either a Set Mode (SMD) instruction or a Return (RTN) instruction with R2=15 can be used. The User-Mode indicator in the PSW is then set to 1.

1.59 INSTRUCTIONS

The P856/857 instruction set is divided into ten groups:

- load and store
- arithmetic
- logical
- character handling
- branch
- shift
- table handling
- control
- input/output
- external transfers

Table 1-1 lists the instructions and indicates the operating flow diagram for each instruction. The instruction formats are described in paragraph 1.62. Instructions are specified by addressing mode as well as the op-code. The addressing mode is described in paragraph 1.65 and listed in Table 1-2.

1.60 Invalid Instructions

An invalid instruction code initiates a Trap microprogram (paragraph 1-55, Figure 1-12). The program must determine what action (such as interrupt) to take following a Trap.

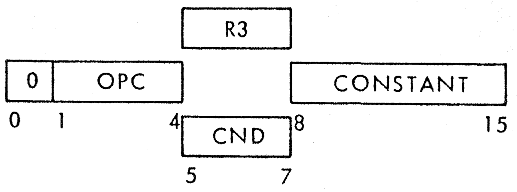
1.61 Some P857 instructions require the Floating Point Processor (FPP) or the Memory Management Unit (MMU). Both of these system extensions are optional. If the FPP is not present, the corresponding instructions initiate the Trap. If the MMU is not present, the corresponding instructions give no significant result when they are executed.

1.62 Instruction Format

There are two instruction formats, indicated by bit 0 of the instruction word:

- Format 0 Constant handling instructions.
- Format 1 Memory reference or register-to-register operations.

1.63 Format 0 Instructions (Type T8)



OPC : Operation code

R3 : Register (scratchpad A0-A7) on which the operation is performed.

CND : Condition for relative branching (when specified by OPC).

Table 1-1 P856/857 Instruction List

Flow Diagram 1:	Mnemonic	OPC	L/S (Bit 15)	Address Type	Instruction Name
Load and Store Instructions					
15	LDK	0	0	T8,2	Load constant
	LDR	0	0	T1,3	Load reg/reg; update stack pointer
	LD	0	0	T4-7	Load register
	STR	0	1	T3	Store reg/reg; update stack pointer
	ST	0	1	T4-7	Store register
23	MLK	7	0	T2	Multiple load constant
	MLR	7	0	T3	Multiple load register
	ML	7	0	T4-7	Multiple load
	MSR	7	1	T3	Multiple store register
	MS	7	1	T4-7	Multiple store
29	ELR	10	0	T3	Extended load register (MMU)
	EL	10	0	T4-7	Extended load (MMU)
	ESR	10	1	T3	Extended store register (MMU)
	ES	10	1	T4-7	Extended store (MMU)
Arithmetic Instructions					
17	ADK	2	0	T8,2	Add constant
	ADR	2	0/1	T1,3	Add reg/reg
	AD	2	0/1	T4-7	Add
	IMR	2	1	T3	Increment memory/reg
	IM	2	1	T4-7	Increment memory
18	SUK	3	0	T8,2	Subtract constant
	SUR	3	0/1	T1,3	Subtract reg/reg
	SU	3	0/1	T4-7	Subtract
	NGR	3	1	T1	Negate register
	C2R	3	1	T3	Two's complement/reg
	C2	3	1	T4-7	Two's complement
25	MUK	8	0	T2	Multiply with constant
	MUR	8	0	T1,3	Multiply reg/reg
	MU	8	0	T4-7	Multiply
26	DVK	9	0	T2	Divide with constant
	DVR	9	0	T1,3	Divide reg/reg
	DV	9	0	T4-7	Divide
29	DAK	10	0	T2	Double add with constant
	DAR	10	0	T1,T3	Double add reg/reg
	DA	10	0	T4-7	Double add
Arithmetic Instructions					
30	DSK	11	0	T2	Double subtract with constant
	DSR	11	0	T1,3	Double subtract reg/reg
	DS	11	0	T4-7	Double subtract
28	FFL	9	0	T1	Integer to floating point (FPP)
	FFX	9	1	T1	Floating point to integer (FPP)
27	FADR	9	0/1	T3	Floating-point addition/reg (FPP)
	FAD	9	0/1	T4-7	Floating-point addition (FPP)
	FSUR	9	0/1	T3	Floating-point subtract/reg (FPP)
	FSU	9	0/1	T4-7	Floating-point subtract (FPP)
	FMUR	9	0/1	T3	Floating-point multiply/reg (FPP)
	FMU	9	0/1	T4-7	Floating-point multiply (FPP)
	FDVR	9	0/1	T3	Floating-point divide/reg (FPP)
	FDV	9	0/1	T4-7	Floating-point divide (FPP)
Logical Instructions					
19	ANK	4	0	T8,2	Logical AND with constant (R3≠0, R1≠0)
	ANR	4	0/1	T1,3	Logical AND reg/reg (R1≠0)
	AN	4	0/1	T4-7	Logical AND (R1≠0)
20	ORK	5	0	T8,2	Logical OR with constant (R3≠0, R1≠0)
	ORR	5	0/1	T1,3	Logical OR reg/reg (R1≠0)
	OR	5	0/1	T4-7	Logical OR (R1≠0)
	XRK	6	0	T8,2	EXclusive OR with constant (R1≠0)
	XRR	6	0/1	T1,3	EXclusive OR reg/reg (R1≠0)
	XR	6	0/1	T4-7	EXclusive OR (R1≠0)
32	CWC	13	X	T8	Compare word to short constant
	CWK	13	0	T2	Compare word with constant
	CWR	13	0/1	T1,3	Compare word reg/reg
	CW	13	0	T4-7	Compare word
37	C1R	15	0/1	T1,3	One's complement reg/reg
	C1	15	0/1	T4-7	One's complement
19	TM	4	1	T1	Test mask
20	TNM	6	1	T1	Test not mask
19	CMR	4	1	T3	Clear memory/reg
	CM	4	1	T4-7	Clear memory

Flow Diagram 1:	Mnemonic	OPC	L/S (Bit 15)	Address Type	Instruction Name	Bits 8	9	10
					Character Handling Instructions			
31	ECR	12	0	T1	Exchange character reg/reg (R1≠0)			
	LCK	12	0	T2	Load character with constant (R1≠0)			
	LCR	12	0	T3	Load character/reg (R1≠0)			
	LC	12	0	T4-7	Load character			
	SCR	12	1	T3	Store character/reg (R1≠0)			
	SC	12	1	T4-7	Store character			
32	CCK	13	1	T2	Compare character/constant (R1≠0)			
	CCR	13	1	T3	Compare character/reg (R1≠0)			
	CC	13	1	T4-7	Compare character (R1≠0)			
					Branch Instructions			
16	AB	1	0	T8,2	Absolute conditional branch			
	ABR	1	0/1	T1,3	Absolute conditional branch to reg			
	ABI	1	0	T4-7	Absolute branch			
29	RF	10	0	T8	Relative forward conditional branch			
30	RB	11	0	T8	Relative backward conditional branch			
35	CF	14	1	T2	Call function (direct) (R1≠0)			
	CFR	14	1	T1,3	Call function/reg (R1≠0)			
	CFI	14	1	T4-7	Call function (via memory) (R1≠0)			
33	RTN	14	0	T3	Return (R1=0)			
34	EXK	14	1	T2	Execute constant (R1=0)			
	EXR	14	1	T1,3	Execute/register (R1=0)			
	EX	14	1	T4-7	Execute (R1=0)			
					Shift Instructions	8	9	10
21	SLA	7	X	T8	Single left arithmetic (R3≠0)	0	0	0
	SRA	7	X	T8	Single right arithmetic (R3≠0)	0	0	1
	SLL	7	X	T8	Single left logical (R3≠0)	0	1	0
	SRL	7	X	T8	Single right logical (R3≠0)	0	1	1
	SLC	7	X	T8	Single left circular (R3≠0)	1	1	0
	SRC	7	X	T8	Single right circular (R3≠0)	1	1	1
	SLN	7	0	T8	Single left, normalize (R3≠0)	1	0	0
	SRN	7	0	T8	Single right, normalize (R3≠0)	1	0	1
22	DLA	7	X	T8	Double left arithmetic (R3=0)	0	0	0
	DRA	7	X	T8	Double right arithmetic (R3=0)	0	0	1
	DLL	7	X	T8	Double left logical (R3=0)	0	1	0
	DRL	7	X	T8	Double right logical (R3=0)	0	1	1
	DLC	7	X	T8	Double left circular (R3=0)	1	1	0
	DRC	7	X	T8	Double right circular (R3=0)	1	1	1

Flow Diagram 1:	Mnemonic	OPC	L/S (Bit 15)	Address Type	Instruction Name	Bits 8	9	10
					Shift Instructions	8	9	10
22	DLN	7	0	T8	Double left, normalize	1	0	0
	DRN	7	0	T8	Double right, normalize	1	0	1
					Table Handling Instructions			
36	MVF	14	0	T8	Move table forward	0	0	0
37	MVB	15	0	T8	Move table backward	0	0	0
36	MVSU	14	0	T8	Move table, system to user (MMU)	1	0	0
37	MVUS	15	0	T8	Move table, user to system (MMU)	1	0	0

Flow Diagram 1:	Mnemonic	OPC	L/S (Bit 15)	Address Type	Instruction Name	Bits 8	9	10	11	12	13	14	15
					Control Instructions	8	9	10	11	12	13	14	15
19	HLT	4		T8	Halt	0	1	1	1	1	1	1	1
	INH	4		T8	Inhibit interrupt	1	0	1	1	1	1	1	1
	RIT	4	1	T8	Reset internal interrupt	1	1	DA					1
20	ENB	5		T8	Enable interrupt	0	1	0	0	0	0	0	0
	LKM	5		T8	Link to monitor	0	0	0	0	0	1	0	0
	SMD	5		T8	Set mode	0	0	0	0	0	0	0	1

Flow Diagram 1:	Mnemonic	OPC	L/S (Bit 15)	Address Type	Instruction Name	Bits 8	9
					Input/Output Instructions (R3≠0)	8	9
24	CIO	8	X	T8	Control I/O (DA≠0)	1	n
	OTR	8	X	T8	Output from register (DA≠0)	0	n
	INR	9	X	T8	Input to register	0	n
	SST	9	X	T8	Sense status	1	1
	TST	9	X	T8	Test status	1	0

Flow Diagram 1:	Mnemonic	OPC	L/S (Bit 15)	Address Type	Instruction Name
					External Transfer Instructions
33	WER	14	X	T8	Write external register (R3≠0)
37	RER	15	X	T8	Read external register (R3≠0)
27	FLDR	8	0	T3	Floating-point load/reg (R1=2) (FPP)
	FLD	8	0	T4-7	Floating-point load (R1=2) (FPP)
	FSTR	8	1	T3	Floating-point store/reg (R1=2) (FPP)
	FST	8	1	T4-7	Floating-point store (R1=2) (FPP)
23	TLR	7	0	T3	Segment table load/reg (R1=0) (MMU)
	TL	7	0	T4-7	Segment table load (R1=0) (MMU)
	TSR	7	1	T3	Segment table store/reg (R1=0) (MMU)
	TS	7	1	T4-7	Segment table store (R1=0) (MMU)

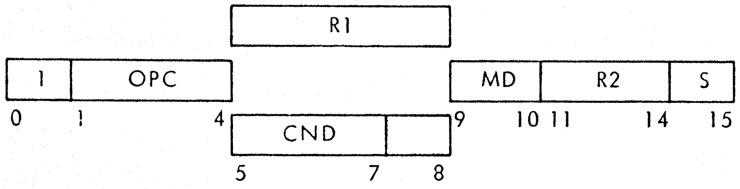
| = P857 only

CND 5 6 7	Branch

0 n n
1 n n
1 1 1
	if CR = bits 6,7
	if CR ≠ bits 6,7
	unconditional

CONSTANT: An 8-bit, positive constant, branch displacement, or OPC extension.

1.64 Format 1 Instructions (Types T1-T7)



OPC : Operation Code

R1 : Register (scratchpad A0-15) on which the operation is performed:
bit 8 = 0 : A0-7
bit 8 = 1 : A8-15

CND : Condition for absolute branching (when specified by OPC).
(conditions same as for format 0)

MD : Addressing mode (Table 1-2).

R2 : Register (scratchpad A0-15) of 2nd operand or address of 2nd operand:
bit 14 = 0 : A0-7
bit 14 = 1 : A8-15

S : Store bit for memory reference instructions:
0 = store result in R1
1 = store result in memory

1.65 Addressing

Format 0 instructions (type 8) address the operand with the R1 field; no second operand is used. Format 1 instructions address the first operand with the R1 field; the second operand is addressed according to the addressing type, T1-T7, as listed in Table 1-2.

Table 1-2 Addressing Types

Type	Format 1 (K00)			
	MD 9 10 (K9 K10)	R2 11 12 13 14	Effective Address of Operand	
T1	0 0	x x x x	R2	Register-to-Register. R2 contains the operand .
T2	0 1	0 0 0 0 (OR2)	P	Long Constant. The following word(after the instruction) is the operand.
T3	0 1	non-zero $\overline{\text{(OR2)}}$	(R2)	Address in Register. R2 contains the address (A0-A15) of the operand.
T4	1 0	0 0 0 0	(P)	Address in Next Word. The following word is the operand address.
T5	1 0	non-zero $\overline{\text{(OR2)}}$	(P) + (R2)	Indexed Address. The following word, indexed by (A0-A15), as specified by R2, contains the operand address.
T6	1 1	0 0 0 0	[(P)]	Indirect Address. The following word specifies the location containing the operand address.
T7	1 1	non-zero $\overline{\text{(OR2)}}$	[(P) + (R2)]	Indirect Indexed Address. The following word, indexed by (A0-A15), specifies the location containing the operand address.
T8	Format 0 ($\overline{\text{K00}}$)			Short Constant. No 2nd operand is used.

1.66 All addressing uses a 16-bit address word, although only the 15 high-order bits are used for memory selection and for word-handling instructions. For character-handling instructions, the least-significant address bit specifies the character, as follows: bit 15 = 0: left character ; bit 15 = 1: right character.

1.67 OPERATION SEQUENCES

1.68 Microprograms, Microinstructions

CPU operations are controlled by microinstructions (μ Inst) located in the Microinstruction Store (Control ROM). Each microinstruction comprises a single 48-bit word divided into 14 command fields.

1.69 A microprogram performs one part of an instruction (e.g. indirect addressing or execution) or one operating sequence (e.g. Initial Program Loading or an Interrupt routine). The microprogram may consist of a single microinstruction (Fetch, SUK, etc.) or a group of microinstructions accessed in a specified sequence. This sequence may vary according to conditions specified in the microinstructions. An example of microinstructions and microprograms grouped into an instruction is shown in Figure 1-7, Operation Terminology.

1.70 Flow Diagrams

General operational flow for the CPU is shown in Figure 1-8. Detailed flow diagrams for each block are referenced on Figure 1-8 and, for the execute-instruction sequences, in the instruction list (Table 1-2). A key to the flow diagrams is provided in Figure 1-7.

1.71 Figure 1-8 shows the general sequence of computer operations. The computer continuously performs microinstructions. The sequences in which the microinstructions are performed are determined by microinstruction addressing (paragraph 2.26).

1.72 If no program is running and the computer is not being operated, it cycles through the Idle loop. RUNF is described with the PSW, paragraph 2.90. Machine-state-pointer control is described with the microinstruction addressing. RUNF is set by pressing START on the control panel. If the control panel is used (Section III), PUP is set in the CPU (logic diagram CC) and the operation now cycles through the control-panel path, beginning with microinstruction /010. Either automatic restart or IPL may be initiated through this path.

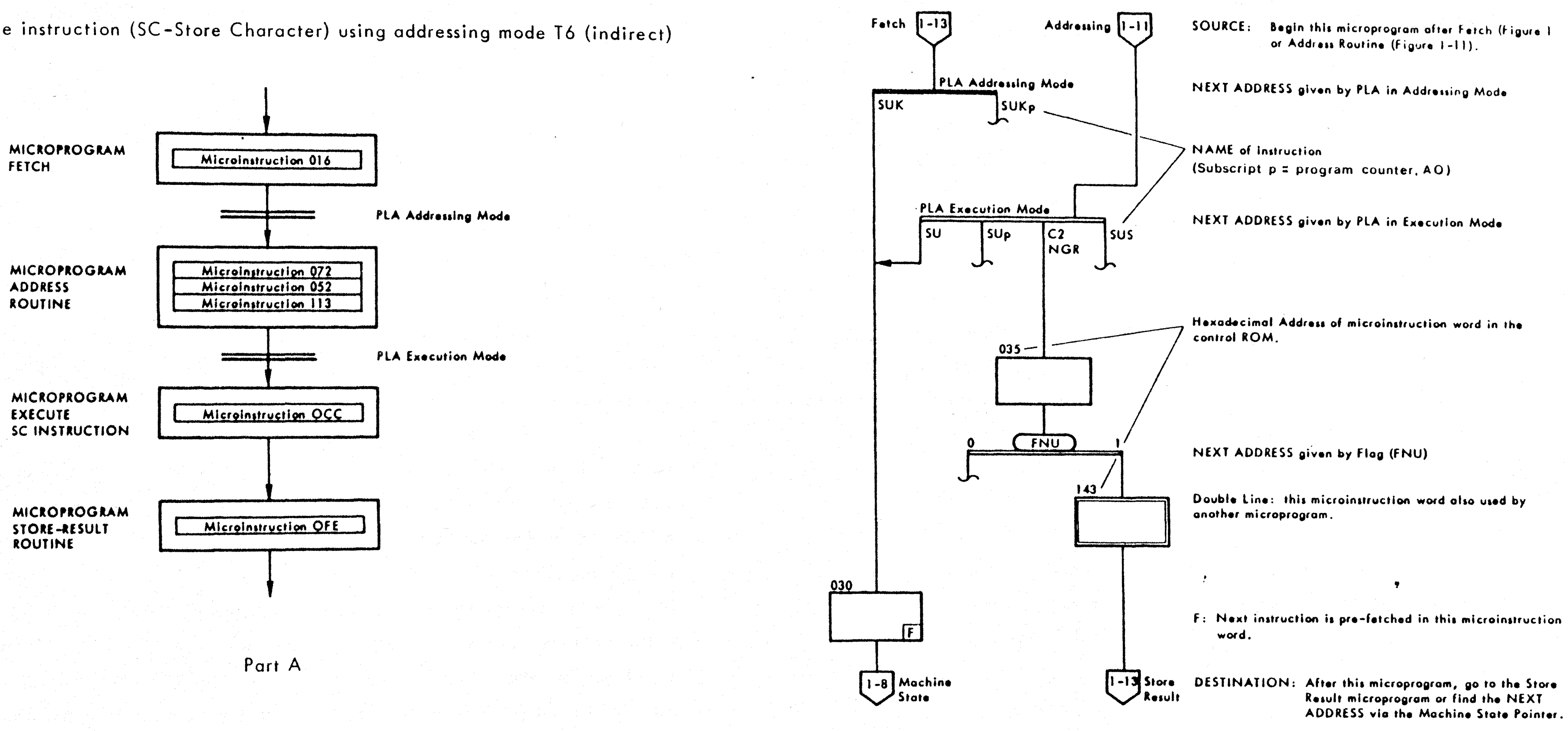
1.73 If RUNF is set and the control panel is not selected, the machine-state-pointer tests for interrupts, and executes the interrupt routine if necessary. KRY is set when the K-register is loaded with the instruction word. KRY is set by the Fetch command during the Fetch routine, some control-panel routines, EX instruction, and some tests.

1.74 Before KRY is set, the computer loops directly to Fetch to load the instruction word into the K register. The instruction must be requested from memory during a preceeding microinstruction. When an instruction is loaded, KRY is set and the PLA addressing mode is used to select the next microinstruction address. For most instructions, the addressing routine is used to obtain the operand. Instruction word addressing and decoding is described in paragraphs 2.37 and 2.48.

1.75 Repeated microinstructions of a routine are selected with explicit addressing (the microinstruction includes the address of the next microinstruction). Microprogram decision tests may modify the explicit addressing with a flag bit (SNA Flag mode). At the end of a microprogram routine, a Bus Request is made for the next instruction word and the Fetch routine is then used to load the instruction. If the machine state pointers allow (program still running, no control-panel operations or interrupts pending), the computer selects the next microinstruction address via the PLA addressing, and continues with the next microprogram.

1.76 Data paths are shown in Figures 2-1 and 2-4. The microinstructions within each microprogram or routine control the data paths and control the operations performed on the data. These controls are described (in Section II) for each of the logic blocks shown on the data-path diagrams. The data handling is also listed on the microprogram flow charts with terminology such as: M + 2 → M, where the data in the M register loops through the data paths shown on Figures 2-1, 2-4 and back to M, with a left-shift being performed enroute to produce the +2.

Example of one instruction (SC-Store Character) using addressing mode T6 (indirect)


MICROPROGRAM FETCH
Microinstruction 016
PLA Addressing Mode
MICROPROGRAM ADDRESS ROUTINE
Microinstruction 072
Microinstruction 052
Microinstruction 113
PLA Execution Mode
MICROPROGRAM EXECUTE SC INSTRUCTION
Microinstruction OCC
MICROPROGRAM STORE-RESULT ROUTINE
Microinstruction OFE
Fetch
1-13
Addressing
1-11
PLA Addressing Mode
SUK
SUKp
PLA Execution Mode
SU
SUp
C2
NGR
SUS
035
FNU
0
1
143
030
F
1-8
Machine State
1-13
Store Result
SOURCE: Begin this microprogram after Fetch (Figure I or Address Routine (Figure 1-11).
NEXT ADDRESS given by PLA in Addressing Mode
NAME of Instruction
(Subscript p = program counter, AO)
NEXT ADDRESS given by PLA in Execution Mode
Hexadecimal Address of microinstruction word in the control ROM.
NEXT ADDRESS given by Flag (FNU)
Double Line: this microinstruction word also used by another microprogram.
F: Next instruction is pre-fetched in this microinstruction word.
DESTINATION: After this microprogram, go to the Store Result microprogram or find the NEXT ADDRESS via the Machine State Pointer.

Part A

Address Type Designations	
T1 (2nd Operand in R2):	T2-7
RT1 RT1P - R2=0, thus operand in P. RT1D - Double Length; (R2) ← Q, (R2+1) ← M RT1DP - Double Length and R2=0	RT2-7 Result ← R1 RT2-7S - Store; Result ← memory RT2-7C - Constant T3A — R2 ≠ A15 T3B — R2 = A15

Part B

Figure 1-7 Operation Terminology, Flow Diagram Key

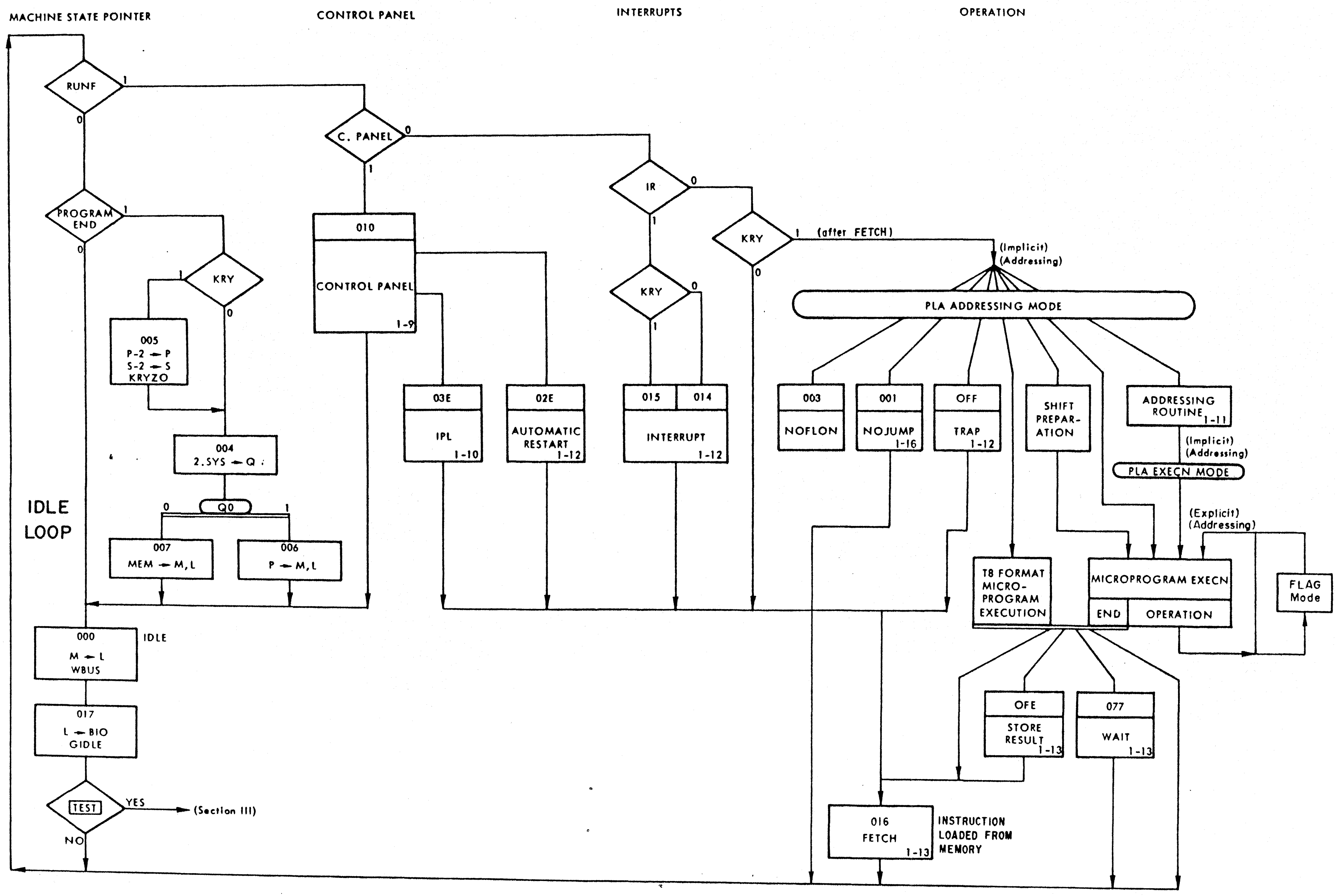


Figure 1-8 CPU Operational Flow and Machine State Pointe

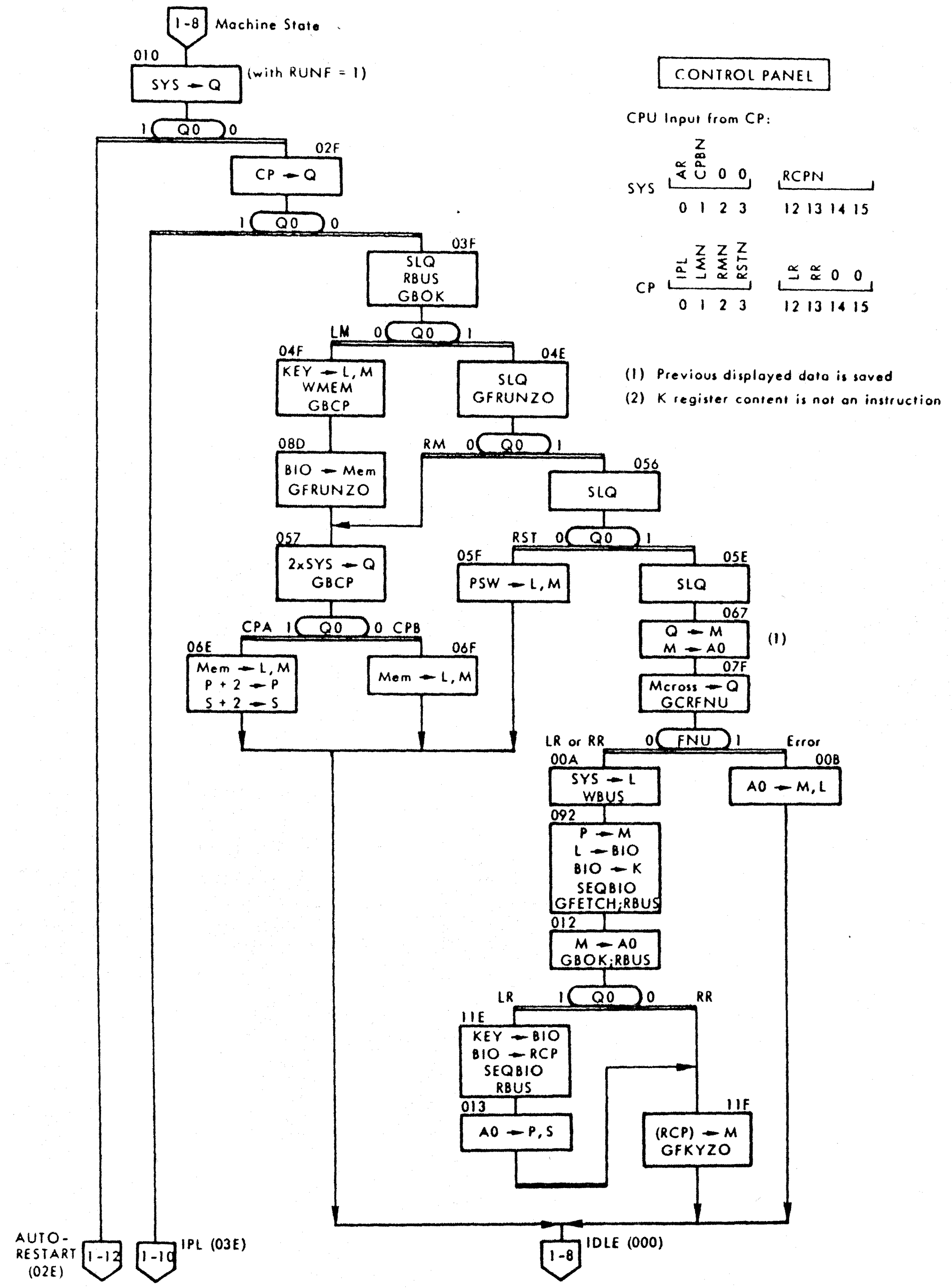
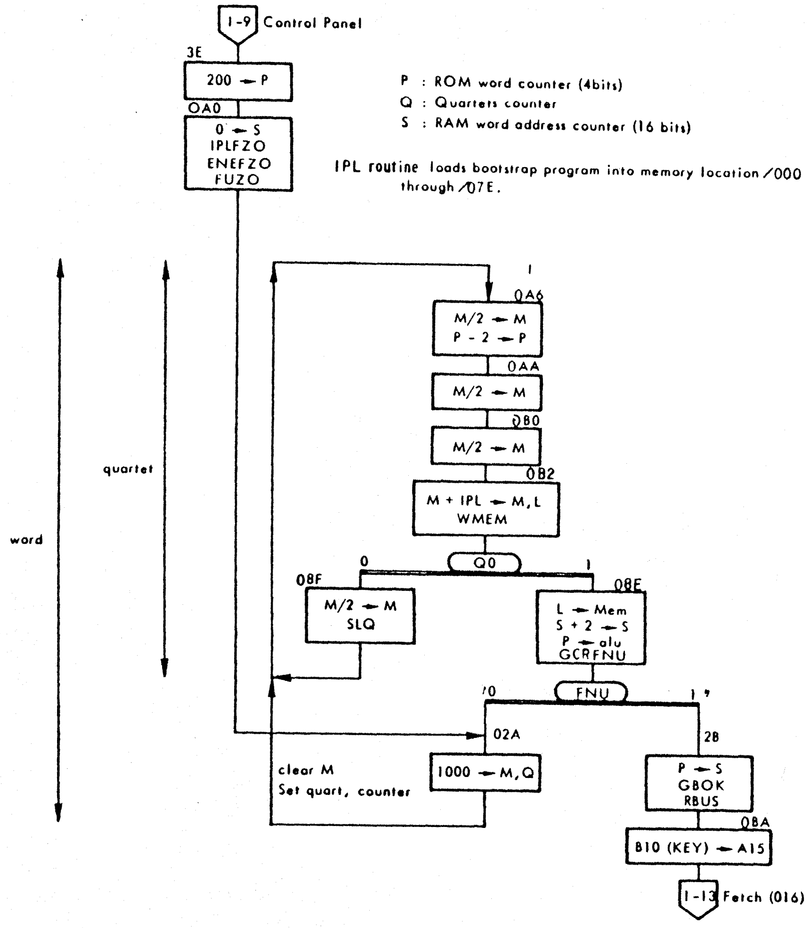


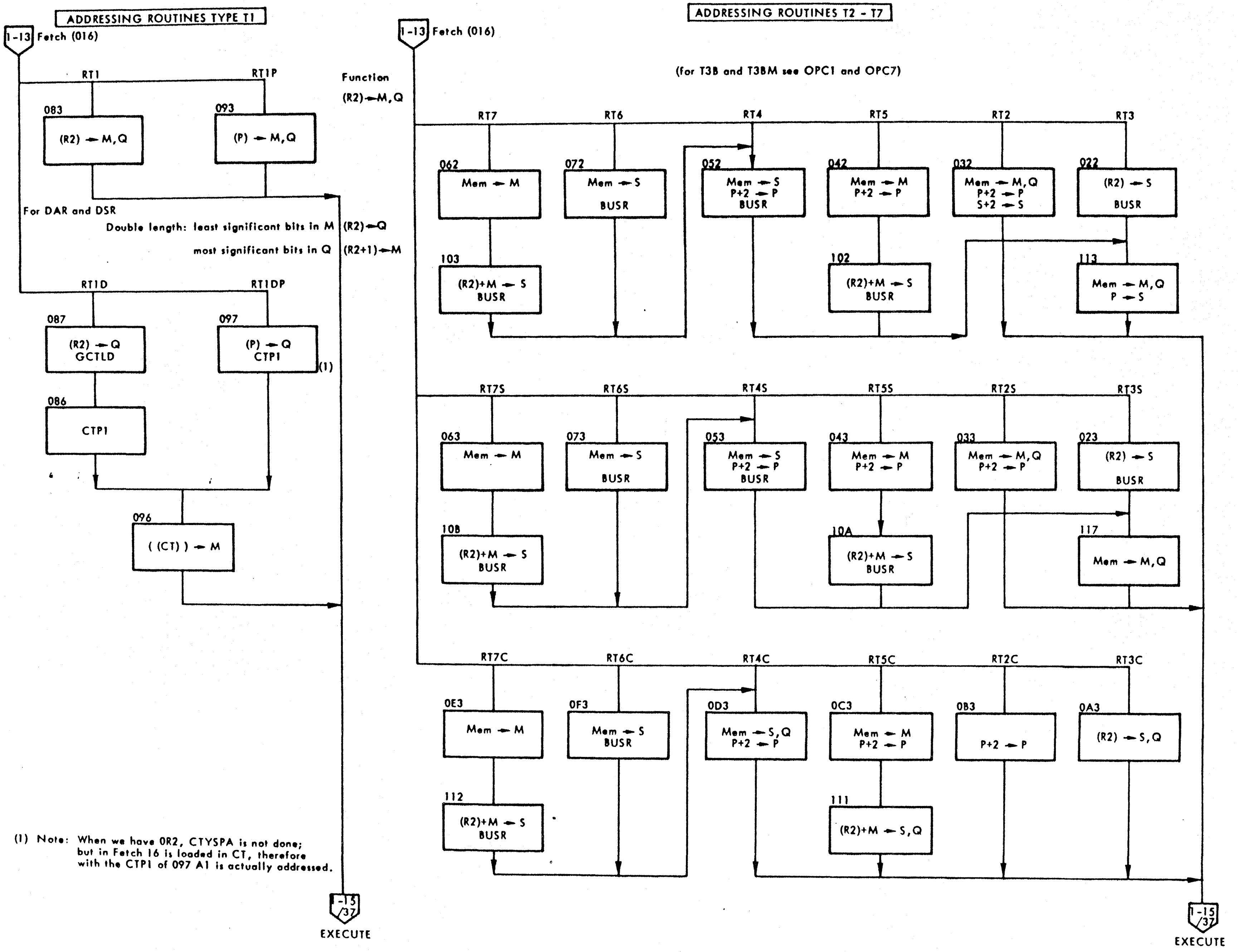
Figure 1-9 Control Panel



NOTE:

- Bootstrap listing in Table 2 - 10
- Loading routine in Paragraph 3.14

Figure 1-10 IPL



(1) Note: When we have 0R2, CTYSPA is not done;
but in Fetch 16 is loaded in CT, therefore
with the CTP1 of 097 A1 is actually addressed.

Figure 1-11 Addressing

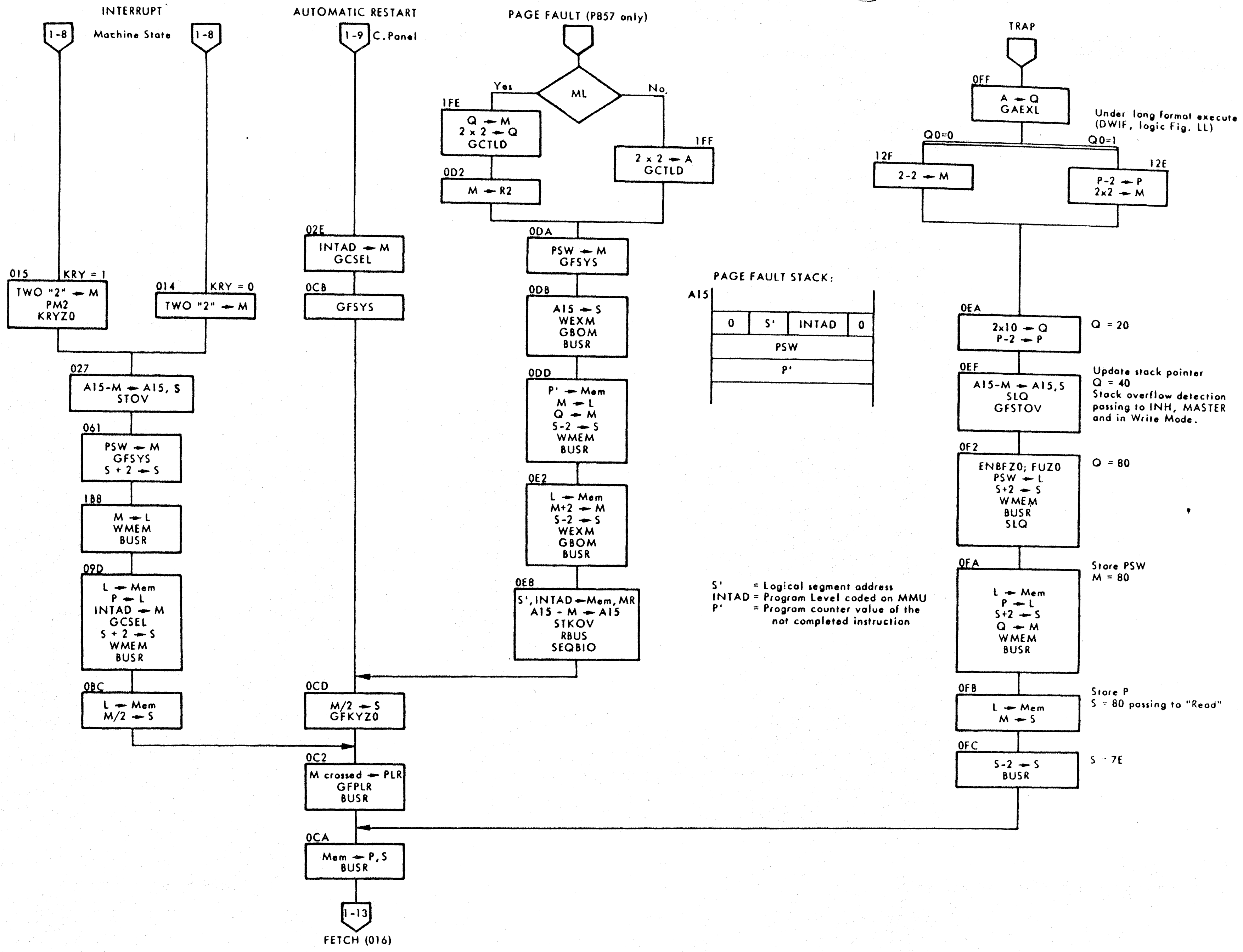
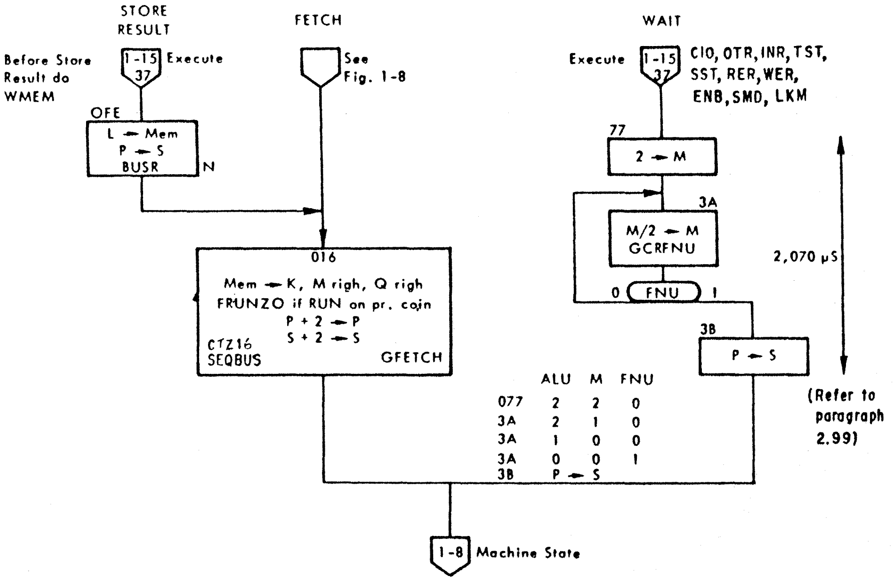


Figure 1-12 Interrupt, Restart, Fault, Trap



	ALU	M	FNU
077	2	2	0
3A	2	1	0
3A	1	0	0
3A	0	0	1
3B	P ← S		

Figure 1-13 Store, Fetch, Wait

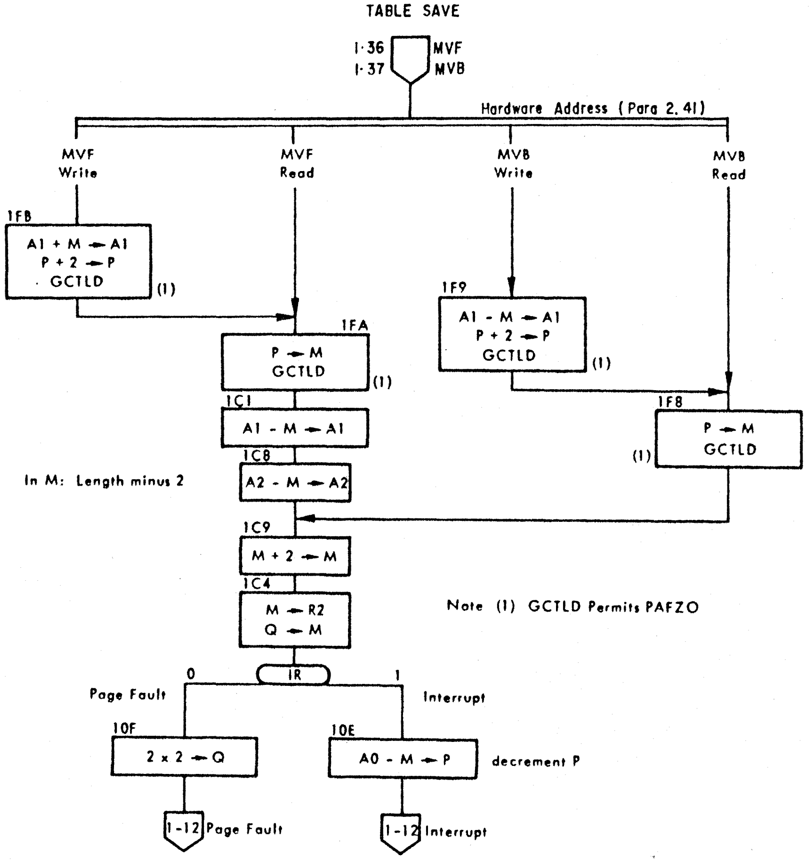


Figure 1-14 Table Save (P857 only)

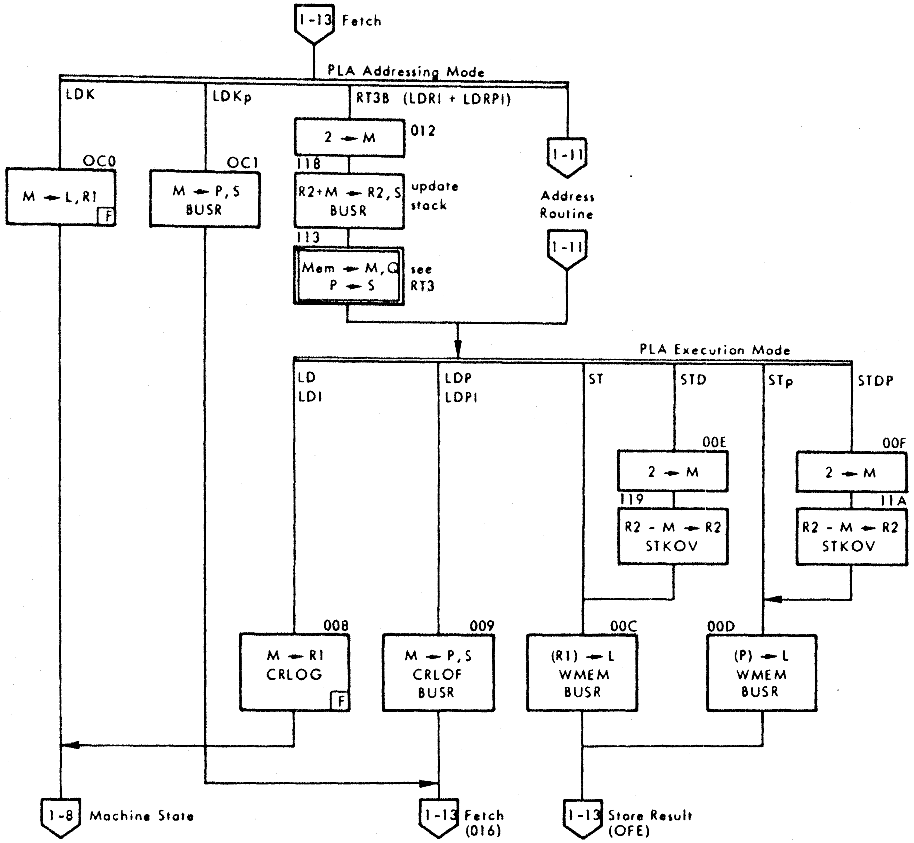


Figure 1-15 OPC 0 (LD, ST)

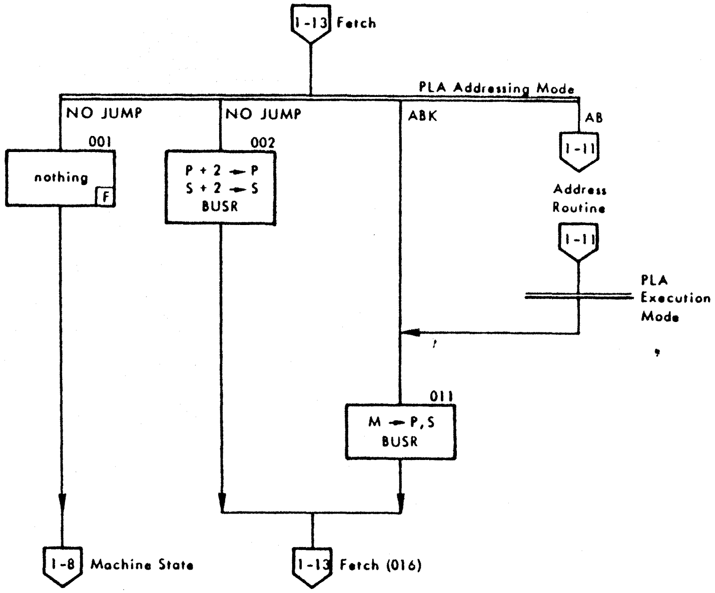


Figure 1-16 OPC 1 (AB)

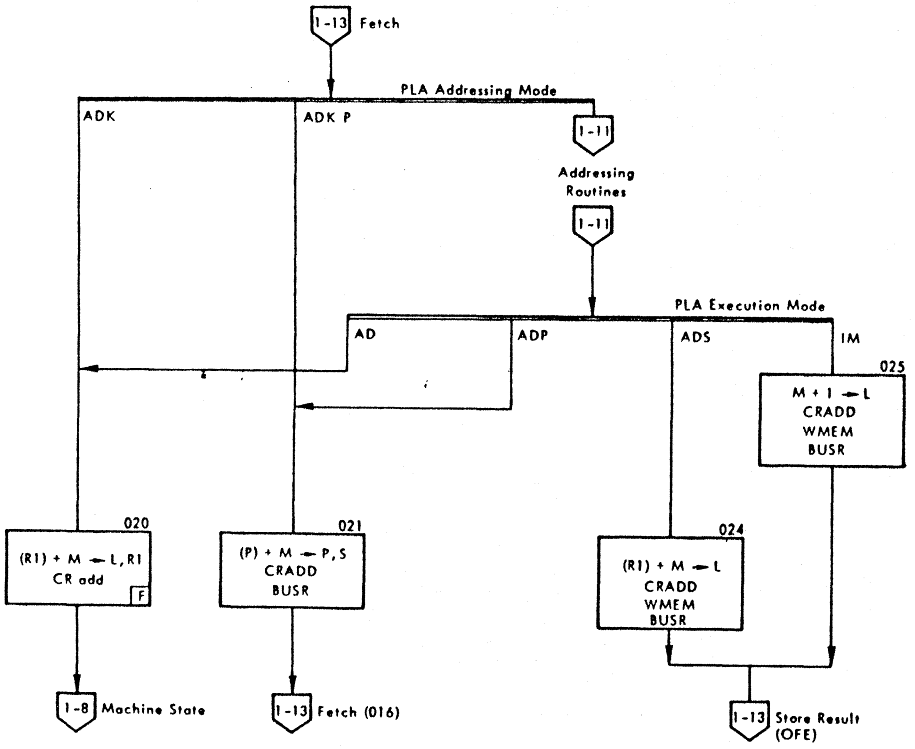
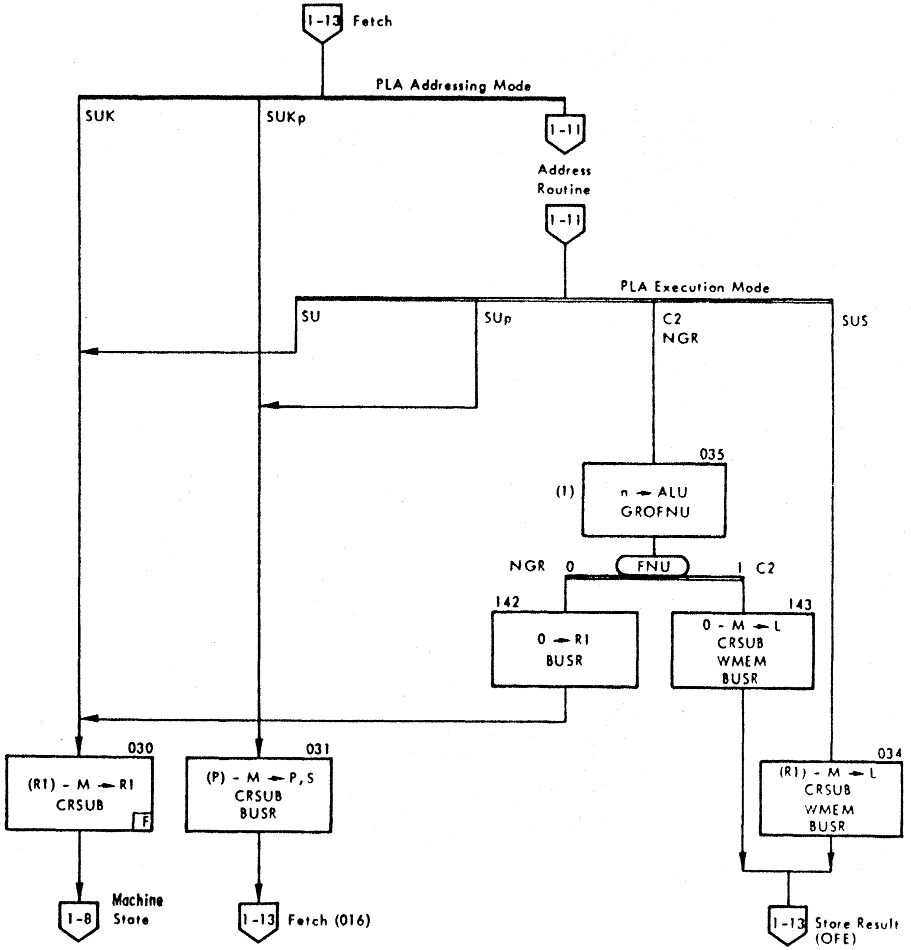
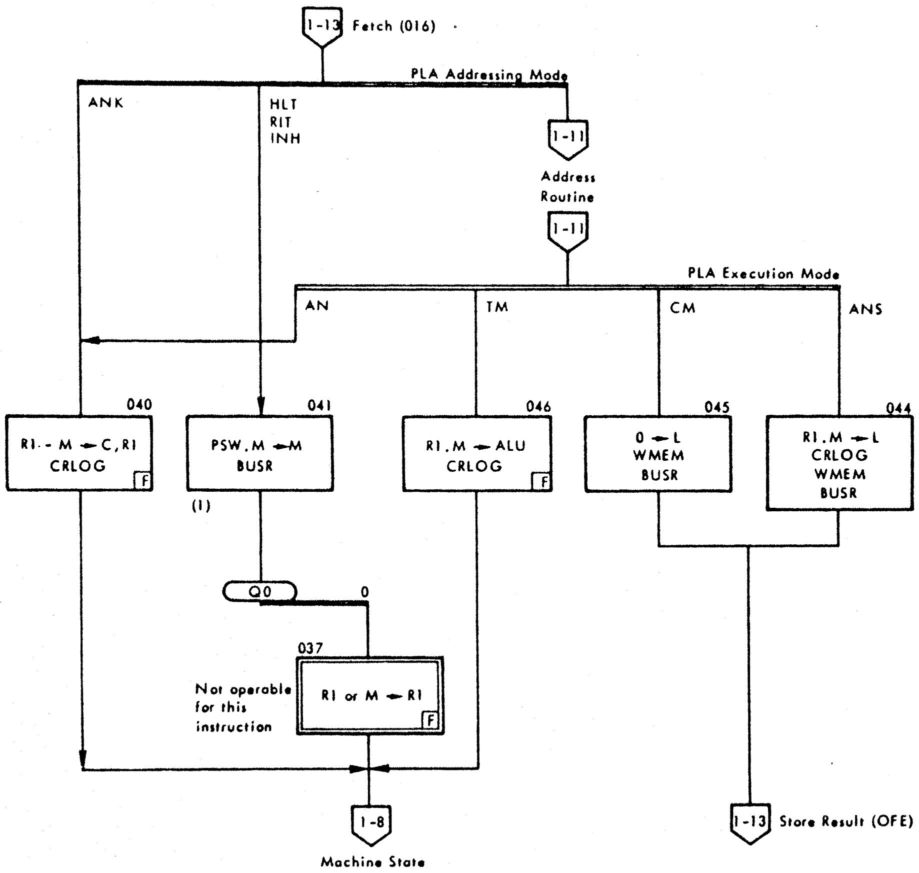


Figure 1-17 OPC 2 (AD, IM)



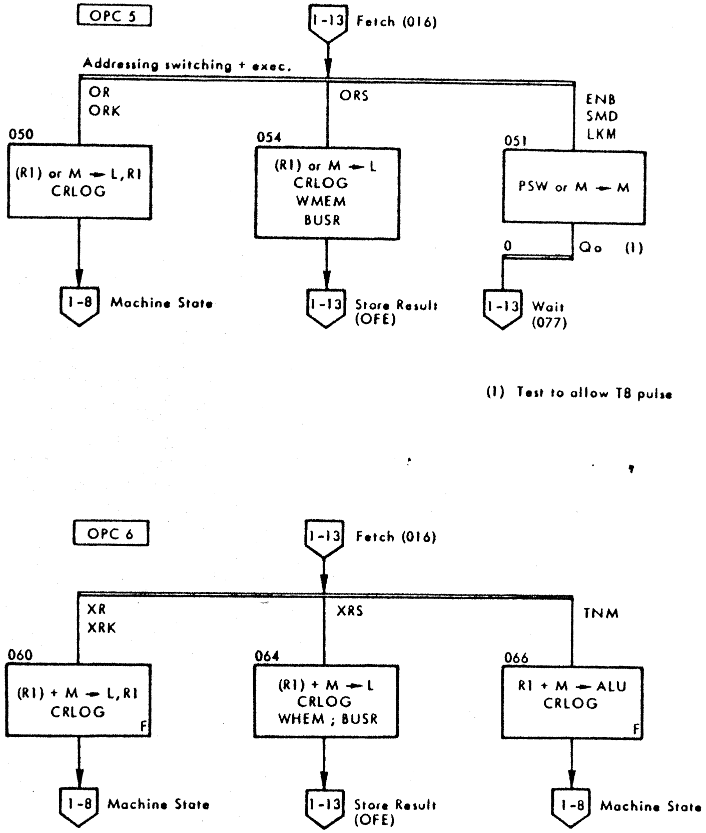
(1) n = K05, K06, K07, K08, 0 if n = 0 (OR1) ⇒ C2 if n ≠ 0 (OR1/) ⇒ NGR

Figure 1-18 OPC 3 (SU, NG, C2)



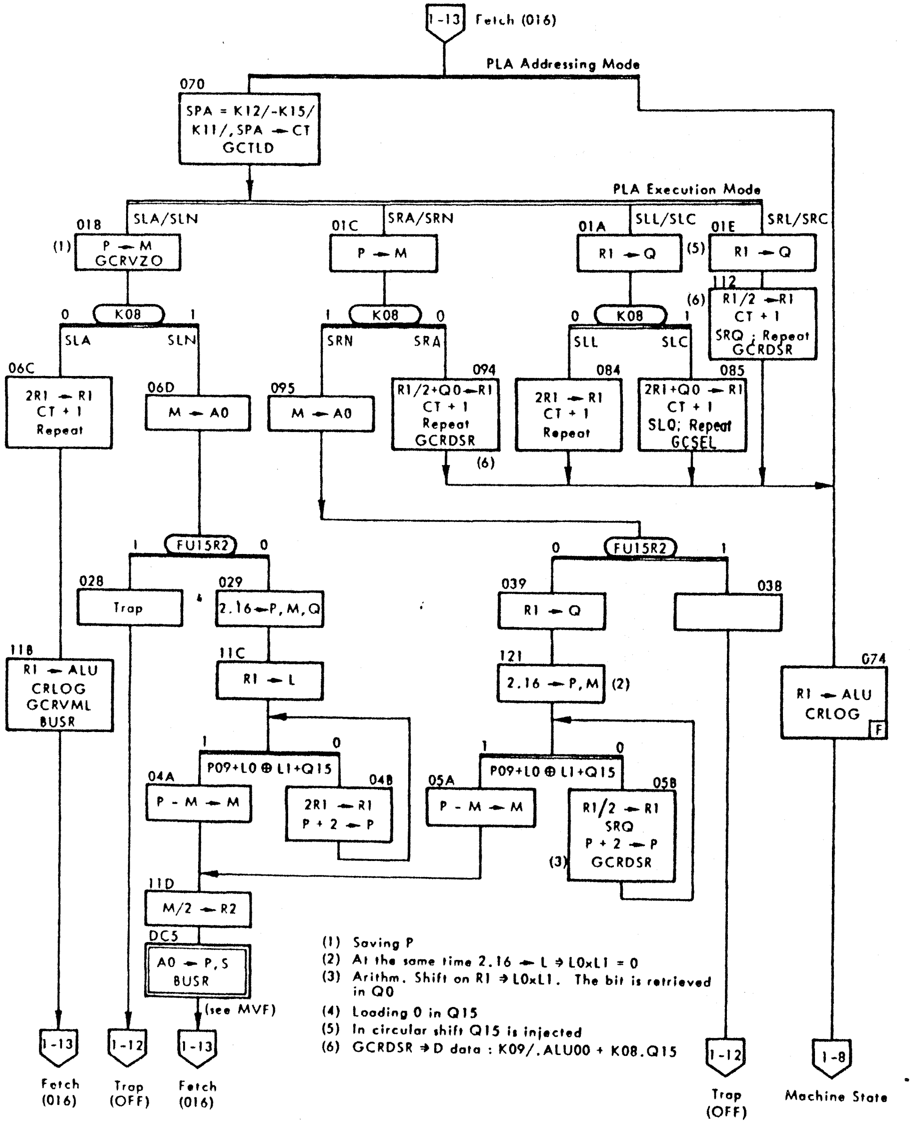
(1) Note : - Test Q0 to have a T8 pulse CLG = REPSW. T8
- M register is updated because PFF is reloaded by M11.

Figure 1-19 OPC 4 (AN, TM, CM, AC, HLT, INH, RIT)



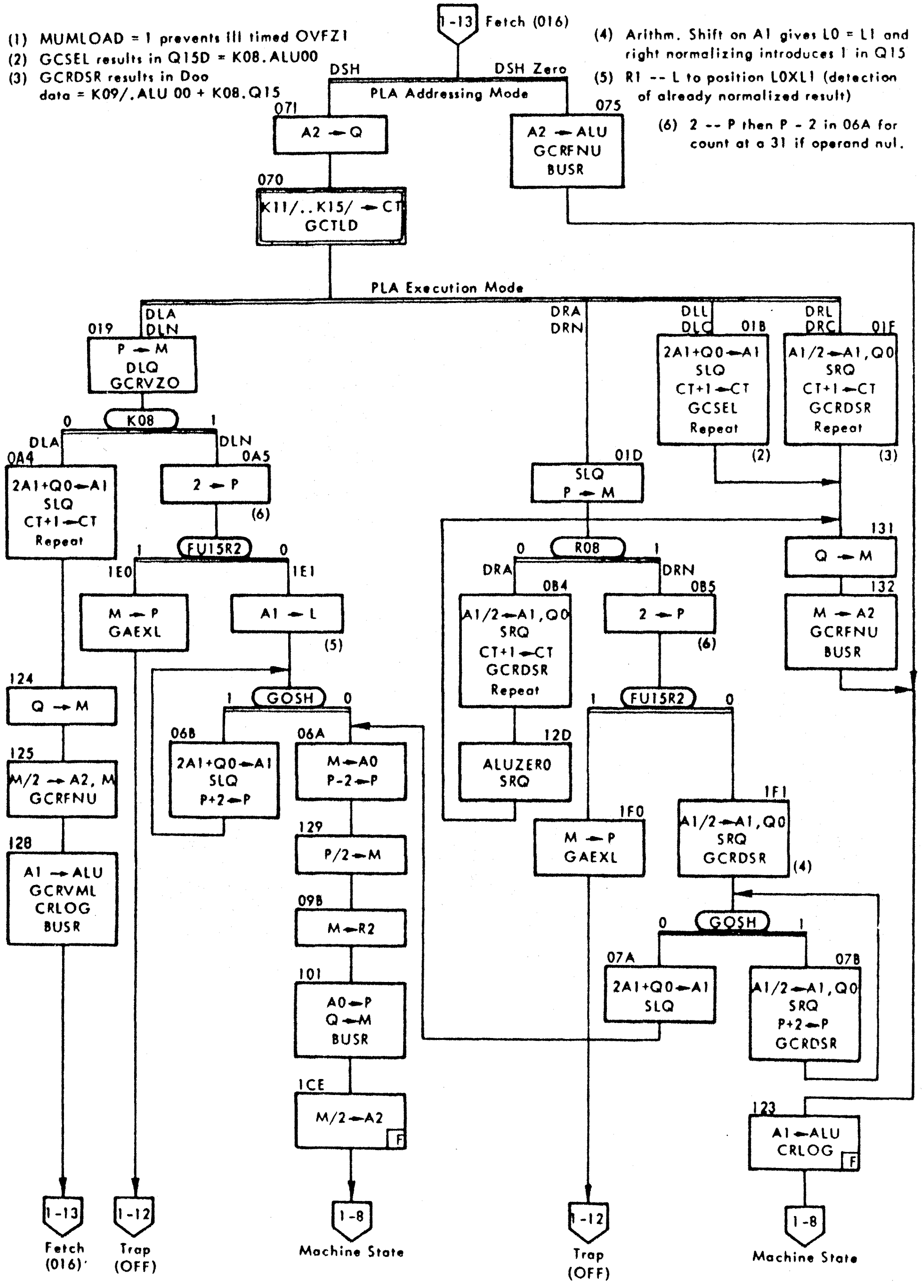
(1) Test to allow T8 pulse

Figure 1-20 OPC 5, 6 (OR, ENB, LKM, SMD, XR, TNM)



(1) Saving P
(2) At the same time 2.16 ← L ⇒ L0xL1 = 0
(3) Arithm. Shift on R1 ⇒ L0xL1. The bit is retrieved in Q0
(4) Loading 0 in Q15
(5) In circular shift Q15 is injected
(6) GCRDSR ⇒ D data : K09/.ALU00 + K08.Q15

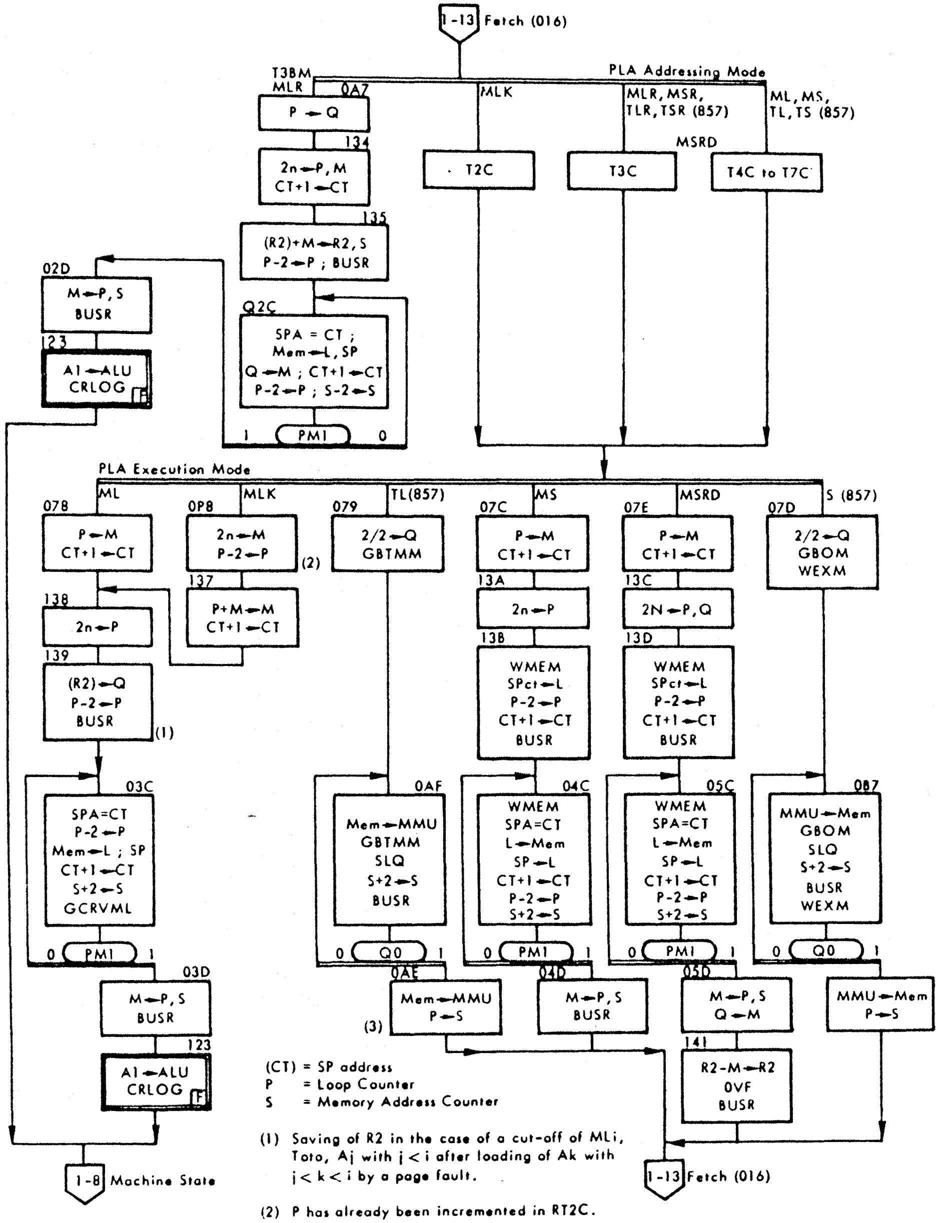
Figure 1-21 OPC 7 Single Shifts (SL, S R)



(1) MUMLOAD = 1 prevents III timed OVFZ1
(2) GCSEL results in Q15D = K08.ALU00
(3) GCRDSR results in Doo data = K09/.ALU 00 + K08.Q15

(4) Arithm. Shift on A1 gives L0 = L1 and right normalizing introduces 1 in Q15
(5) R1 -- L to position L0XL1 (detection of already normalized result)
(6) 2 -- P then P - 2 in 06A for count at a 31 if operand nul.

Figure 1-22 OPC 7 Double Shifts (DL, DR)



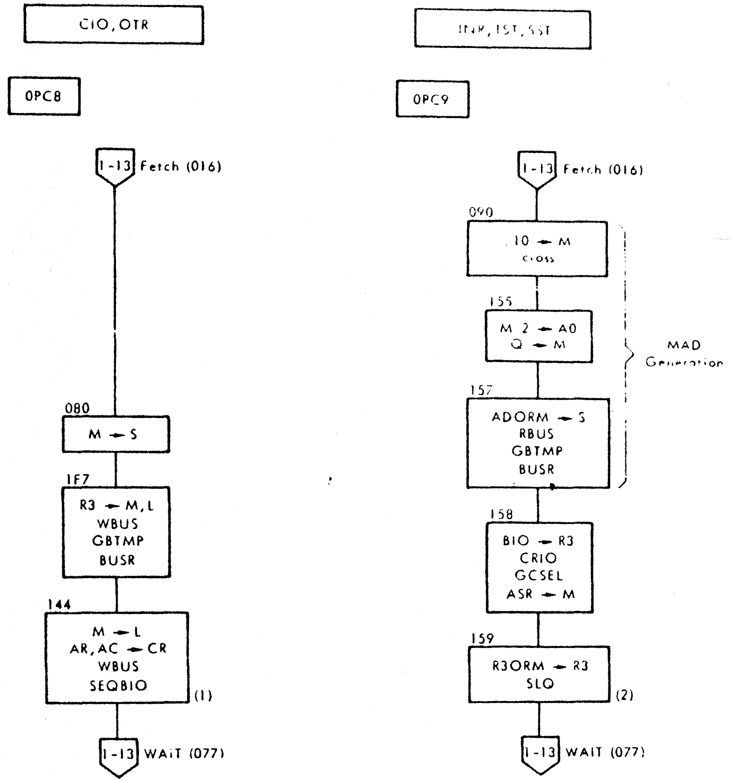
(CT) = SP address
P = Loop Counter
S = Memory Address Counter

(1) Saving of R2 in the case of a cut-off of MLi, Toto, Aj with j < i after loading of Ak with j < k < i by a page fault.

(2) P has already been incremented in RT2C.

(3) GBMMU does MMU ← BIO if K15 = 1 i.e. MMUYBIO if K15 = 0.

Figure 1-23 OPC 7 Multiple Load/Store, Table Load/Store (ML/MS, TL/TS)



(1) - M ← L and WBUS maintain BIO for 120 ns af·er ending edge of TMR. In the same cycle SEQBIO inhibits the BSYCPU resetting on T6.
- WBUS sets WRITE to "one"; so in INR, SST, T·T, WRITE = 0 and in CIO, OTR, WRITE = 1.

(2) SLQ for MUQ1 = 1 permits clock echo if FACIN·

Figure 1-24 OPC 8, 9 I/O Instructions

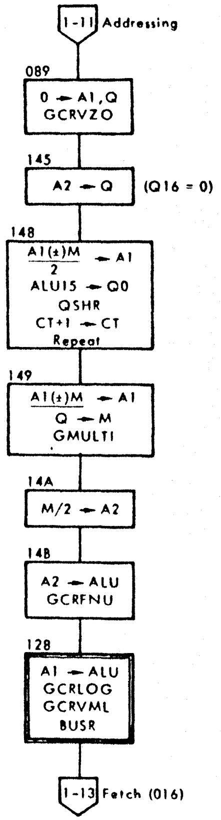
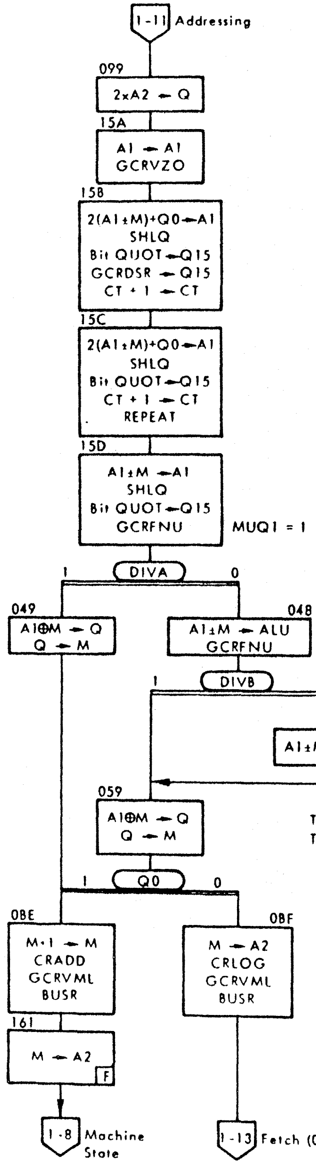


Figure 1-25 OPC 8 Multiplication (MU)



Bit 0 of dividend second word is thrown out

OVF is reset.
Dividend sign is stored.

GCRDSR sets OVF if division is impossible.

REMAINDER CORRECTION

O.K. if A ≥ 0 and ALU ≥ 0
or A < 0 and ALU = 0

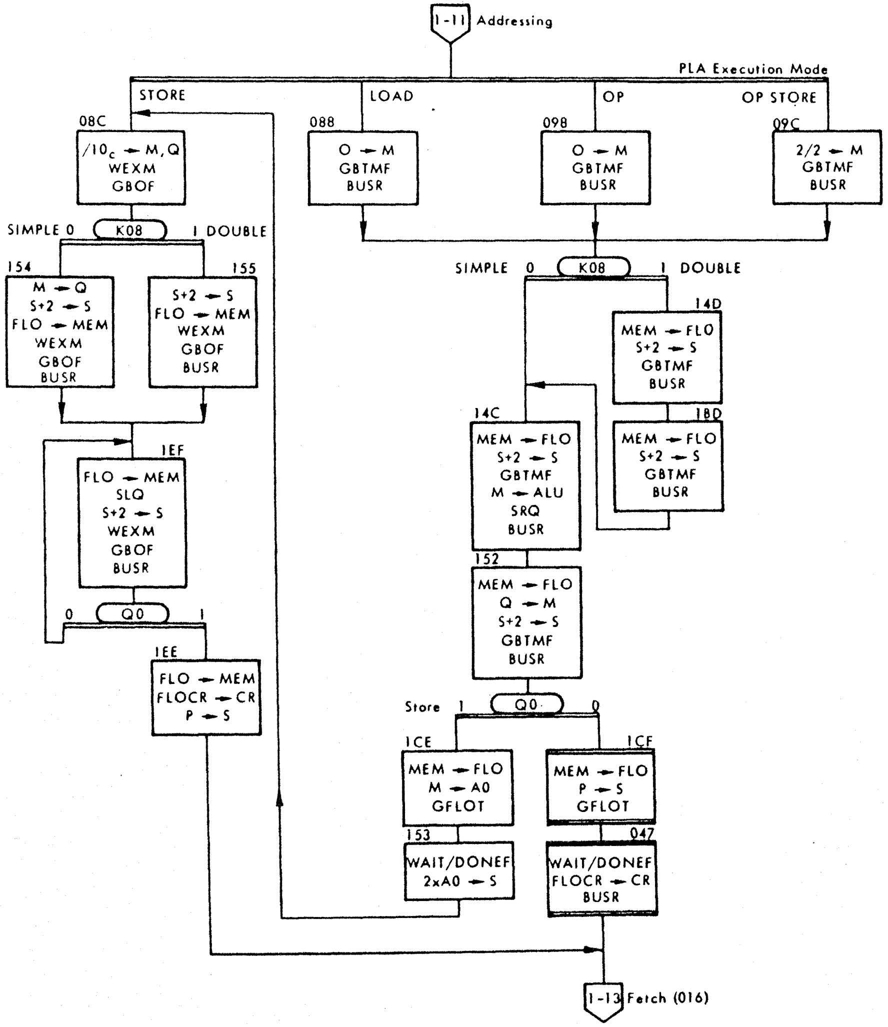
Clock Q16 inhibited

O.K. if A 0 and - M ALU 0
Quotient = QUOT + $ALU_{00} \oplus M_{00}$

Test DIVA = SFD/FSIG/+SFD FNU MUQ1 = 1
Test DIVB = SFD.FNV/+FSIG MUQ1 = 0

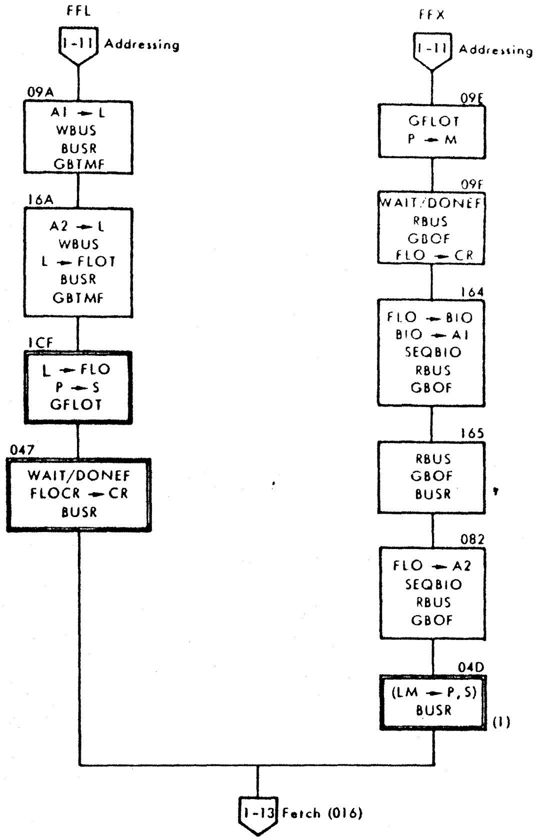
QUOTIENT CORRECTION

1-26 OPC 9 Division (DV)



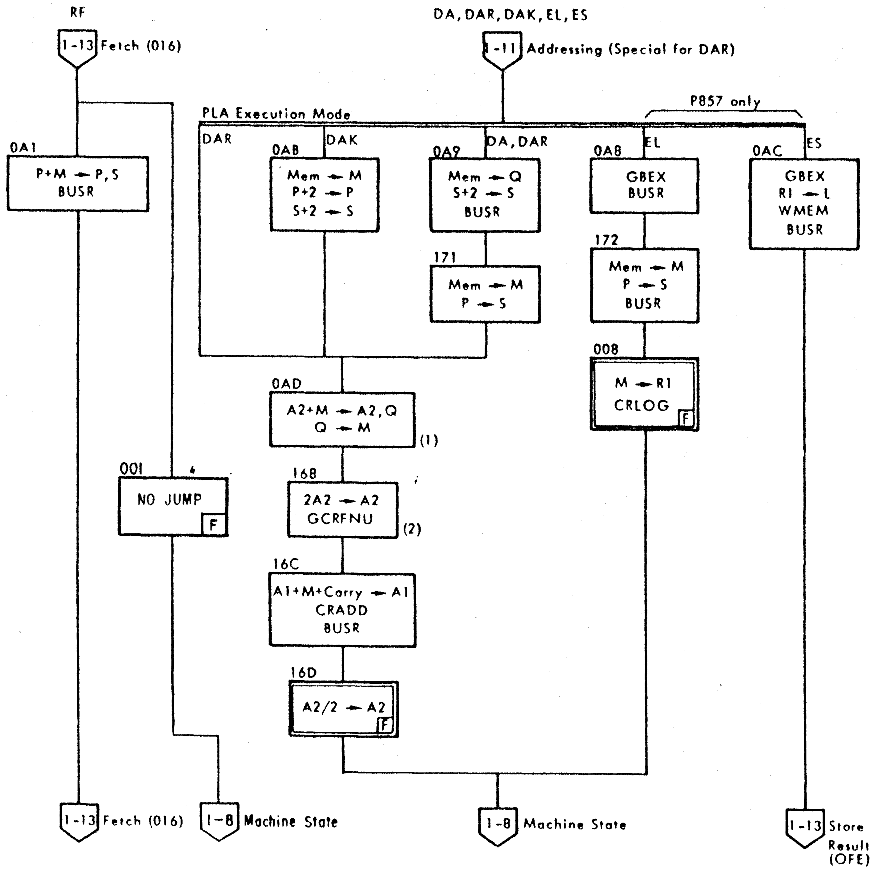
After the addressing routines, the effective address of the operant is in Q register (see routines type "C")
(1) contents of Q are now: | S | AD |
S = Store flag
AD = Address (right shifted)

Figure 1-27 OPC 8, 9 Floating Point:OP, OP Store, Load,Store (FA, FS, FM, FD, FLD, FST) P857 only



(1) This location is necessary to reset BSYCPU - M → P, S needs the transfer P → M in 09E

Figure 1-28 OPC 9 Floating Point Convertions (FFL, FFX), P857 only



(1) Carry in Q00

(2) FNU correctly positioned on 15 bits of the second word.

Figure 1-29 OPC10 (RF, DA, DAR, DAK, EL, ES)

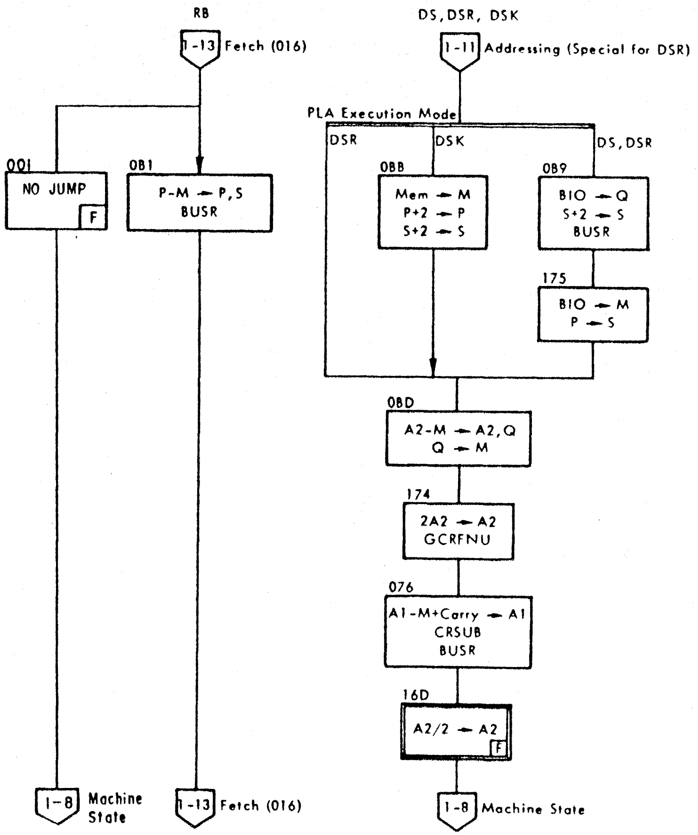


Figure 1-30 OPC 11 (RB, DS)

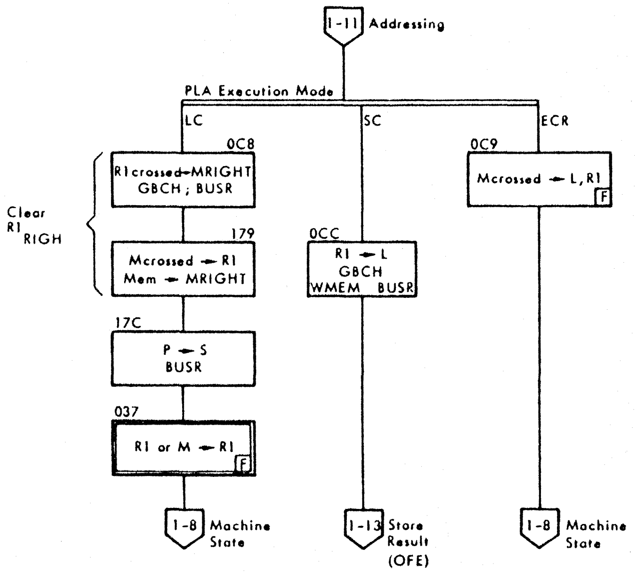


Figure 1-31 OPC 12 (LC, SC, FCR)

1/9/75

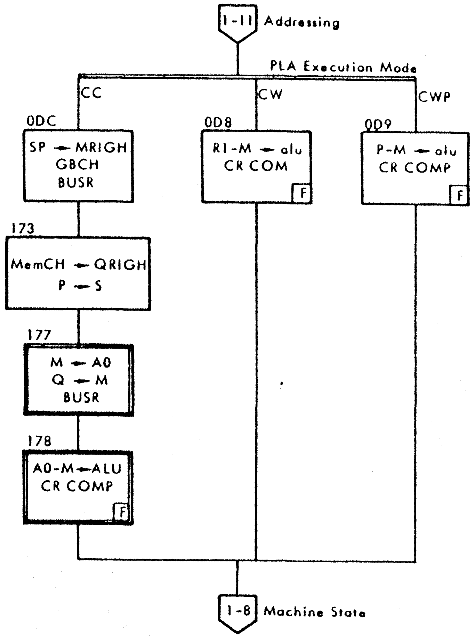


Figure 1-32 OPC 13 (CC, CW)

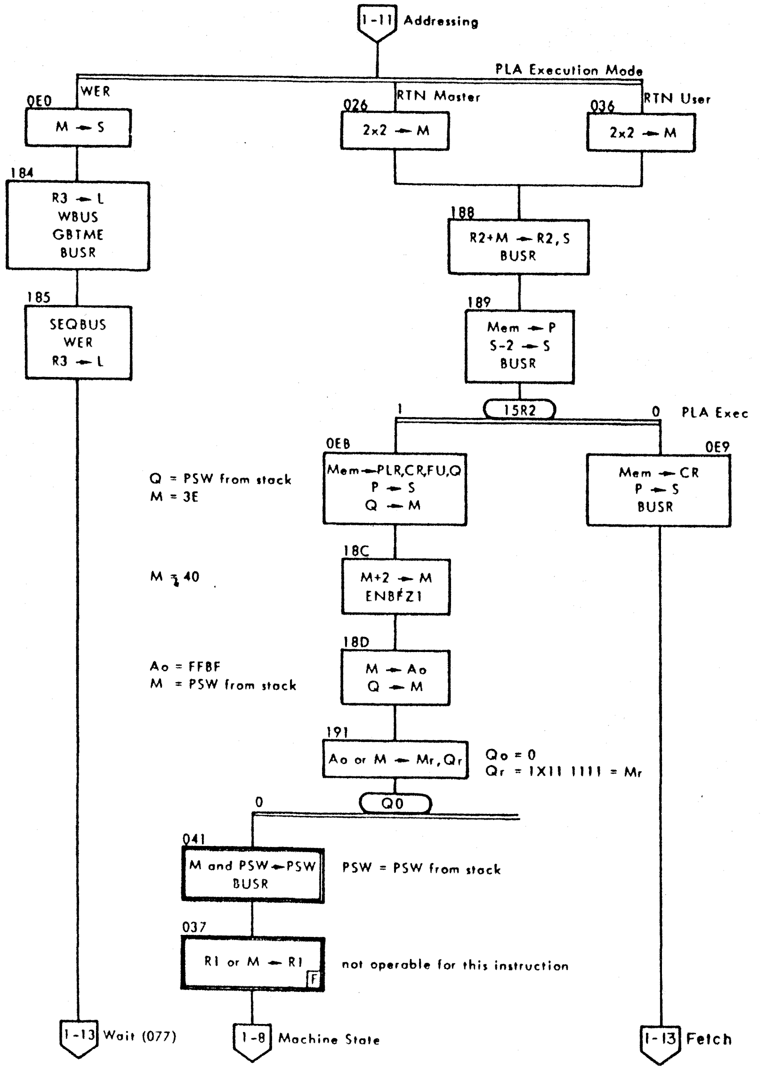
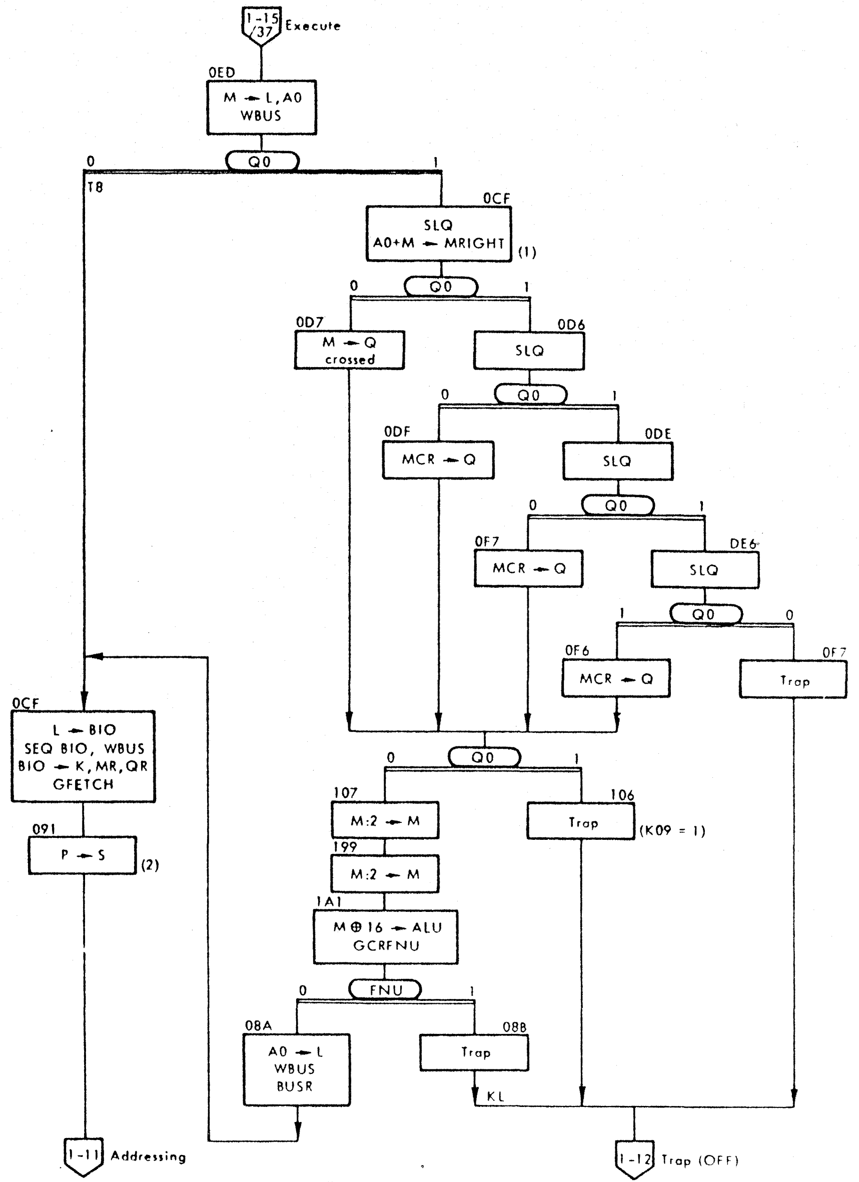


Figure 1-33 OPC 14 WER and RTN



(1) Contents of M are | 0 | k9 | k15 | 0 |

(2) This position permits the BSYZO which had been inhibited in OCF by SERBIO so that the BIO's are not destroyed on T6. S is loaded because addressing routines type "S" (without P → S) were used.

Figure 1-34 OPC 14 Execute (EX)

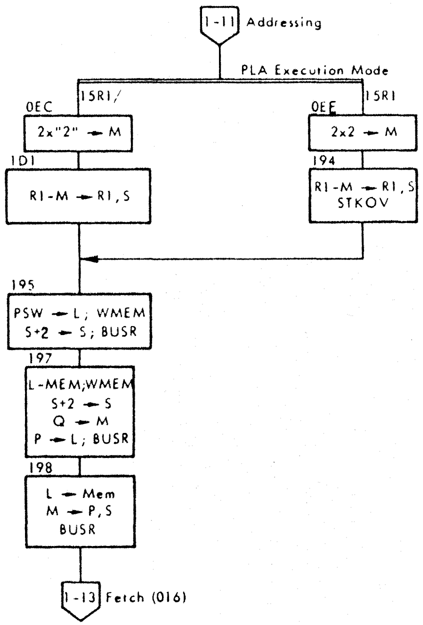
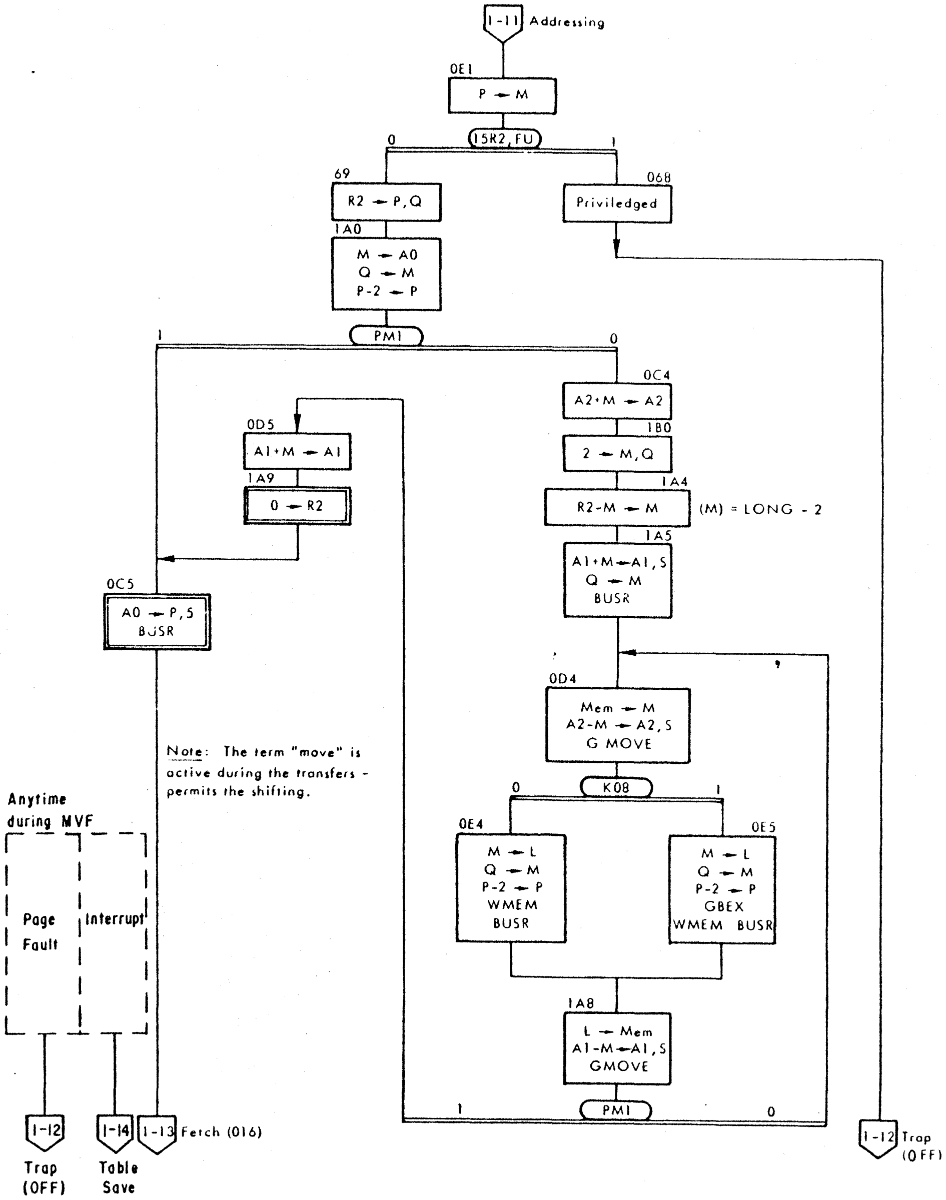


Figure 1-35 OPC 14 Call Function (CF)



Note: The term "move" is active during the transfers - permits the shifting.

Figure 1-36 OPC 14 MVF and MVSU (P857 only)

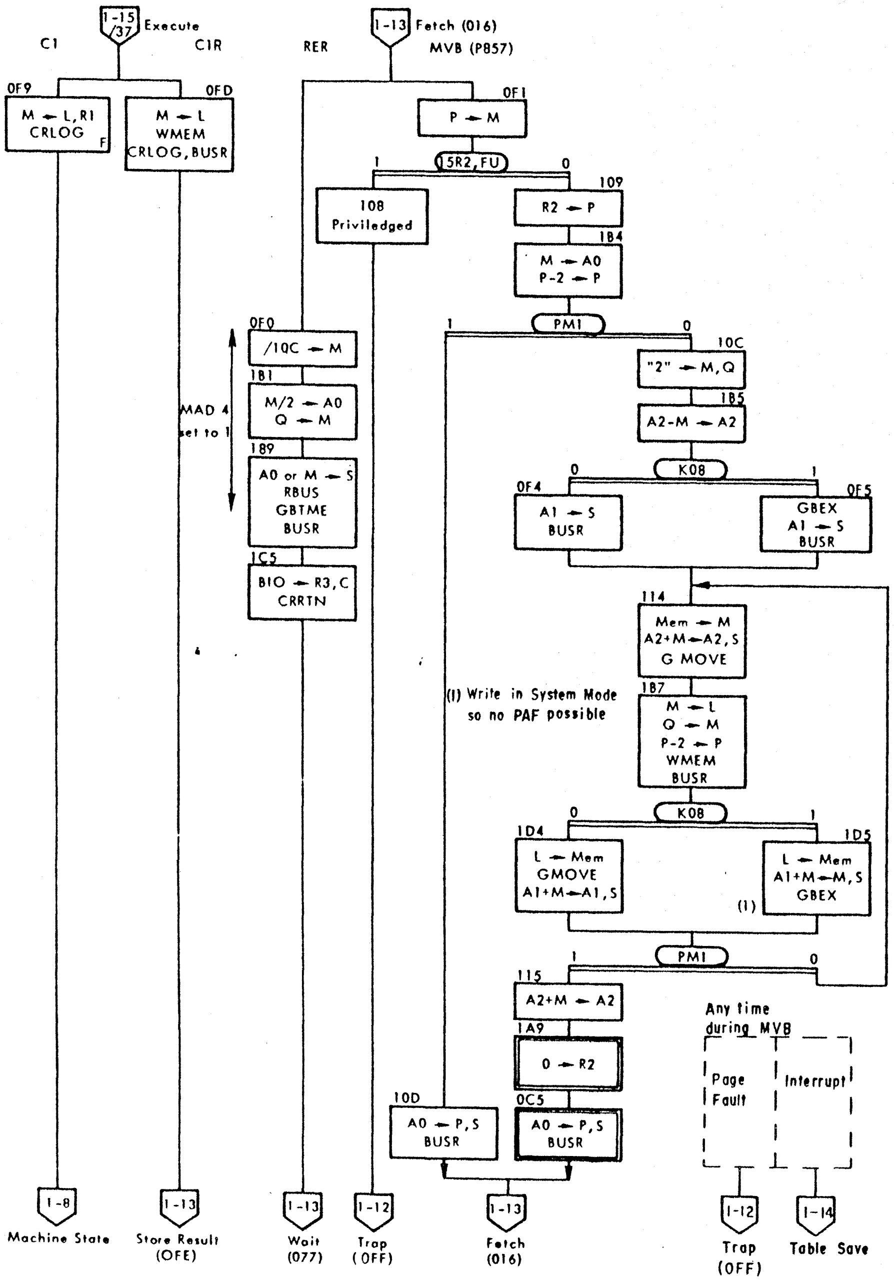


Figure 1-37 OPC 15 (CI, RER, MVB, MVUS)