Contents

Main Storage Processor

The combination of the control processor and associated control storage, the I/O interface, and the main storage processor and associated main storage makes up the System/34 processing unit. In System/34, the control processor controls the main storage processor.

The main storage processor is contained on five to eleven storage and processor logic cards. The number of cards is specified by the amount of main storage. Three cards are used for main storage processor and storage logic, and the other two to eight cards contain storage as follows:

- Five cards for 32K bytes of storage and processor logic
- Six cards for 48K bytes of storage and processor logic
- Seven cards for 64K bytes of storage and processor logic
- Nine cards for 96K bytes of storage and processor logic
- Eleven cards for 128K bytes of storage and processor logic

DATA FLOW AND CLOCK

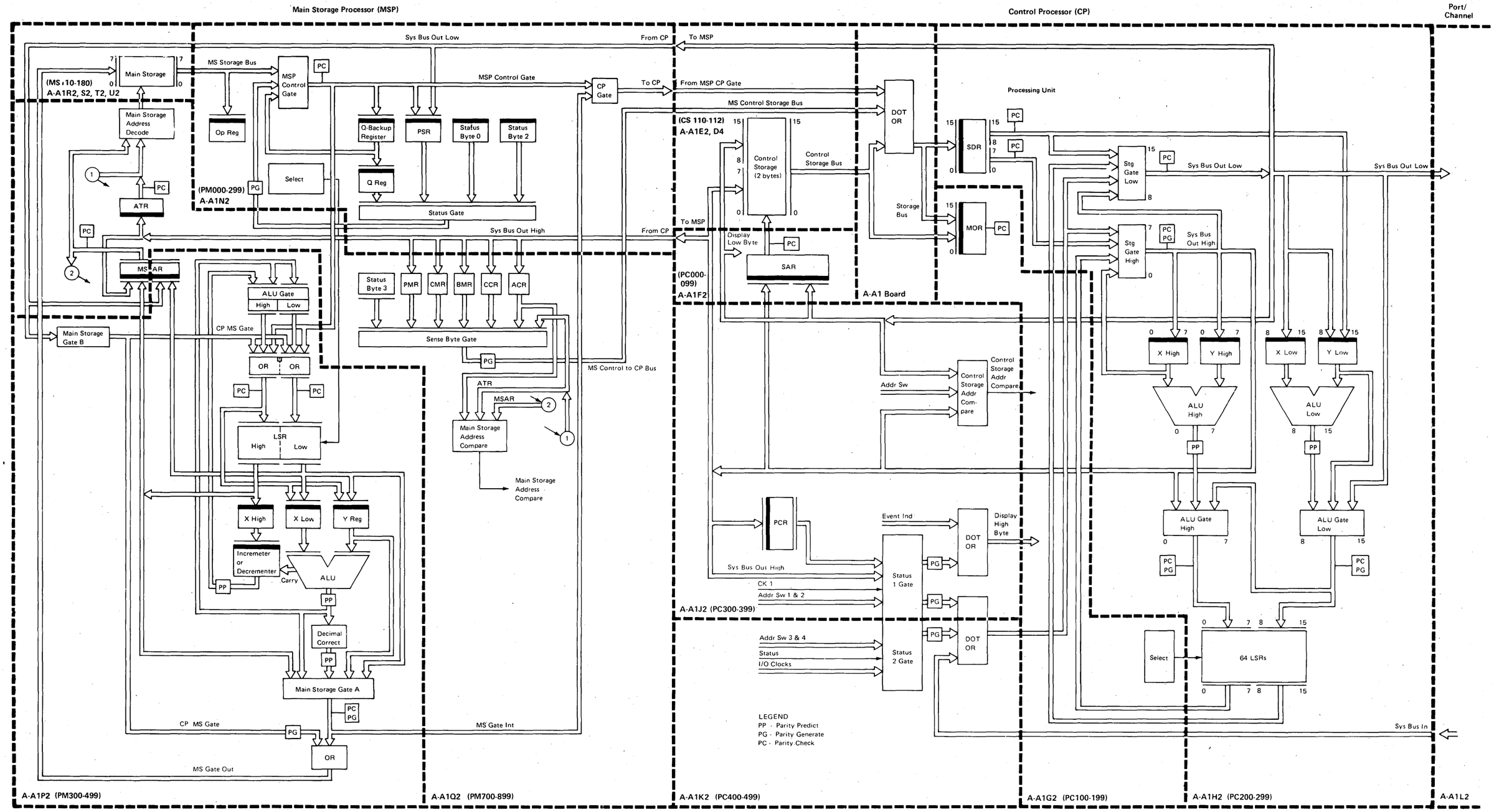
Data Flow

Data flows serially through the main storage processor in 8-bit bytes (plus 1 parity bit), through an arithmetic and logic unit, and is distributed to the remaining functional units of the main storage processor.

Parity Checking and Generation

The main storage processor checks for missing or extra bits during data movement by checking for an odd number of bits after the move. The parity checking and parity generating points are shown in the data flow.

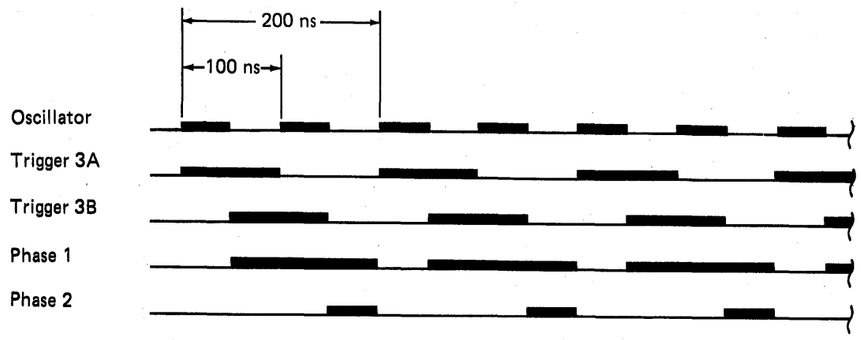
Parity predict circuits calculate the parity of the result of the arithmetic and logic unit operation. This calculated parity is compared against the parity generated. If there is a difference, a parity check occurs.



*Data flow bus lines may not pass through FRUs as shown

Clock

The main storage processor uses the free-running, 100-nanosecond oscillator (10 megahertz) from the control processor. The main storage processor generates its own clock times from this. The rise of the oscillator output causes trigger 3A to change condition, while the fall of the oscillator output causes trigger 3B to change condition. Triggers 3A and 3B generate phase 1 and phase 2 signals. Phase 1 is a 150-nanosecond signal that sets latches during the time the data is valid. Phase 2 is a 50-nanosecond signal that generates the local storage register write pulses.

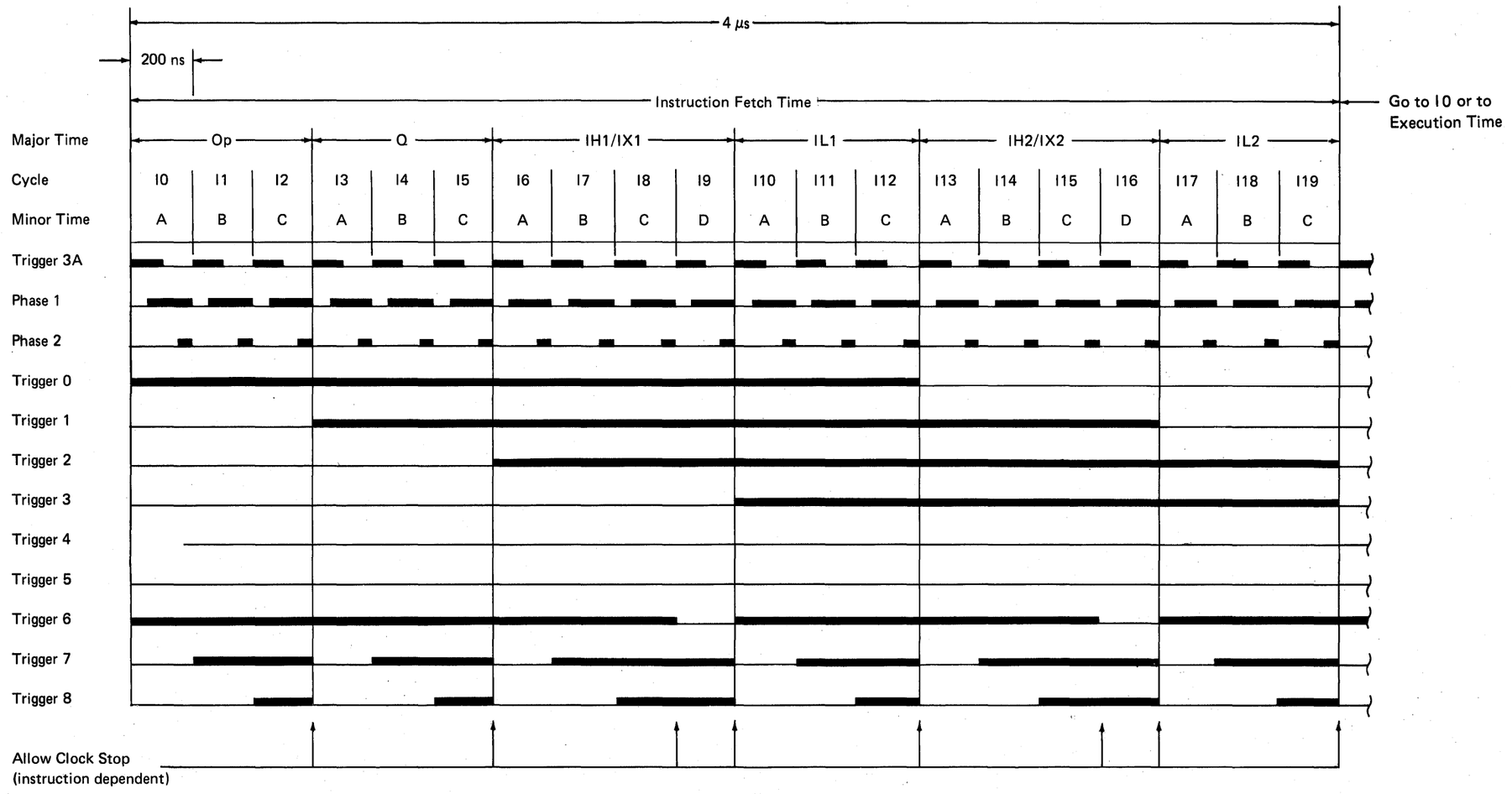


Each system instruction is divided into instruction fetch time and instruction execution time.

The instruction fetch time is divided into six major times:

- Op time
- Q time
- IH1/IX1 time
- IL1 time
- IH2/IX2 time
- IL2 time

These times are twenty 200-nanosecond instruction cycles (I-cycles) that are numbered from I0 through I19. These I-cycles fetch the instruction bytes. Three cycles are needed to fetch each byte from main storage. If indexing is needed, an additional cycle is taken so that the displacement byte can be added to the index register and the real address can be placed in the operation local storage register location.



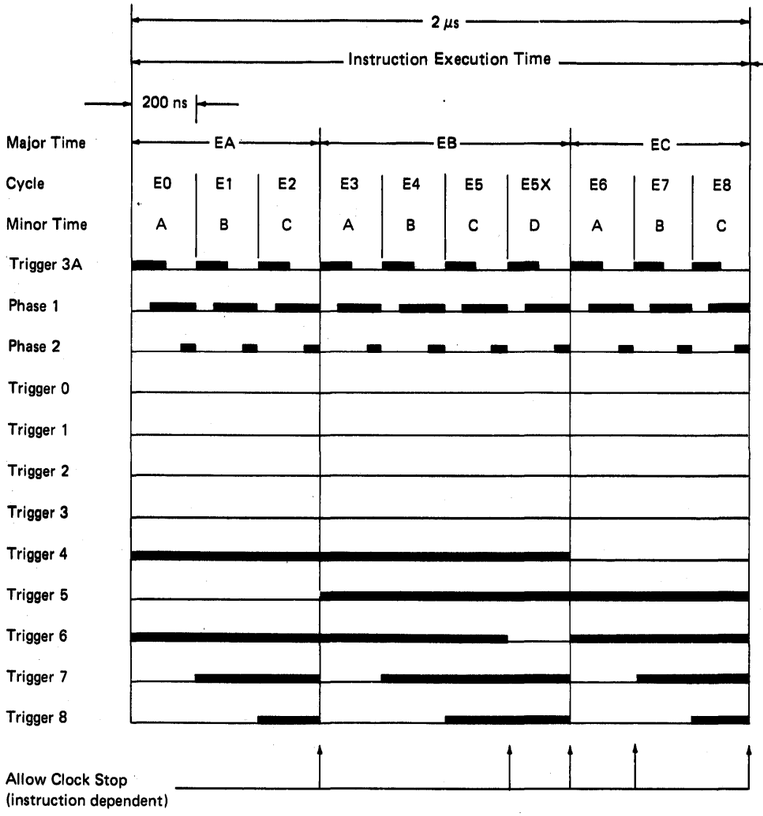
The instruction execution time is divided into three major times: EA, EB, and EC. These times are ten 200-nanosecond execution cycles (E-cycles): three E-cycles for EA time, four E-cycles for EB time, and three E-cycles for EC time. During execution time, the operands are fetched from main storage and operated on as indicated by the instruction being executed.

Nine triggers are needed to divide the instruction fetch and instruction execution times into the needed cycles. The timing charts show how the cycle time and trigger 3A and trigger 3B work together. The output frequency from trigger 3A is one-half the oscillator frequency.

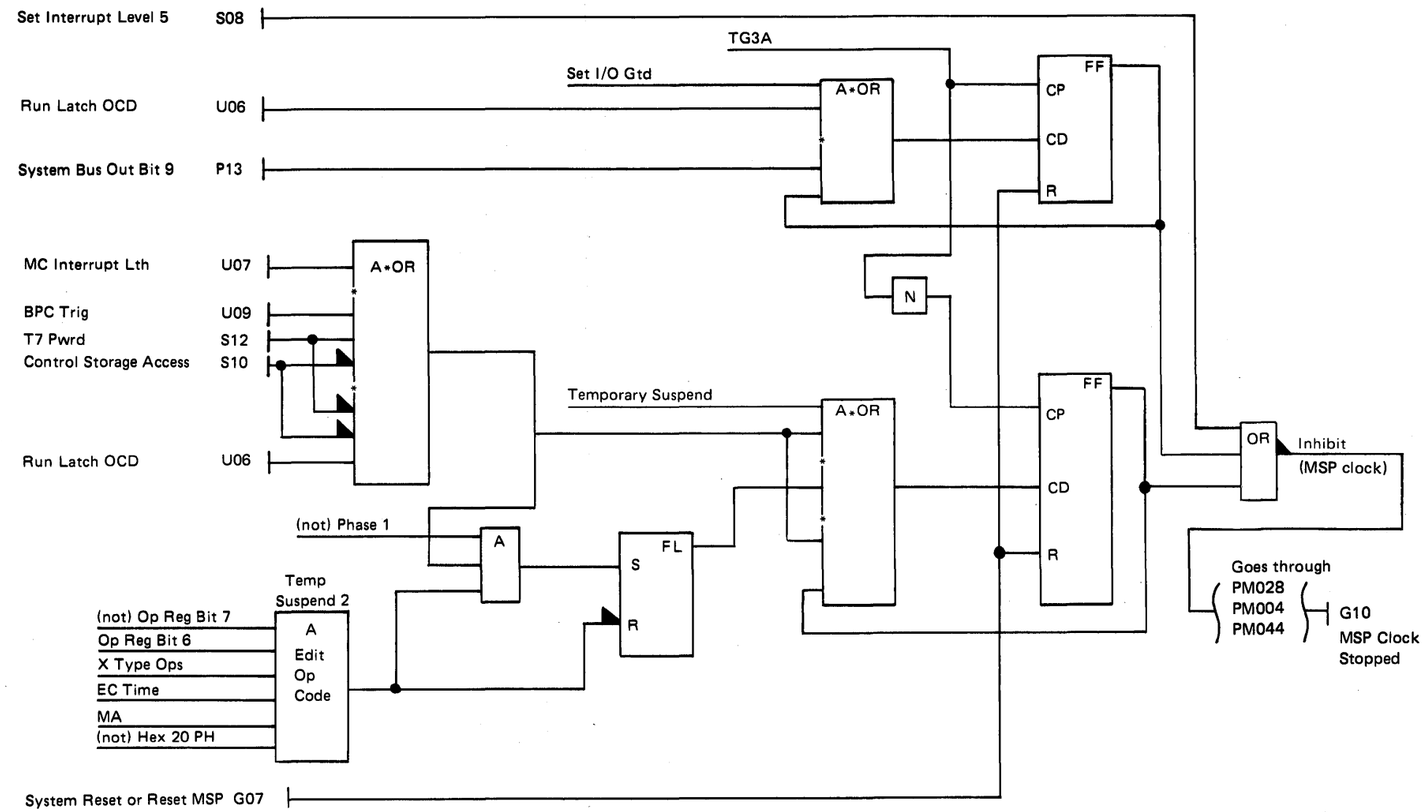
While an instruction is being executed, the main storage processor clock logic sets the clock to the needed cycle time. This permits skipping cycles or returning to a cycle in the major time. After instruction execution is complete, the clock logic is set to I0 time.

Because the main storage processor clock can be stopped and started by the control processor, the clock inhibit logic permits stopping at specific points in the instruction fetch or instruction execution times. The times at which the clock can be stopped are determined by the instruction being fetched or executed.

The main storage processor clock can be stopped only at specific times because the control processor (1) must store the contents of the main storage processor registers while the main storage address register (MSAR) is being used by the control processor instruction, and (2) must return the contents before starting the main storage processor clock. The main storage processor stops only when a change in the MSAR can occur without affecting the system instruction execution.



MSP Clock Control/Stop MSP Clock
MSP Control Card A-A1N2



The following conditions bring up inhibit: PM220

- End of MSP instruction (temp. suspend) and CP stopped
- MC interrupt and temporary suspend
- Main storage or MSP registers being accessed by the CP

OPERATIONS

Instruction and Execution Cycles

The two types of machine cycles for the internal operation of the main storage processor are: instruction cycles (I-cycles) and execution cycles (E-cycles).

Instruction cycles read instruction bytes from main storage, and execution cycles execute the instruction.

Instruction cycles move the instruction bytes from main storage to the various registers needed to execute the instruction. If the instruction does not need any operands from main storage, the operation is completed without execution cycles. Instructions that do not need execution cycles are:

- Branch
- Jump
- Supervisor call
- Load address

Most operations need data from one or two main storage fields. Main storage holds the operands needed for working with these data fields. Execution cycles process the data fields.

Sequential Instruction Execution

The main storage processor works step by step. Because of this, the instructions are placed in increasing main storage locations. Instruction sequence is maintained by keeping the address of the storage location in the instruction address register. The instruction address register is increased by 1 as each instruction byte is read from storage so that the next higher storage location can be addressed. This process continues until all the instruction bytes have been addressed. The instruction is then executed. After the instruction has been completed, the instruction address register addresses the next instruction from storage.

Branching

Branching permits the main storage processor to change the instruction sequence under specific conditions. Branching also permits changing the sequence of user program instructions. If the branch condition is met, the main storage processor places the address of the branch-to location in the instruction address register, which now becomes the location of the next branch instruction to be executed. By branching to a different storage location and skipping specific instructions, the sequence of the stored program is changed.

I/O Data Movement

The user program requests I/O data movement with a supervisor call (SVC) instruction. The SVC instruction sets interrupt level 5, if interrupt level 5 is enabled. When the control processor senses an SVC instruction from the main storage processor, the control processor determines which operation is requested by analyzing the constants stored in the main storage processor as a result of executing the SVC instruction. The control processor then controls all data movement between the main storage processor and the I/O devices.

Instruction Formats

The main storage processor performs three types of instructions:

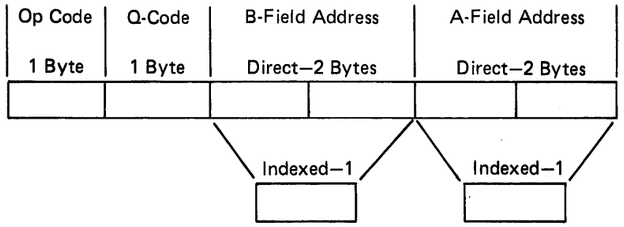
- Two-address instructions
- One-address instructions
- Command instructions

Two-address instructions have two separate fields in main storage and, therefore, contain two addresses. Most one-address instructions have only one field in main storage and, therefore, contain only one address (the load address instruction contains the needed data instead of an address). Command instructions do not need main storage data fields at all and do not contain addresses.

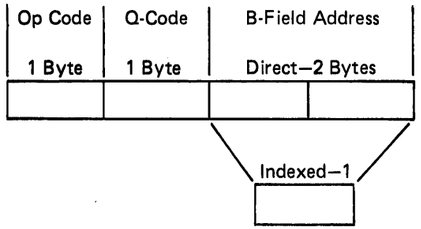
Each instruction has an operation code and a Q-code. These codes are followed either by a control code or by one or two addresses. The length of the instruction is from 3 bytes to 6 bytes, as determined by the type of instruction and the type of addressing.

The first half-byte (bits 0-3) of the operation code determines the format of the instruction (one-address, two-address, and so on) and the method of addressing used. If all 4 bits are set to 1, the instruction is a command instruction. The bits are broken into two groups (bits 0-1 and bits 2-3). If both bits in either group are set to 1, the instruction is a one-address instruction; if neither group has both bits set to 1, the instruction is a two-address instruction. If only 1 bit is set to 1 in either of the groups in a two-address instruction, the address is indexed. The following instruction format chart shows the operation code bits and the number of bytes in the associated address.

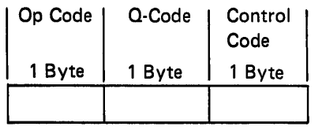
The second half-byte (bits 4-7) of the operation code determines the operation. Use of the Q-code and the control code is controlled by the operation requested. The complete main storage instruction set is shown below under *Instruction List*.



Two-Address Instruction



One-Address Instruction



Command Instruction

Instruction Format	Instruction Type[1]	Op Code (Bits 0-3)	First Operand Address	Second Operand Address
Two-Address	X	0 0 0 0	2 bytes direct	2 bytes direct
		0 0 0 1	2 bytes direct	Indexed by XR1
		0 0 1 0	2 bytes direct	Indexed by XR2
		0 1 0 0	Indexed by XR1	2 bytes direct
		0 1 0 1	Indexed by XR1	Indexed by XR1
		0 1 1 0	Indexed by XR1	Indexed by XR2
		1 0 0 0	Indexed by XR2	2 bytes direct
		1 0 0 1	Indexed by XR2	Indexed by XR1
		1 0 1 0	Indexed by XR2	Indexed by XR2
One-Address (nonbranch)	Y	0 0 1 1	2 bytes direct	
		0 1 1 1	Indexed by XR1	
		1 0 1 1	Indexed by XR2	
One-Address (branch)	Z	1 1 0 0	2 bytes direct	
		1 1 0 1	Indexed by XR1	
		1 1 1 0	Indexed by XR2	
Command	F	1 1 1 1		

[1]See FSL page PM082.

Instruction List

Operation Type	Instruction Type[1]	Mnemonic	Operation	Q-Code Use
Two-Address	X	ZAZ	Zero and add zoned	Field length
	X	AZ	Add zoned decimal	
	X	SZ	Subtract zoned decimal	
	X	MVC	Move characters	
	X	ALC	Add logical characters	
	X	SLC	Subtract logical characters	
	X	CLC	Compare logical characters	
	X	ED	Edit	
	X	ITC	Insert and test characters	
	X	MVX	Move hexadecimal character	Select a half byte
One-Address	Y	MVI	Move logical immediate	Immediate data
	Y	CLI	Compare logical immediate	
	Y	SBN	Set bits on masked	Bit selection
	Y	SBF	Set bits off masked	
	Y	TBN	Test bits on masked	
	Y	TBF	Test bits off masked	
	Y	ST	Store register	Register selection
	Y	L	Load register	
	Y	A	Add to register	
	Z	LA	Load address	
	Z	BC	Branch on condition	Branch condition
Command	F	SVC	Supervisor call	Part of system support program product interface
	F	LPMR	Load program mode register	Main storage processor register selection
	F	JC	Jump on condition	Branch condition

Legend for Instruction Type:

F = Command instruction

X = Two-address instruction (can be indexed by bits 0-3)

Y = One-address instruction (can be indexed by bits 0 and 1)

Z = One-address instruction (can be indexed by bits 2 and 3)

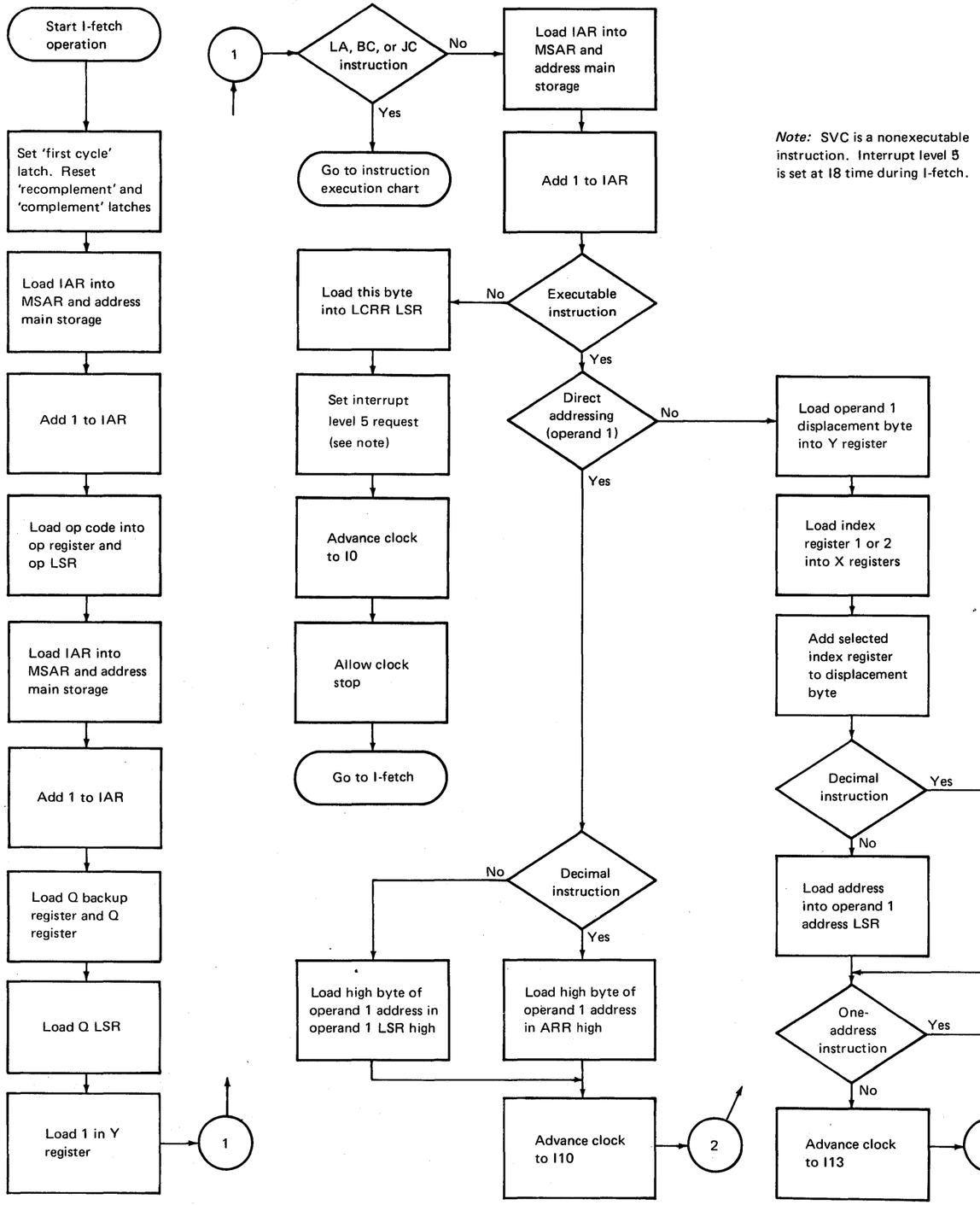
[1]See FSL page PM082.

Instruction Fetch Operation

The instruction address register contains the address of the leftmost byte of the instruction. This byte specifies the operation to be performed. During the instruction fetch, the operation code byte is loaded in the operation register. The output of the operation register is decoded to determine the instruction and the type of addressing, if needed. One is added to the instruction address register. The second byte loaded during instruction fetch is the Q-byte. The Q-byte is loaded in the Q-backup register, the Q-register, and the Q-byte of the Op-Q local storage register.

Again, 1 is added to the instruction address register. The third byte read from main storage is a displacement byte, the first byte of the operand 1 address, or a control code. Note that 1 is added to the instruction address register after each storage cycle. The operations that follow the third storage cycle are controlled by the instruction and the type of addressing specified. Address calculations for indexed addresses are done during I-fetch and are stored in the main storage processor local storage registers for use during the execution cycles. However, some instructions do not need execution cycles. In such cases, the main storage processor clock is set to time I0 and the next I-fetch cycle starts the next instruction. The I-fetch cycle:

- Loads the instruction op-code byte in the operation register and decodes the instruction
- Loads the Q-byte in the Q-backup register, Q-register, and Q-byte of the Op-Q local storage register
- Calculates and stores the addresses in the operand 1 and operand 2 local storage registers
- Loads the R-byte into the length count recall register (LCRR) local storage register stack for a nonexecutable instruction
- Calculates and stores addresses in the instruction address register and address recall register for branch-on-condition and jump-on-condition instructions.



Note: SVC is a nonexecutable instruction. Interrupt level 5 is set at I8 time during I-fetch.

Addressing

The main storage processor selects one of two types of addressing when executing instructions: direct addressing or indexed addressing.

Most addresses given in the instruction are for the location of the low-order (rightmost) byte of the field. Therefore, as the instruction is executed, the operand address local storage register is decreased to lower the main storage address. An exception is the insert-and-test-characters instruction, which is executed from the high-order byte to the low-order byte. In this case, the operand address local storage register is increased in the same way that the instruction address register is increased during instruction fetch cycles.

Direct Addressing

Direct addressing needs a 2-byte address for each field selected by the instruction. The first address that follows the Q-byte in the instruction is the address of the result field or the first operand. In an instruction with two addresses, the second address is the source field (second operand) and the first operand field is used as both a source field and a result field. The first operand source field is changed during the instruction execution cycles. The second operand is not changed except when the two operands overlap.

Indexing

Indexing gives the user a method for changing addresses in a program without changing the instruction. An indexed address is a single byte in the instruction (third instruction byte). This single byte (displacement byte) is added to the contents of a 2-byte index register to form the operand address. This operand address is stored in the operand address local storage register.

Indexing is used to:

- Perform an instruction with an indexed address
- Add the index register to a constant
- Branch to an address to execute the instruction at a different storage location
- Perform an instruction or a series of instructions many times without using too many storage locations

Either of two index registers (XR1 or XR2) can be selected for indexing. The recognition of an indexed address and the selection of index registers are described under *Instruction Formats* earlier in this section.

Address Translation

Address translation must be used to access any real address in main storage from 64K through 128K.

A user program, which is link-edited to load at one address, permits a different main storage address greater than 64K to be selected. The system operator does not have to keep track of which blocks of storage are available for program execution. For example, the system operator may need to execute a program that is link-edited for a 2K area of real addresses between hexadecimal 2000 and hexadecimal 27FF. Without address translation, the system operator cannot load the program until the specified 2K area is available. With address translation, the System Support Program Product updates the ATR with a 2K address block and issues a supervisor call instruction. The system then moves the program to the selected 2K area of main storage. The program is then executed using address translation, as if the program were in the specified 2K area. With address translation, the addresses specified by a user program become logical addresses and not real addresses.

During instruction and execution cycles, addresses are loaded into the main storage address register. The 11 low-order bits (5-7, 8-15) of the main storage address register contain an address inside a 2K area of storage. The 5 high-order bits (0-4) of the main storage address register select the output from the sixty-four 1-byte address translation registers. Note that 32 registers are used for main storage processor address translation and 32 registers are used for I/O address translation. The contents of the addressed address translation register are then sent to the main storage address decode logic. This decoded output is the real main storage address. This describes the operation for translate mode only. For nontranslate mode, the 5 high-order bits (0-4) of the main storage address register are used directly to obtain the real main storage address.

In the following address translation example, the logical addresses specified by the user program are between hexadecimal 2000 and hexadecimal 27FF. Assume that the only available 2K area of storage is between hexadecimal addresses 17800 and 17FFF. Therefore, the addresses must be translated. If an address of hexadecimal 27FF is the logical address, the real address is hexadecimal 17FFF.

In address translation, the main storage processor:

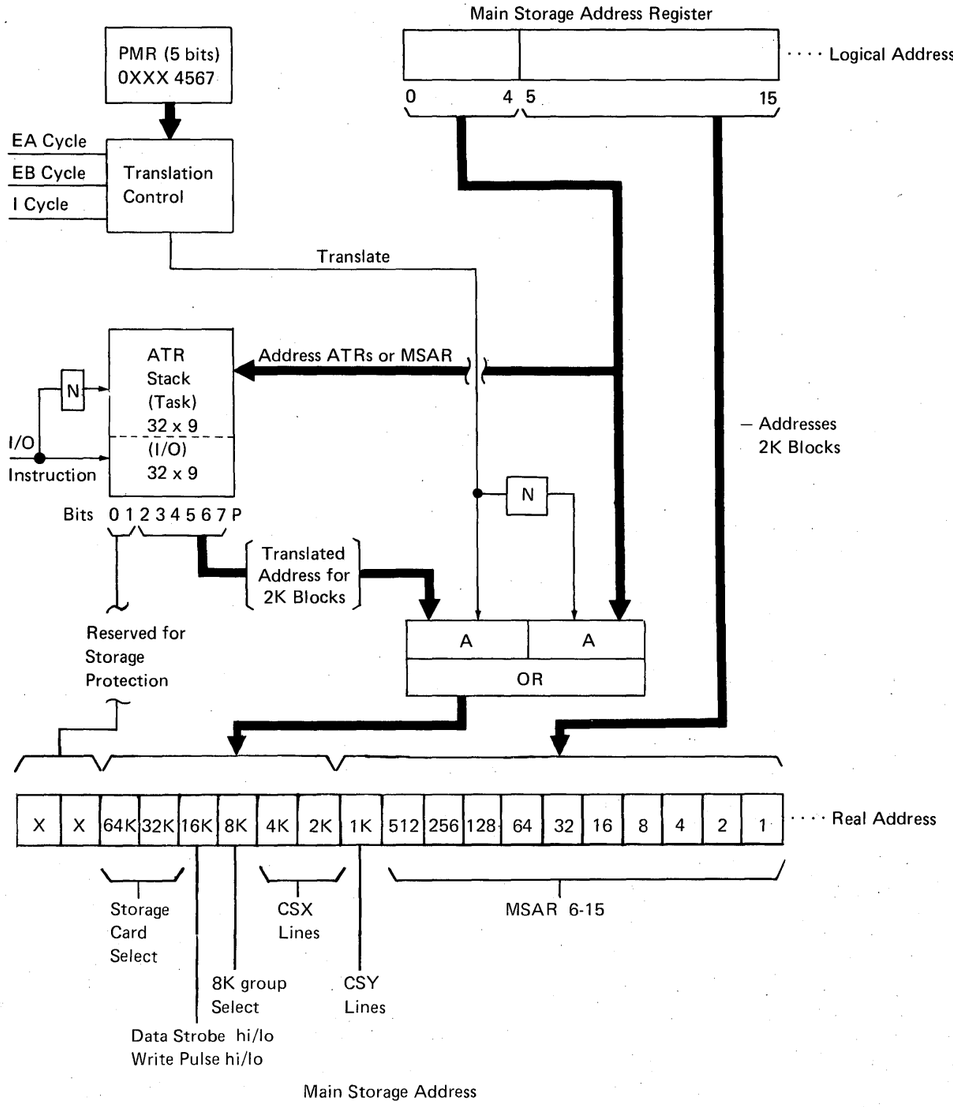
- Loads the logical address into the main storage address register (hexadecimal 27FF).
- Selects the output from the selected address translation register bits 0-7, as instructed by bits 0-4 of the main storage address register and the main storage processor I/O selection circuits.
- Moves the contents of the address translation register to the main storage address decode logic. Here, the output of the address translation register and the 11 bits from the main storage address register are combined to generate the real address.
- Generates the real address. The real address is hexadecimal 17FFF.

The address translation register is selected by the 5 high-order bits of the main storage address register. The address translation register must be loaded with the number of the 2K block of storage in which the program is loaded. In the above example, the program was loaded into the 47th 2K area of storage; therefore, hexadecimal 2F must be loaded into address translation register 4.

The following table shows the addresses in 2K areas for 32K bytes of storage. The hexadecimal address specified in any given ATR area number may be any 2K hexadecimal block (00-3F) and need not be in sequence.

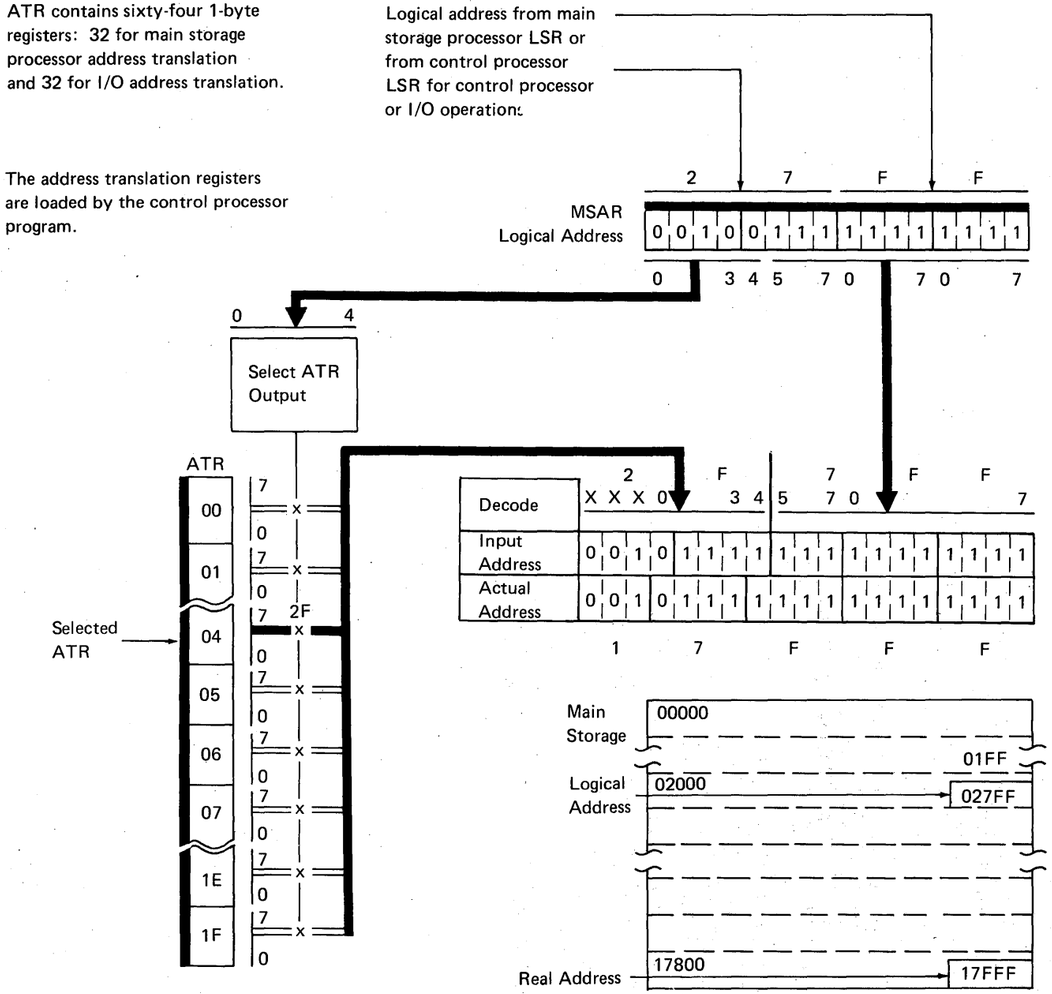
Area Number (Hex)	Starting Address (Hex)	Ending Address (Hex)	Starting Address (Decimal)	Ending Address (Decimal)
00	0000	07FF	0000	2047
01	0800	0FFF	2048	4095
02	1000	17FF	4096	6143
03	1800	1FFF	6144	8191
04	2000	27FF	8192	10239
05	2800	2FFF	10240	12287
06	3000	37FF	12288	14335
07	3800	3FFF	14336	16383
08	4000	47FF	16384	18431
09	4800	4FFF	18432	20479
0A	5000	57FF	20480	22527
0B	5800	5FFF	22528	24575
0C	6000	67FF	24576	26623
0D	6800	6FFF	26624	28671
0E	7000	77FF	28672	30719
0F	7800	7FFF	30720	32767

ATR Logical Circuit



ATR contains sixty-four 1-byte registers: 32 for main storage processor address translation and 32 for I/O address translation.

The address translation registers are loaded by the control processor program.



INSTRUCTION EXECUTION

This section describes how each System/34 main storage instruction is executed. The method of describing how instructions are usually executed is:

1. Show the instruction format and operation codes.
2. Describe what the instruction does.
3. Show the program status byte settings.
4. Use a flowchart to show how the instruction is executed.
5. Use a timing chart to show the details of instruction execution.

The flowcharts and timing charts show the associated operation and machine timing.

Data flow and logical functions are controlled by selecting the proper gating lines in the main storage processor. These functions are shown on this page in appropriate charts.

Instructions for the main storage processor are divided into four groups:

- Arithmetic instructions
- Data control instructions
- Logical instructions
- Command instructions

CP Gate Selection

Bits 01	Lines Gated Through
00	No Input/Output Selected
01	MSP Ctrl Gt (8-15, P)
10	MS Gt Int (8-15,P)
11	MSP Ctrl Gt ANDed with MS Gt Int

+CP Gt Sel Bit 1 A1-P2J10
+CP Gt Sel Bit 0 A1-P2J13

ALU Gate Selection

Bits 01	ALU Gt Hi	ALU Gt Lo
00	Degate	Degate
01	ALU Lo	Degate
10	ALU Hi	ALU Lo
11	ALU Hi/ALU Lo	ALU Lo

+ALU Gt Sel Bit 1 A1-P2M12
+ALU Gt Sel Bit 0 A1-P2M13

LSR Selection

Bits 0123	Register Name
0000 thru 0111	Reserved
1000	Operand 1 Address
1001	Operand 2 Address
1010	Instruction Address Register (IAR)
1011	Operation Register/Q-Register
1100	Index Register 1 (XR1)
1101	Index Register 2 (XR2)
1110	Address Recall Register (ARR)
1111	Length Count Recall Register (LCRR)

+LSR Sel Bit 3 A1-P2U10
+LSR Sel Bit 2 A1-P2S09
+LSR Sel Bit 1 A1-P2U09
+LSR Sel Bit 0 A1-P2S05

MSP X_L Y_L Selection

Bit 0	XL	YL
0	LSR Gt	LSR
1	LSR	LSR Gt

+MSP XL, YL Select Bit A1-P2M07

ALU Control Selection

Bits 012	ALU Function Lo	ALU Function Hi
000	X and Y	X minus 1 plus carry
001	X or Y	X plus carry
010	X or (not) Y	X minus 1 plus carry
011	X and (not) Y	X plus carry
100	X minus 1 plus carry	X minus 1 plus carry
101	X plus Y plus carry	X plus carry
110	X minus Y minus 1 plus carry	X minus 1 plus carry
111	X plus carry	X plus carry

+ALU Func Sel Bit 2 A1-P2P05
+ALU Func Sel Bit 1 A1-P2P10
+ALU Func Sel Bit 0 A1-P2G13

Main Storage Gate A Selection

Bits 0123	Selection
0000	Control Processor System Bus Out
0001	Y Register
0010	LSR High
0011	LSR Low
0100	Zone = F, Numeric = Decimal Correct
0101	Zone = D, Numeric = Decimal Correct
0110	Not Used
0111	ALU
1000	Zone = F, Numeric = Y (12-15)
1001	Zone = D, Numeric = Y (12-15)
1010	Not Used
1011	Not Used
1100	Zone = Y (8-11), Numeric = X (12-15)
1101	Zone = Y (12-15), Numeric = X (12-15)
1110	Zone = X (8-11), Numeric = Y (12-15)
1111	Zone = X (8-11), Numeric = Y (8-11)

+MS Gt Sel Bit 3 A1-P2G10
+MS Gt Sel Bit 2 A1-P2G09
+MS Gt Sel Bit 1 A1-P2M10
+MS Gt Sel Bit 0 A1-P2M11

Zero and Add Zoned (ZAZ)

This instruction moves data from operand 2, byte by byte starting with the rightmost byte, into the rightmost byte positions of operand 1. If operand 1 is longer than operand 2, the main storage processor fills the extra positions with high-order decimal zeros (hexadecimal F0).

The main storage processor sets the zone bits of all bytes, except the rightmost byte in operand 1, to hexadecimal F (binary 1111). The zone bits of the rightmost byte in operand 1 are set to:

- Hexadecimal F if the value moved is either zero or positive
- Hexadecimal D (binary 1101) if the value moved is negative

Program Status Byte Settings

Bit	Name	Condition Indicated
7	Equal	Zero result
6	Low	Negative result
5	High	Positive result
4	Decimal overflow	Bit not affected
3	Test false	Bit not affected
2	Binary overflow	Bit not affected

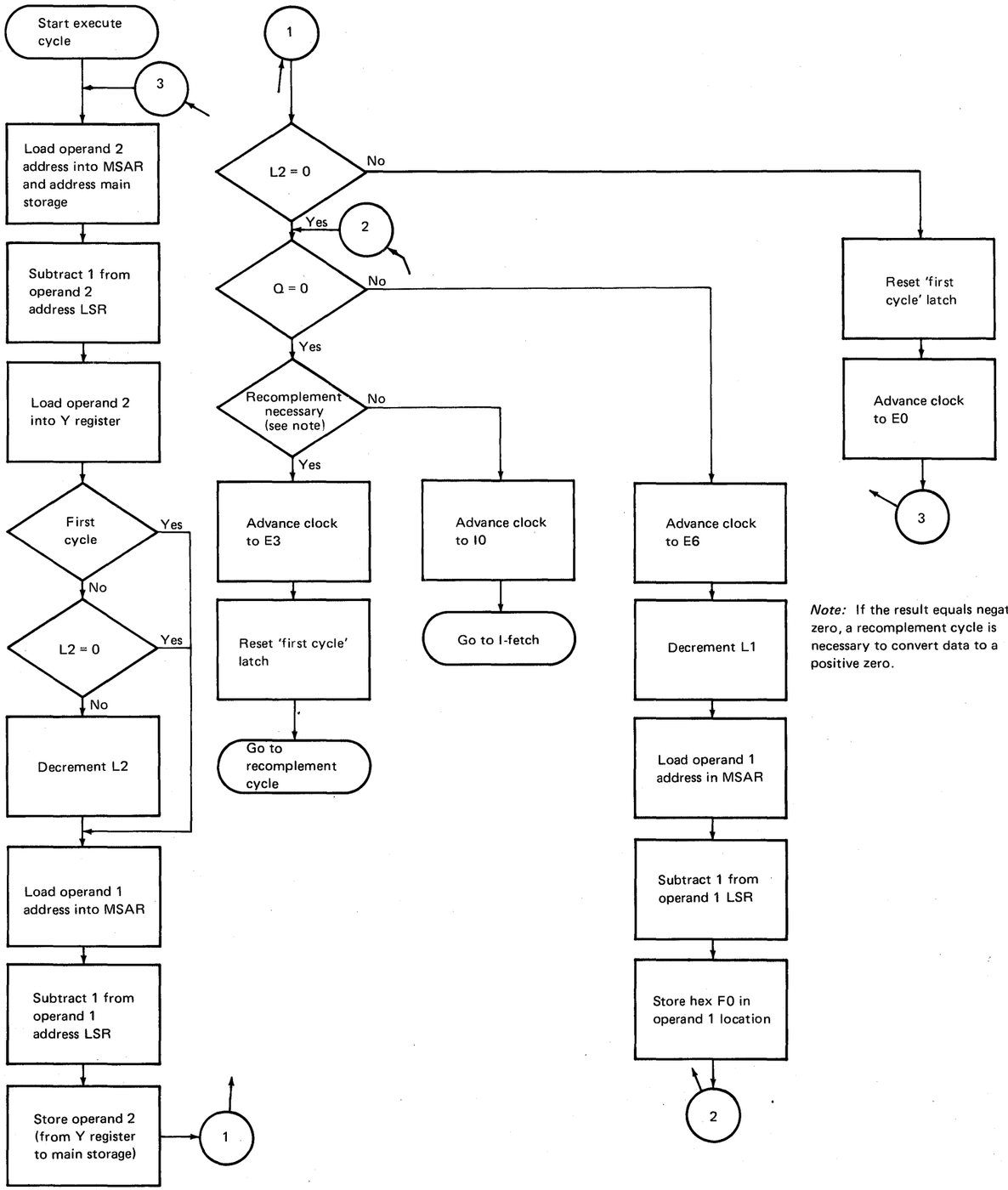
ZERO AND ADD ZONED INSTRUCTION FORMAT

Operands	Op Code (hex)	Q-Byte[1] (hex)		Operand Addresses[2] (hex)			
	Byte 1	Byte 2		Byte 3	Byte 4	Byte 5	Byte 6
A1(L1),A2(L2)	04	L1-L2	L2-1	Operand 1 address		Operand 2 address	
A1(L1),D2(L2,R1)	14	L1-L2	L2-1	Operand 1 address		Op 2 disp from XR1	
A1(L1),D2(L2,R2)	24	L1-L2	L2-1	Operand 1 address		Op 2 disp from XR2	
D1(L1,R1),A2(L2)	44	L1-L2	L2-1	Op 1 disp from XR1	Operand 2 address		
D1(L1,R1),D2(L2,R1)	54	L1-L2	L2-1	Op 1 disp from XR1	Op 2 disp from XR1		
D1(L1,R1),D2(L2,R2)	64	L1-L2	L2-1	Op 1 disp from XR1	Op 2 disp from XR2		
D1(L1,R2),A2(L2)	84	L1-L2	L2-1	Op 1 disp from XR2	Operand 2 address		
D1(L1,R2),D2(L2,R1)	94	L1-L2	L2-1	Op 1 disp from XR2	Op 2 disp from XR1		
D1(L1,R2),D2(L2,R2)	A4	L1-L2	L2-1	Op 1 disp from XR2	Op 2 disp from XR2		

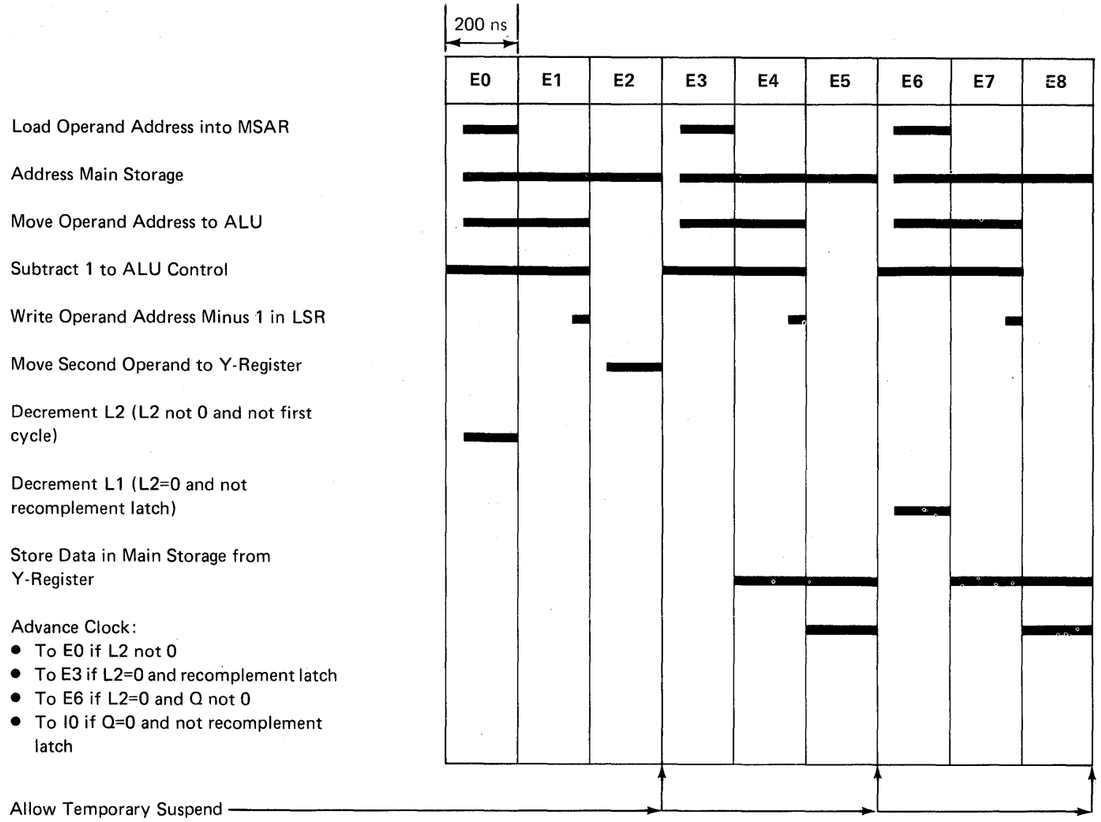
[1]The Q-byte designates the operand length:
L1-L2 (4 bits) = the number of bytes in operand 1, minus the number of bytes in operand 2.
L2-1 (4 bits) = the number of bytes in operand 2, minus 1.
Maximum length of operand 1 is 31 bytes; maximum length of operand 2 is 16 bytes.

[2]The operands may overlap. Address operands by their rightmost bytes.

Sequence and Timing



Note: If the result equals negative zero, a recomplement cycle is necessary to convert data to a positive zero.



See *Recomplement Cycle* later in this section.

Add Zoned Decimal (AZ)

This instruction with algebraic results adds operand 2 to operand 1, byte by byte, and stores the result in operand 1. Both operands are executed as unpacked decimal numbers.

The main storage processor sets the zone bits of all bytes, except the rightmost byte in operand 1, to hexadecimal F (binary 1111). The zone bits of the rightmost byte in operand 1 are set to hexadecimal F if the result of the operation is either positive or zero, or to hexadecimal D (binary 1101) if the result is negative.

Program Status Byte Settings

Bit	Name	Condition Indicated
7	Equal	Zero result
6	Low	Negative result
5	High	Positive result
4	Decimal overflow	Carry occurred from the leftmost position of operand 1
3	Test false	Bit not affected
2	Binary overflow	Bit not affected

See *Recomplement Cycle* later in this section.

Sequence and Timing
See *Subtract Zoned Decimal (SZ)*

ADD ZONED DECIMAL INSTRUCTION FORMAT

Operands	Op Code (hex)	Q-Byte[1] (hex)		Operand Addresses[2] (hex)			
	Byte 1	Byte 2		Byte 3	Byte 4	Byte 5	Byte 6
A1(L1),A2(L2)	06	L1-L2	L2-1	Operand 1 address		Operand 2 address	
A1(L1),D2(L2,R1)	16	L1-L2	L2-1	Operand 1 address		Op 2 disp from XR1	
A1(L1),D2(L2,R2)	26	L1-L2	L2-1	Operand 1 address		Op 2 disp from XR2	
D1(L1,R1),A2(L2)	46	L1-L2	L2-1	Op 1 disp from XR1	Operand 2 address		
D1(L1,R1),D2(L2,R1)	56	L1-L2	L2-1	Op 1 disp from XR1	Op 2 disp from XR1		
D1(L1,R1),D2(L2,R2)	66	L1-L2	L2-1	Op 1 disp from XR1	Op 2 disp from XR2		
D1(L1,R2),A2(L2)	86	L1-L2	L2-1	Op 1 disp from XR2	Operand 2 address		
D1(L1,R2),D2(L2,R1)	96	L1-L2	L2-1	Op 1 disp from XR2	Op 2 disp from XR1		
D1(L1,R2),D2(L2,R2)	A6	L1-L2	L2-1	Op 1 disp from XR2	Op 2 disp from XR2		

[1]The Q-byte designates the operand length:
L1-L2 (4 bits) = the number of bytes in operand 1, minus the number of bytes in operand 2.
L2-1 (4 bits) = the number of bytes in operand 2, minus 1.
Maximum length of operand 1 is 31 bytes; maximum length of operand 2 is 16 bytes.

[2]The operands may overlap. Address operands by their rightmost bytes.

Subtract Zoned Decimal (SZ)

This instruction with algebraic results subtracts operand 2 from operand 1, byte by byte, and stores the result in operand 1. Both operands are executed as unpacked decimal numbers.

The main storage processor sets the zone bits of all operand 1 bytes, except the rightmost byte, to hexadecimal F (binary 1111). The zone bits of the rightmost byte in operand 1 are set to hexadecimal F if the result of the operation is either positive or zero, or to hexadecimal D (binary 1101) if the result is negative.

Program Status Byte Settings

Bit	Name	Condition Indicated
7	Equal	Zero result
6	Low	Negative result
5	High	Positive result
4	Decimal overflow	Carry occurred from the leftmost position of operand 1
3	Test false	Bit not affected
2	Binary overflow	Bit not affected

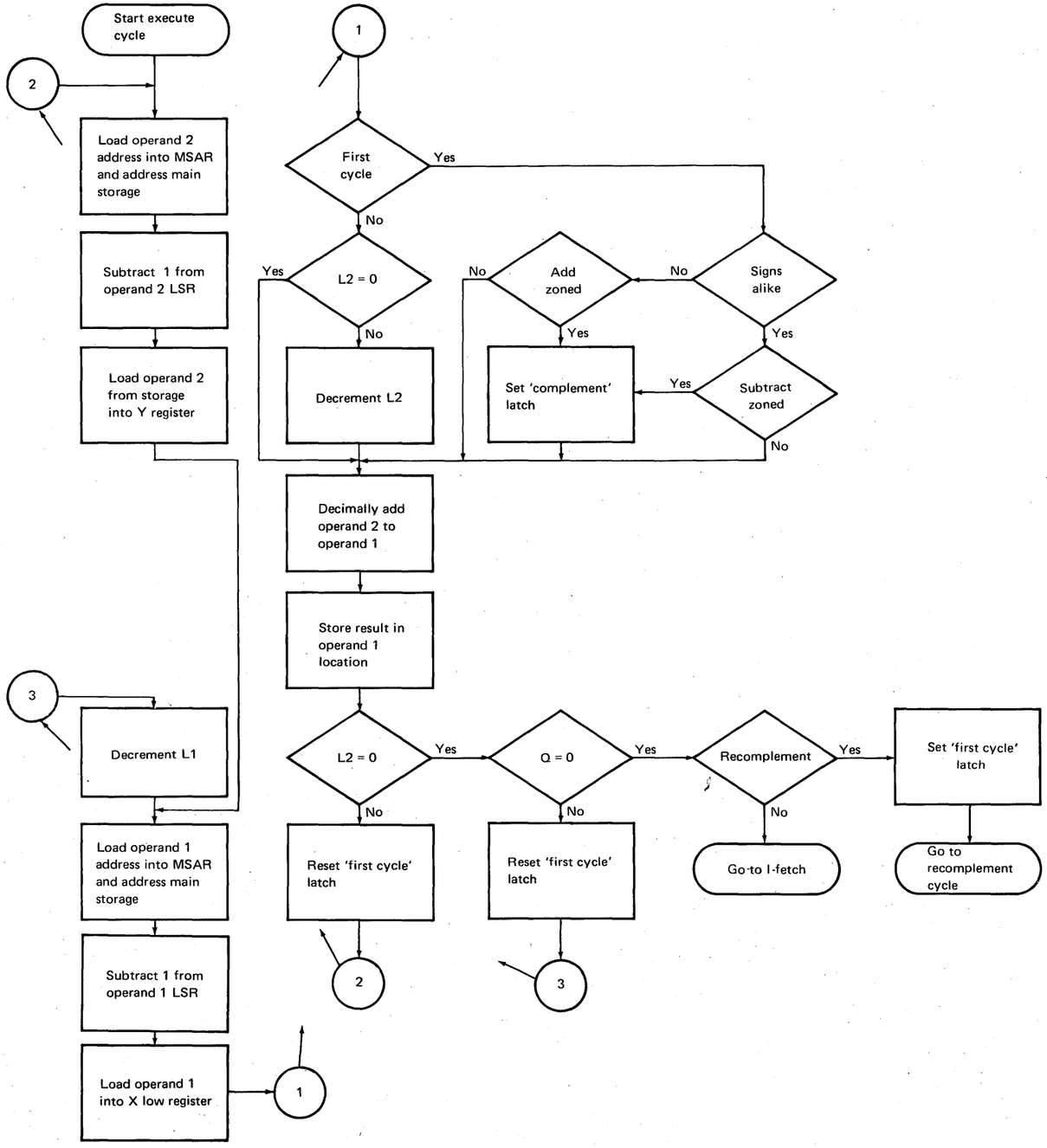
SUBTRACT ZONED DECIMAL INSTRUCTION FORMAT

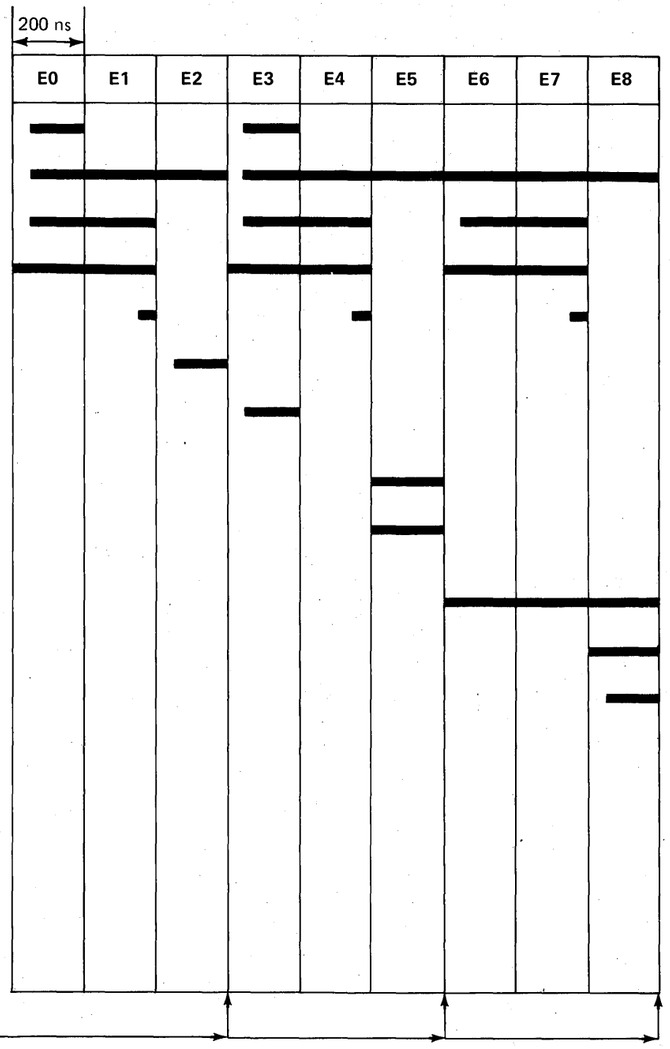
Operands	Op Code (hex)	Q-Byte[1] (hex)		Operand Addresses[2] (hex)			
	Byte 1	Byte 2		Byte 3	Byte 4	Byte 5	Byte 6
A1(L1),A2(L2)	07	L1-L2	L2-1	Operand 1 address		Operand 2 address	
A1(L1),D2(L2,R1)	17	L1-L2	L2-1	Operand 1 address		Op 2 disp from XR1	
A1(L1),D2(L2,R2)	27	L1-L2	L2-1	Operand 1 address		Op 2 disp from XR2	
D1(L1,R1),A2(L2)	47	L1-L2	L2-1	Op 1 disp from XR1	Operand 2 address		
D1(L1,R1),D2(L2,R1)	57	L1-L2	L2-1	Op 1 disp from XR1	Op 2 disp from XR1		
D1(L1,R1),D2(L2,R2)	67	L1-L2	L2-1	Op 1 disp from XR1	Op 2 disp from XR2		
D1(L1,R2),A2(L2)	87	L1-L2	L2-1	Op 1 disp from XR2	Operand 2 address		
D1(L1,R2),D2(L2,R1)	97	L1-L2	L2-1	Op 1 disp from XR2	Op 2 disp from XR1		
D1(L1,R2),D2(L2,R2)	A7	L1-L2	L2-1	Op 1 disp from XR2	Op 2 disp from XR2		

[1]The Q-byte designates the operand length:
L1-L2 (4 bits) = the number of bytes in operand 1, minus the number of bytes in operand 2.
L2-1 (4 bits) = the number of bytes in operand 2, minus 1.
Maximum length of operand 1 is 31 bytes; maximum length of operand 2 is 16 bytes.

[2]The operands may overlap. Address operands by their rightmost bytes.

Sequence and Timing for Add or Subtract Zoned Decimal (AZ, SZ)



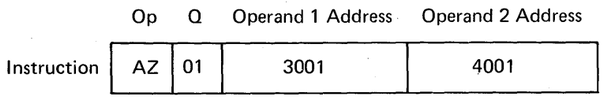


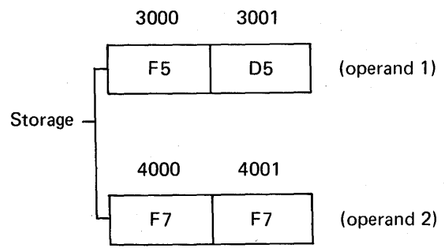
Recomplement Cycle

The zero and added zoned, add zoned decimal, and subtract zoned decimal instructions are the only instructions that use the recomplement cycle. The result of the decimal addition can be stored in the operand 1 location as a true value or as the complement of the true value. If the result is a true value, no recomplement cycle is necessary.

A recomplement cycle is necessary when:

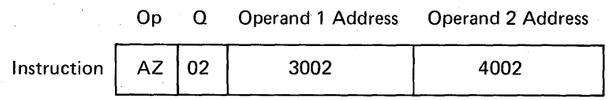
- The result is negative zero or the stored result is in complement form.
- Operand 2 is complemented and there is no carry from the high-order byte after the algebraic addition.

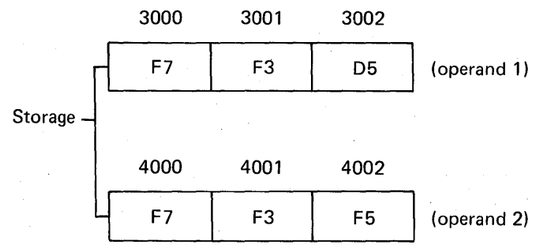




Instruction Execution Chart

After Cycle	Q	Complement	PSR 4	Recomplement	Operand 1 in Storage	Next Clock
I-Fetch	01	Off	XXX	Off	F5D5	E0
1	01	On	Off	Off	F5D8	E0
2	01	On	Off	Off	F7D8	E3
Recomplement 1	01	On	Off	On	F7F2	E3
Recomplement 2	00	On	Off	On	F2F2	I0





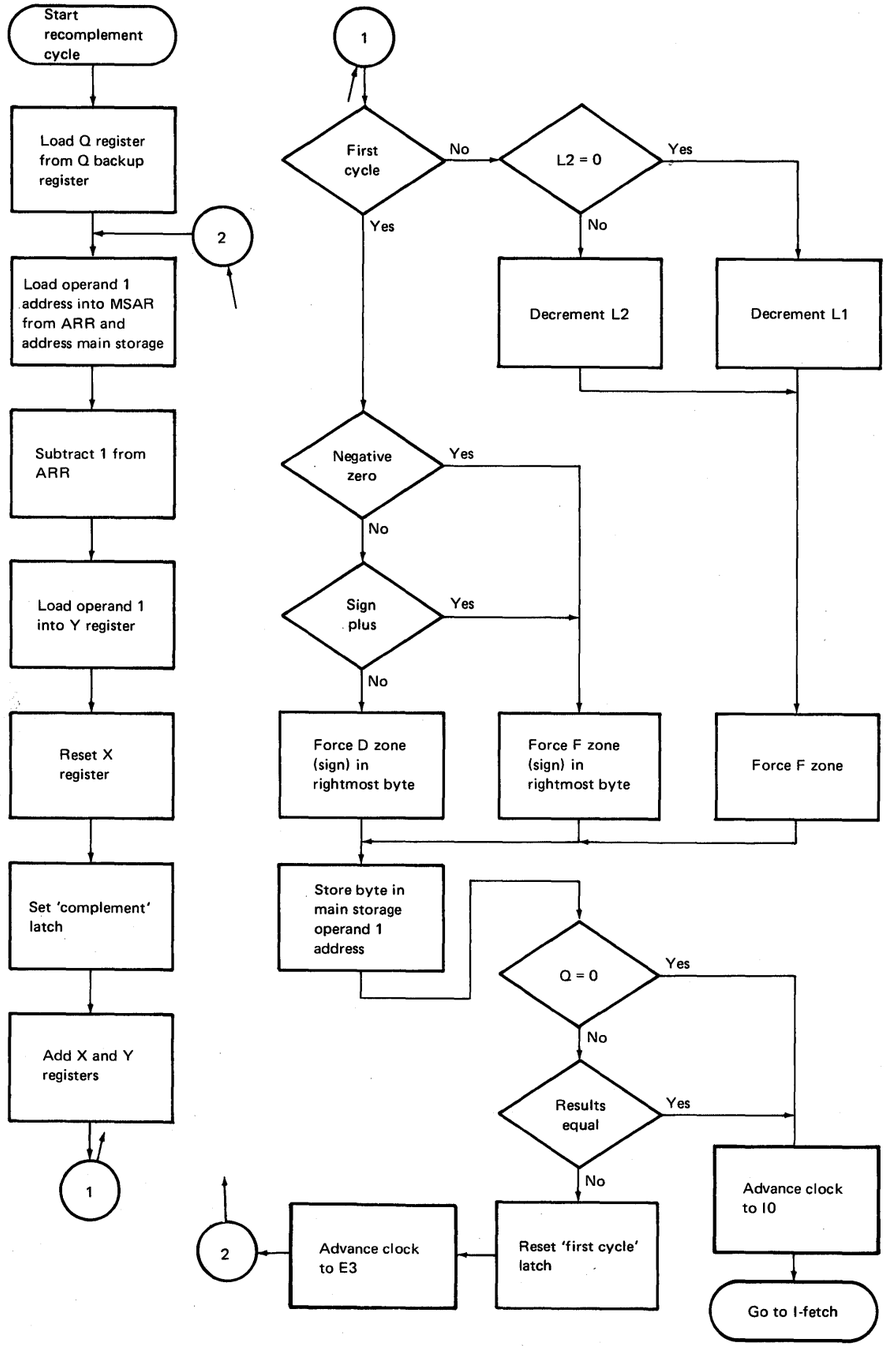
Instruction Execution Chart

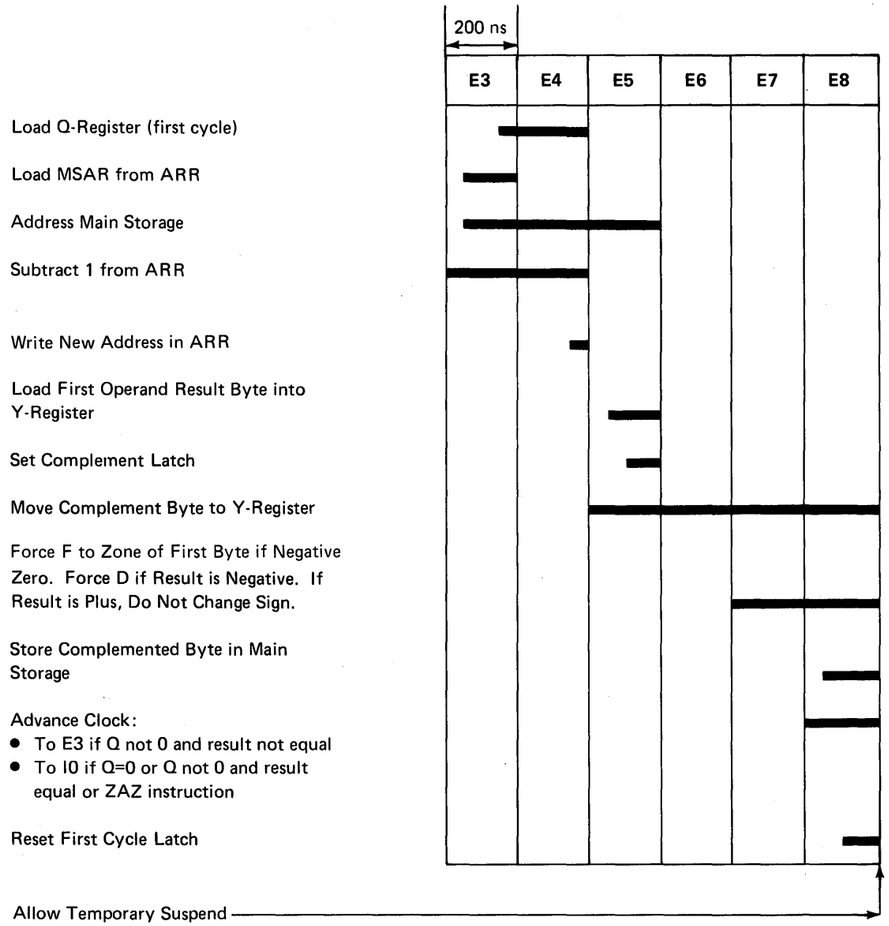
After Cycle	Q	PSR 6 (Internal)	PSR 7	Recomplement	Operand 1 in Storage	Next Clock
I-Fetch	02	Off	On	Off	F7F3D5	E0
1	02	On	On	Off	F7F3D0	E0
2	01	On	On	Off	F7F0D0	E0
3	00	On	On	Off	F0F0D0	F3
Recomplement 1	02[1]	Off	On	On	F0F0F0	I0

[1] Final Q = 2, therefore the 2 recomplement cycles are saved.

Example: Negative zero, recomplement cycle necessary.

Note: A recomplement cycle is necessary if there is no carry from the high-order byte; a recomplement cycle is not necessary if there is a carry from the high-order byte.


Start recomplement cycle
Load Q register from Q backup register
Load operand 1 address into MSAR from ARR and address main storage
Subtract 1 from ARR
Load operand 1 into Y register
Reset X register
Set 'complement' latch
Add X and Y registers
1
2
First cycle
No
Yes
L2 = 0
Yes
No
Decrement L2
Decrement L1
Negative zero
Yes
No
Sign plus
Yes
No
Force D zone (sign) in rightmost byte
Force F zone (sign) in rightmost byte
Force F zone
Store byte in main storage operand 1 address
Q = 0
Yes
No
Results equal
Yes
No
Advance clock to I0
Go to I-fetch
Reset 'first cycle' latch
Advance clock to E3
2


200 ns
E3
E4
E5
E6
E7
E8
Load Q-Register (first cycle)
Load MSAR from ARR
Address Main Storage
Subtract 1 from ARR
Write New Address in ARR
Load First Operand Result Byte into Y-Register
Set Complement Latch
Move Complement Byte to Y-Register
Force F to Zone of First Byte if Negative Zero. Force D if Result is Negative. If Result is Plus, Do Not Change Sign.
Store Complemented Byte in Main Storage
Advance Clock:
• To E3 if Q not 0 and result not equal
• To I0 if Q=0 or Q not 0 and result equal or ZAZ instruction
Reset First Cycle Latch
Allow Temporary Suspend

This page intentionally left blank.

Add Logical Characters (ALC)

This instruction adds the binary number in operand 2 to the binary number in operand 1 and stores the result in operand 1. Both the operands and the result are executed as unsigned binary numbers.

Program Status Byte Settings

Bit	Name	Condition Indicated
7	Equal	Zero result
6	Low	No carry occurred from the high-order byte and result not zero
5	High	Carry occurred from the high-order byte and result not zero
4	Decimal overflow	Bit not affected
3	Test false	Bit not affected
2	Binary overflow	Carry occurred from the high-order byte

Sequence and Timing

See *Subtract Logical Characters (SLC)*

ADD LOGICAL CHARACTERS INSTRUCTION FORMAT

Operands	Op Code (hex)	Q-Byte[1] (hex)	Operand Addresses[2] (hex)			
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
A1(L1),A2	0E	L1-1	Operand 1 address		Operand 2 address	
A1(L1),D2(,R1)	1E	L1-1	Operand 1 address		Op 2 disp from XR1	
A1(L1),D2(,R2)	2E	L1-1	Operand 1 address		Op 2 disp from XR2	
D1(L1,R1),A2	4E	L1-1	Op 1 disp from XR1	Operand 2 address		
D1(L1,R1),D2(,R1)	5E	L1-1	Op 1 disp from XR1	Op 2 disp from XR1		
D1(L1,R1),D2(,R2)	6E	L1-1	Op 1 disp from XR1	Op 2 disp from XR2		
D1(L1,R2),A2	8E	L1-1	Op 1 disp from XR2	Operand 2 address		
D1(L1,R2),D2(,R1)	9E	L1-1	Op 1 disp from XR2	Op 2 disp from XR1		
D1(L1,R2),D2(,R2)	AE	L1-1	Op 1 disp from XR2	Op 2 disp from XR2		

[1]The Q-byte designates the operand length:
L1-1 = the number of bytes in either operand, minus 1.
Maximum length of each operand is 256 bytes; both operands must be the same length.

[2]The operands may overlap. Address operands by their rightmost bytes.

Subtract Logical Characters (SLC)

This instruction subtracts the binary number in operand 2 from the binary number in operand 1 and stores the result in operand 1. Both the operands and the result are executed as unsigned binary numbers.

For example:

Operand 1	0110	1101
Operand 2	0111	1110
Result	1110	1011

Program Status Byte Settings

CAUTION
The results of the program status byte are not reliable if they are selected.

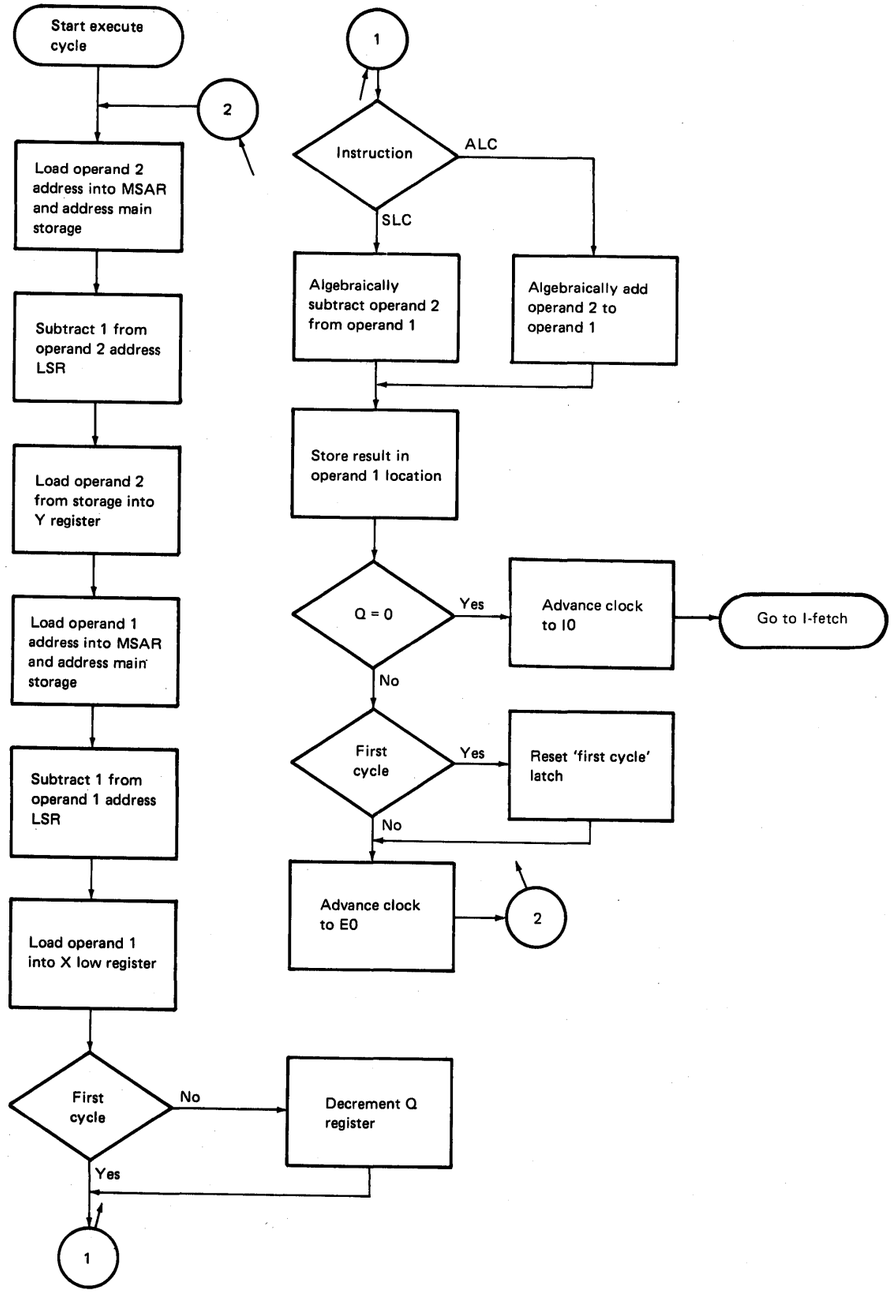
Bit	Name	Condition Indicated
7	Equal	Zero result
6	Low	Operand 1 was smaller than operand 2 before execution
5	High	First operand greater than second operand
4	Decimal overflow	Bit not affected
3	Test false	Bit not affected
2	Binary overflow	Bit not affected

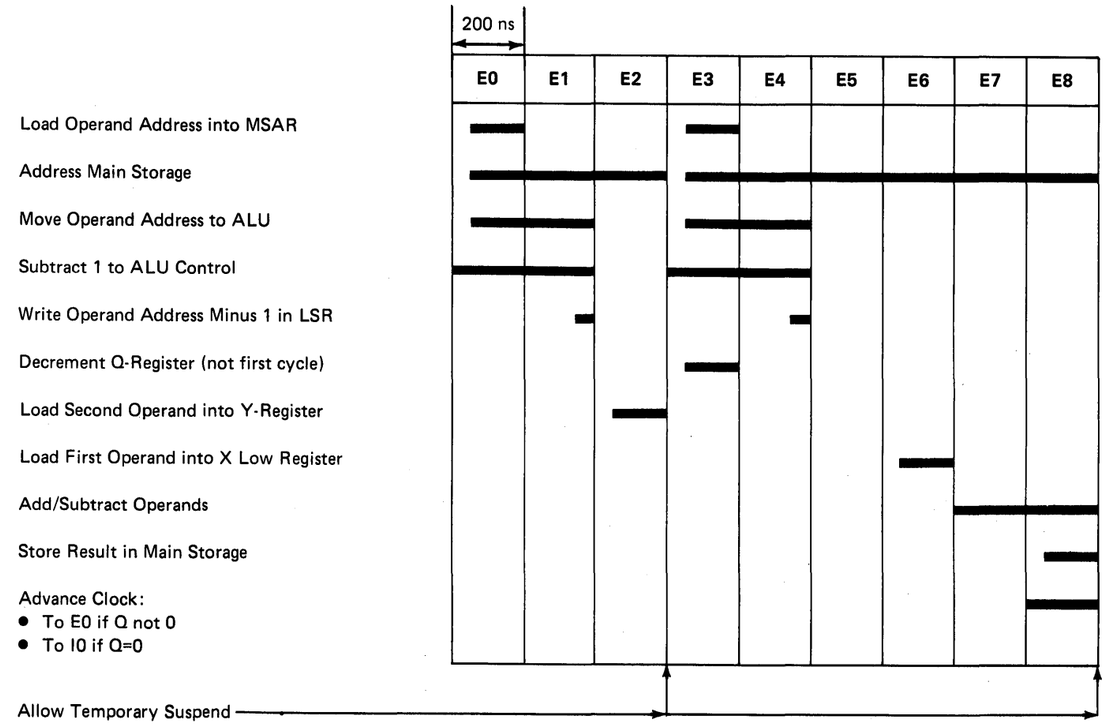
SUBTRACT LOGICAL CHARACTERS INSTRUCTION FORMAT

Operands	Op Code (hex)	Q-Byte[1] (hex)	Operand Addresses[2] (hex)			
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
A1(L1),A2	0F	L1-1	Operand 1 address		Operand 2 address	
A1(L1),D2(,R1)	1F	L1-1	Operand 1 address		Op 2 disp from XR1	
A1(L1),D2(,R2)	2F	L1-1	Operand 1 address		Op 2 disp from XR2	
D1(L1,R1),A2	4F	L1-1	Op 1 disp from XR1	Operand 2 address		
D1(L1,R1),D2(,R1)	5F	L1-1	Op 1 disp from XR1	Op 2 disp from XR1		
D1(L1,R1),D2(,R2)	6F	L1-1	Op 1 disp from XR1	Op 2 disp from XR2		
D1(L1,R2),A2	8F	L1-1	Op 1 disp from XR2	Operand 2 address		
D1(L1,R2),D2(,R1)	9F	L1-1	Op 1 disp from XR2	Op 2 disp from XR1		
D1(L1,R2),D2(,R2)	AF	L1-1	Op 1 disp from XR2	Op 2 disp from XR2		

[1]The Q-byte designates the operand length:
L1-1 = the number of bytes in either operand, minus 1.
Maximum length of each operand is 256 bytes; both operands must be the same length.
[2]The operands may overlap. Address operands by their rightmost bytes.

Sequence and Timing for Add or Subtract Logical Characters (ALC, SLC)





Add to Register (A)

This instruction adds the binary number in operand 1 to the contents of the 2-byte register selected by the Q-byte and stores the result in the specified register. The Q-bytes used to specify various registers are:

Q-Byte (binary)	Q-Byte (hex)	Register Specified
0000 0000	00	None. The system ignores (no-ops) the instruction.
0000 0001	01	XR1.
0000 0010	02	XR2.
0000 0100	04	Program status register.
0000 1000	08	Address recall register.
0001 0000	10	Instruction address register.
0010 0000	20	Instruction address register.
0100 0000	40	Reserved; do not use.
1000 0000	80	Reserved; do not use.

Program Status Byte Settings

CAUTION
The results of the program status byte are not reliable if they are selected.

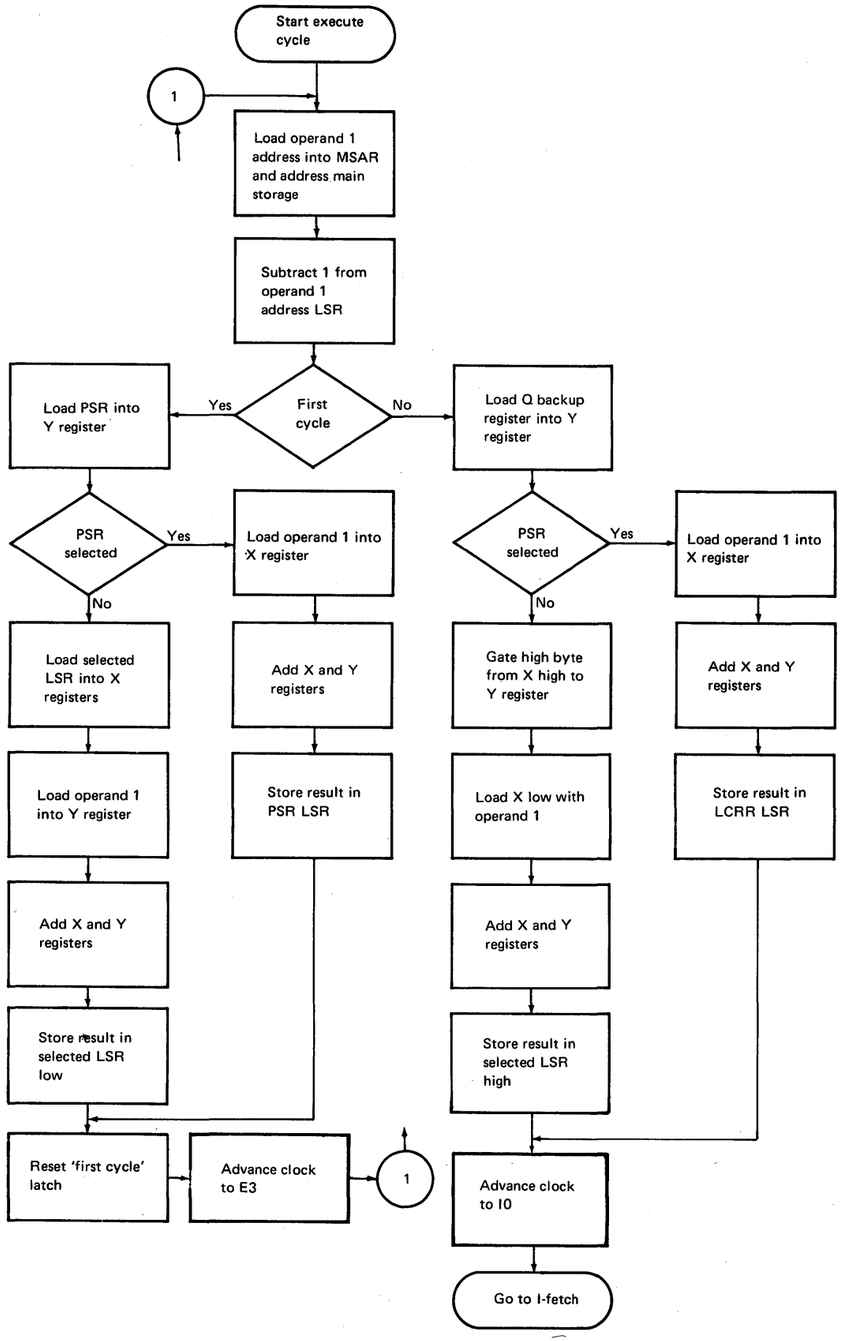
Bit	Name	Condition Indicated
7	Equal	Zero result
6	Low	No carry occurred from the leftmost byte and result not zero
5	High	Carry occurred from the leftmost byte and result not zero
4	Decimal overflow	Bit not used
3	Test false	Bit not used
2	Binary overflow	Carry occurred from the leftmost byte

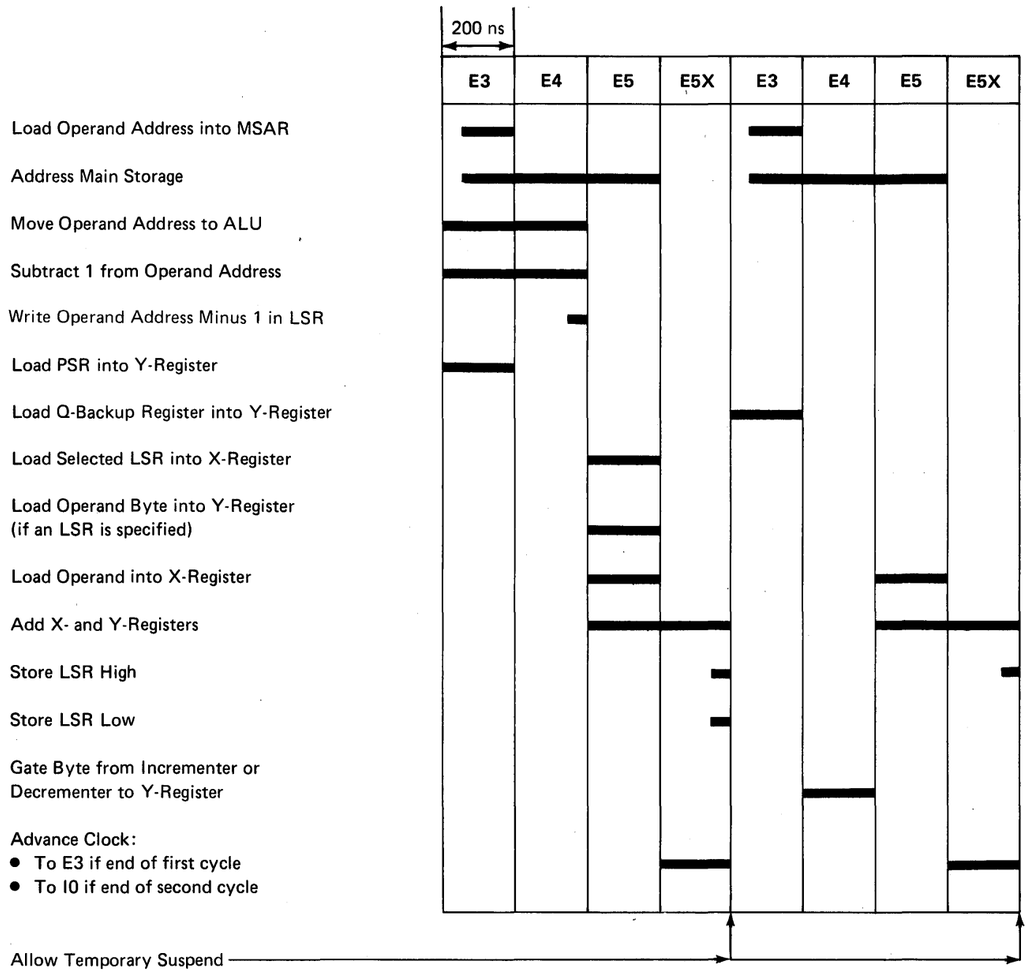
ADD TO REGISTER INSTRUCTION FORMAT

Operands	Op Code (hex)	Q-Byte[1] (binary)	Operand Address[2] (hex)	
	Byte 1	Byte 2	Byte 3	Byte 4
A1,RX	36	Rx	Operand 1 address	
D1(,R1),RX	76	Rx	Op 1 disp from XR1	
D1(,R2),RX	B6	Rx	Op 1 disp from XR2	

[1]Rx specifies the register whose contents are modified by the machine instruction.

[2]Operand 1 is a 2-byte field addressed by its rightmost byte; operand 2 is not used.


Start execute cycle
1
Load operand 1 address into MSAR and address main storage
Subtract 1 from operand 1 address LSR
First cycle
Yes
No
Load PSR into Y register
Load Q backup register into Y register
PSR selected
Yes
No
Load operand 1 into X register
Load selected LSR into X registers
Add X and Y registers
Load operand 1 into Y register
Store result in PSR LSR
Add X and Y registers
Store result in selected LSR low
Reset 'first cycle' latch
Advance clock to E3
1
PSR selected
Yes
No
Load operand 1 into X register
Gate high byte from X high to Y register
Add X and Y registers
Load X low with operand 1
Store result in LCRR LSR
Add X and Y registers
Store result in selected LSR high
Advance clock to I0
Go to I-fetch


200 ns
E3
E4
E5
E5X
E3
E4
E5
E5X
Load Operand Address into MSAR
Address Main Storage
Move Operand Address to ALU
Subtract 1 from Operand Address
Write Operand Address Minus 1 in LSR
Load PSR into Y-Register
Load Q-Backup Register into Y-Register
Load Selected LSR into X-Register
Load Operand Byte into Y-Register (if an LSR is specified)
Load Operand into X-Register
Add X- and Y-Registers
Store LSR High
Store LSR Low
Gate Byte from Incrementer or Decrementer to Y-Register
Advance Clock:
• To E3 if end of first cycle
• To I0 if end of second cycle
Allow Temporary Suspend

Data Control Instructions

Move Hexadecimal Character (MVX)

This instruction moves the numeric part (bits 4-7) or the zone part (bits 0-3) of operand 2 to the numeric or zone part of operand 1, as specified by the Q-byte. The Q-byte codes are:

Q-Byte (binary)	Q-Byte (hex)	Meaning
0000 0000	00	Move data from operand 2 zone portion to operand 1 zone portion
0000 0001	01	Move data from operand 2 numeric portion to operand 1 zone portion
0000 0010	02	Move data from operand 2 zone portion to operand 1 numeric portion
0000 0011	03	Move data from operand 2 numeric portion to operand 1 numeric portion

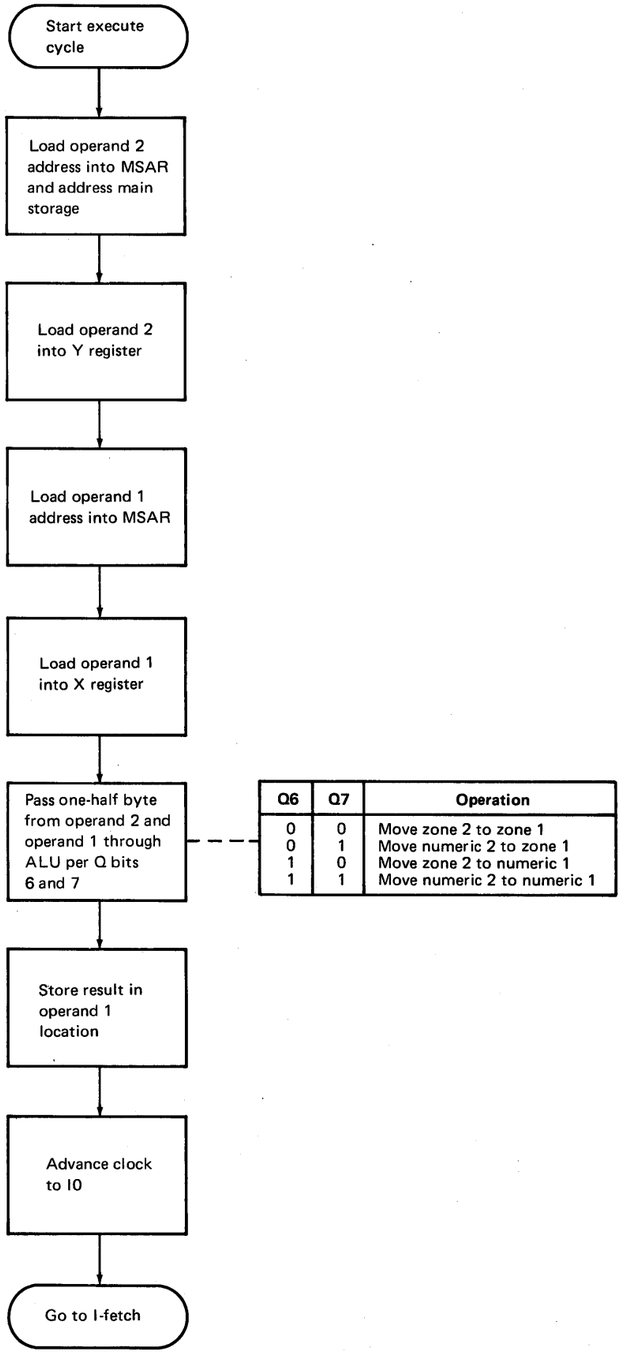
Program Status Byte Settings

This instruction does not affect the program status register.

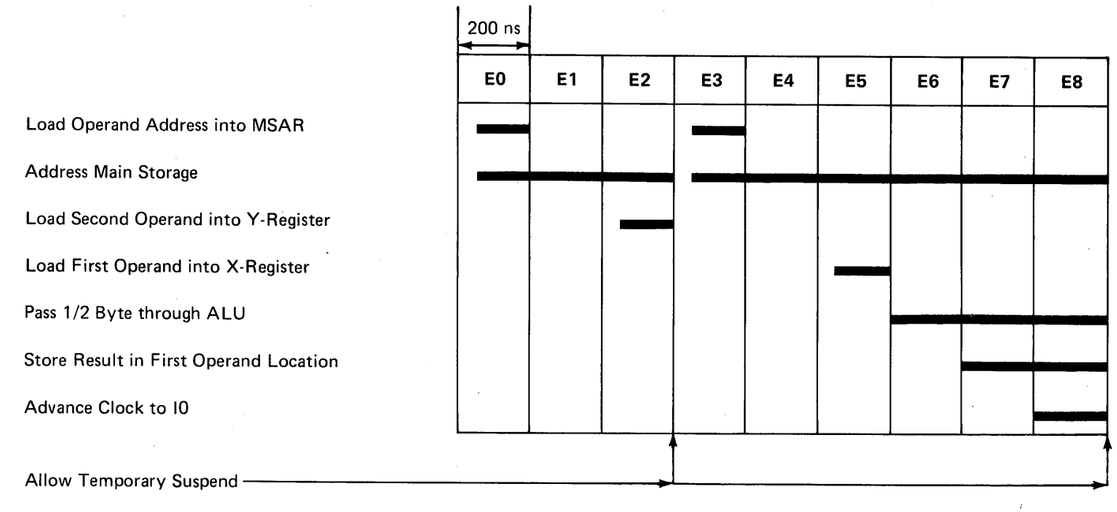
MOVE HEXADECIMAL CHARACTER INSTRUCTION FORMAT

Operands	Op Code (hex)	Q-Byte[1] (hex)	Operand Addresses[2] (hex)			
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
A1(I),A2	08	I	Operand 1 address		Operand 2 address	
A1(I),D2(,R1)	18	I	Operand 1 address		Op 2 disp from XR1	
A1(I),D2(,R2)	28	I	Operand 1 address		Op 2 disp from XR2	
D1(I,R1),A2	48	I	Op 1 disp from XR1	Operand 2 address		
D1(I,R1),D2(,R1)	58	I	Op 1 disp from XR1	Op 2 disp from XR1		
D1(I,R1),D2(,R2)	68	I	Op 1 disp from XR1	Op 2 disp from XR2		
D1(I,R2),A2	88	I	Op 1 disp from XR2	Operand 2 address		
D1(I,R2),D2(,R1)	98	I	Op 1 disp from XR2	Op 2 disp from XR1		
D1(I,R2),D2(,R2)	A8	I	Op 1 disp from XR2	Op 2 disp from XR2		

[1]I = one byte of immediate data that specifies which portion of each 1-byte operand is used in the operation.
[2]Both operands are 1-byte fields.



Q6	Q7	Operation
0	0	Move zone 2 to zone 1
0	1	Move numeric 2 to zone 1
1	0	Move zone 2 to numeric 1
1	1	Move numeric 2 to numeric 1



Move Characters (MVC)

This instruction places the contents of operand 2, byte by byte, into operand 1. One character can be duplicated through a complete field by setting the operand 2 address one byte to the right of the operand 1 address.

Program Status Byte Settings

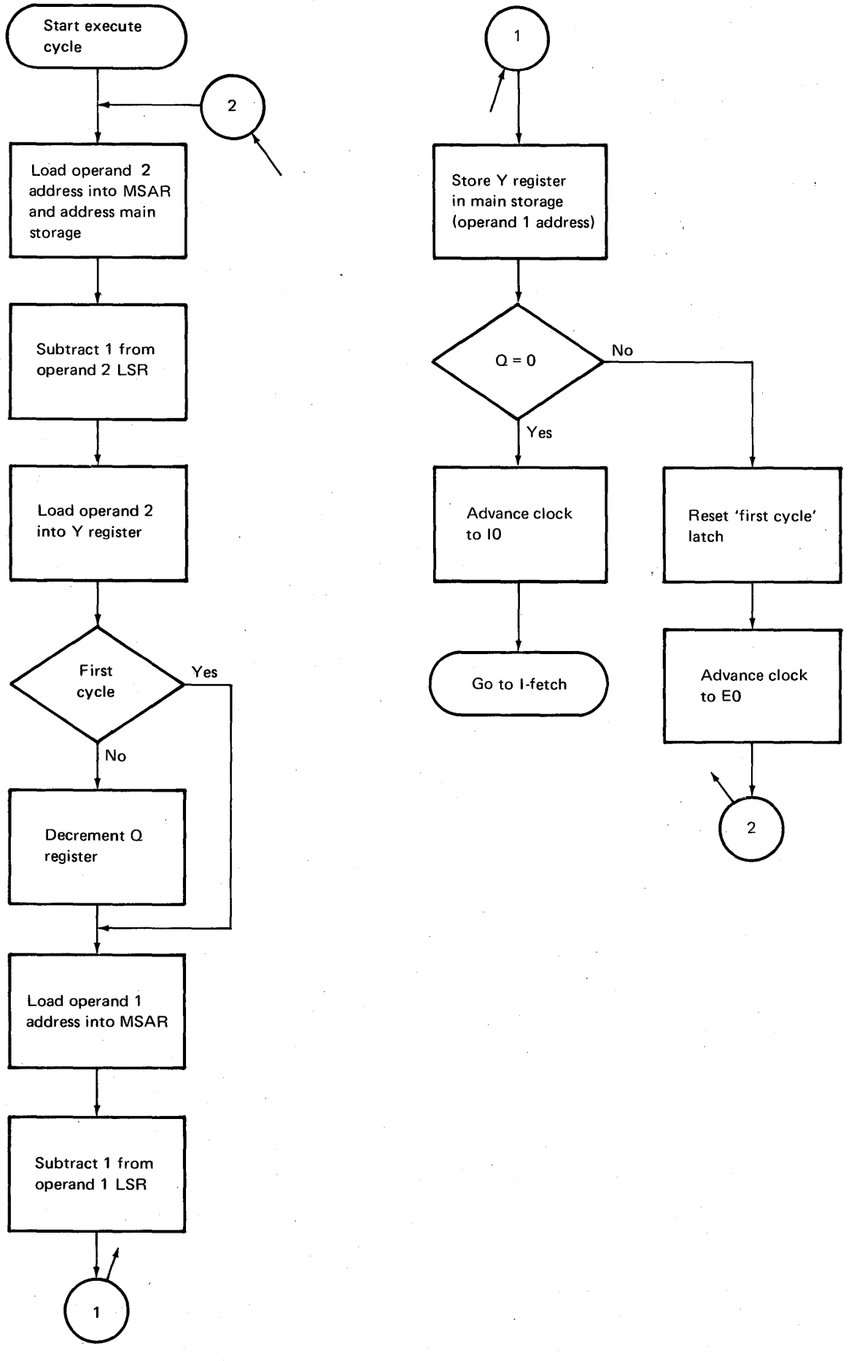
This instruction does not affect the program status register.

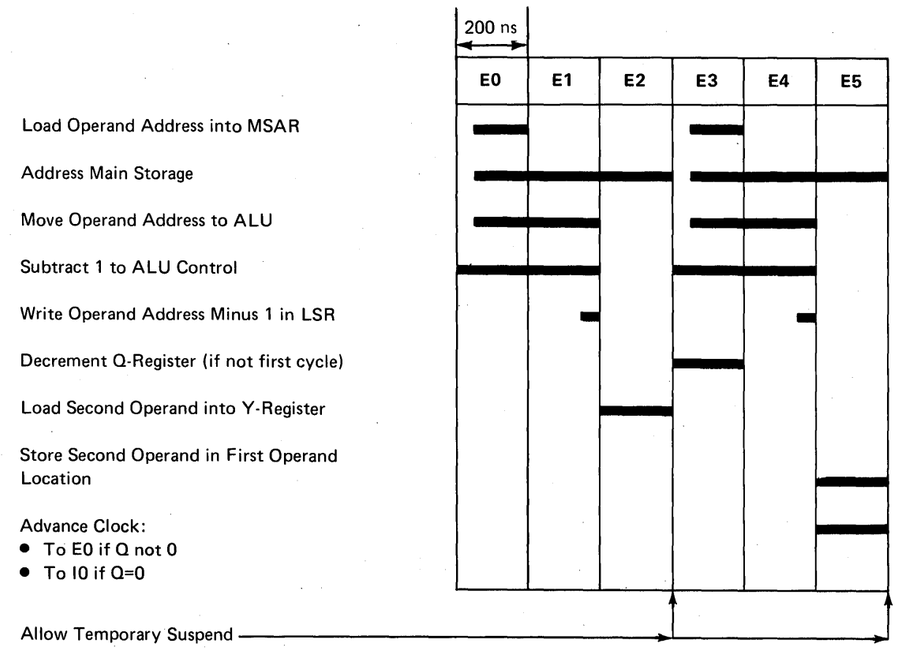
MOVE CHARACTERS INSTRUCTION FORMAT

Operands	Op Code (hex)	Q Byte[1] (hex)	Operand Addresses[2] (hex)			
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
A1(L1),A2	0C	L1-1	Operand 1 address		Operand 2 address	
A1(L1),D2(,R1)	1C	L1-1	Operand 1 address		Op 2 disp from XR1	
A1(L1),D2(,R2)	2C	L1-1	Operand 1 address		Op 2 disp from XR2	
D1(L1,R1),A2	4C	L1-1	Op 1 disp from XR1	Operand 2 address		
D1(L1,R1),D2(,R1)	5C	L1-1	Op 1 disp from XR1	Op 2 disp from XR1		
D1(L1,R1),D2(,R2)	6C	L1-1	Op 1 disp from XR1	Op 2 disp from XR2		
D1(L1,R2),A2	8C	L1-1	Op 1 disp from XR2	Operand 2 address		
D1(L1,R2),D2(,R1)	9C	L1-1	Op 1 disp from XR2	Op 2 disp from XR1		
D1(L1,R2),D2(,R2)	AC	L1-1	Op 1 disp from XR2	Op 2 disp from XR2		

[1]The Q byte designates the operand length:
L1-1 = the number of bytes in either operand, minus 1.
Maximum length of each operand is 256 bytes; both operands must be the same length.

[2]The operands may overlap. Address operands by their rightmost bytes.


Start execute cycle
2
Load operand 2 address into MSAR and address main storage
Subtract 1 from operand 2 LSR
Load operand 2 into Y register
First cycle
Yes
No
Decrement Q register
Load operand 1 address into MSAR
Subtract 1 from operand 1 LSR
1
1
Store Y register in main storage (operand 1 address)
Q = 0
No
Yes
Advance clock to I0
Go to I-fetch
Reset 'first cycle' latch
Advance clock to E0
2


200 ns
E0
E1
E2
E3
E4
E5
Load Operand Address into MSAR
Address Main Storage
Move Operand Address to ALU
Subtract 1 to ALU Control
Write Operand Address Minus 1 in LSR
Decrement Q-Register (if not first cycle)
Load Second Operand into Y-Register
Store Second Operand in First Operand Location
Advance Clock:
• To E0 if Q not 0
• To I0 if Q=0
Allow Temporary Suspend

Edit (ED)

This instruction replaces bytes that contain hexadecimal 20 in operand 1 with characters from operand 2. Starting at the rightmost position in both operands, the main storage processor inspects operand 1 for hexadecimal 20. When the first hexadecimal 20 is found, the first byte is moved from operand 2 into that operand 1 position. The following bytes in operand 1 are then inspected for the next sequential hexadecimal 20. When the main storage processor locates the second hexadecimal 20, it moves the second byte from operand 2 into that operand 1 position. The operation continues until the last byte in operand 1 has been inspected for hexadecimal 20. During the operation, the system sets the zone bits of all replaced operand 1 bytes to hexadecimal F (binary 1111).

Program Status Byte Settings[1]

Bit	Name	Condition Indicated
7	Equal	Operand 2 zero
6	Low	Operand 2 negative
5	High	Operand 2 positive
4	Decimal overflow	Bit not affected
3	Test false	Bit not affected
2	Binary overflow	Bit not affected

[1] The program status byte settings will be as shown only if one of the following condition exists:
1. The program status byte bit 7 was set before edit is executed.
2. The rightmost byte of operand 1 was hex 20.
3. Operand 2 is not zero.

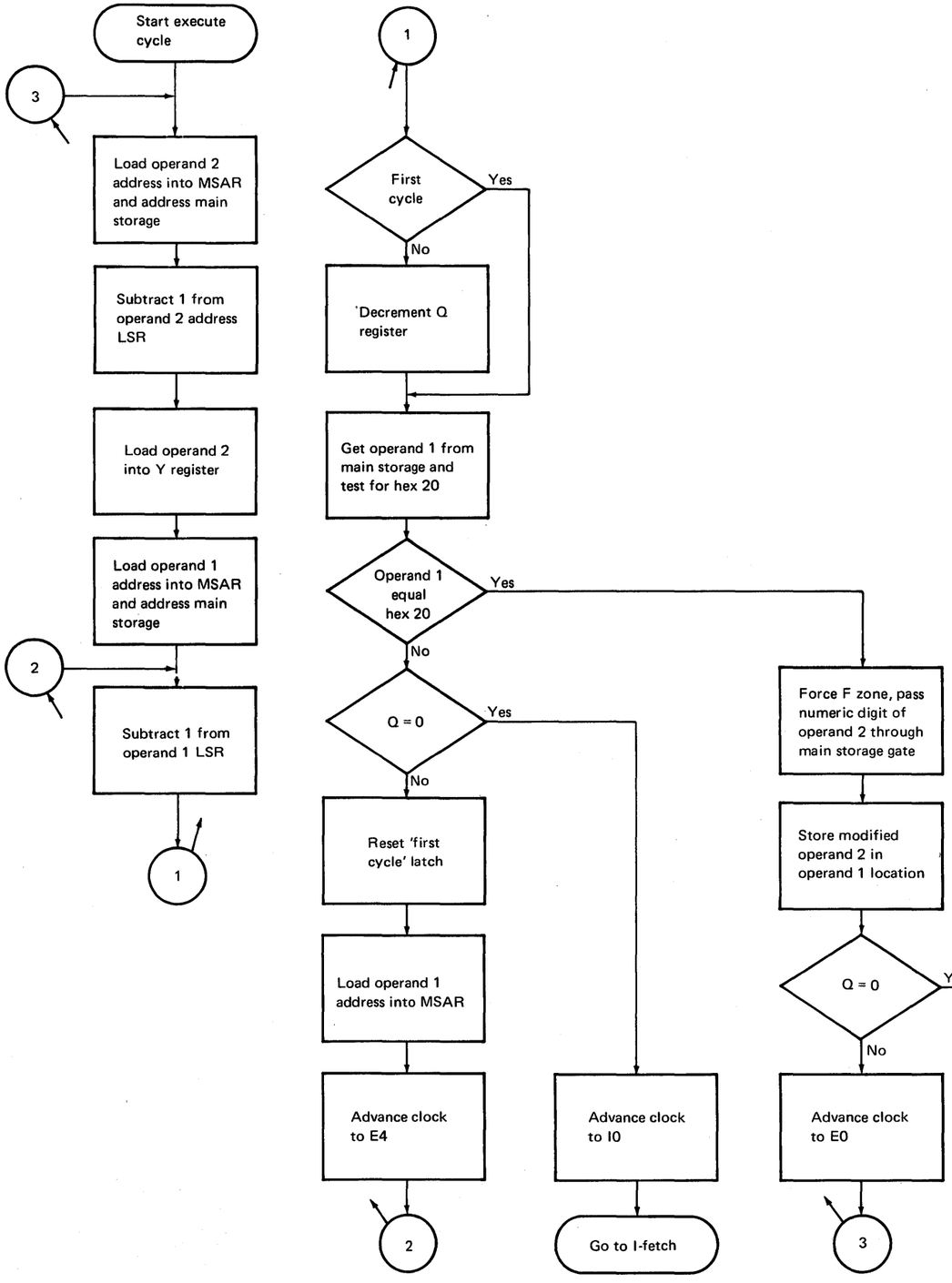
EDIT INSTRUCTION FORMAT

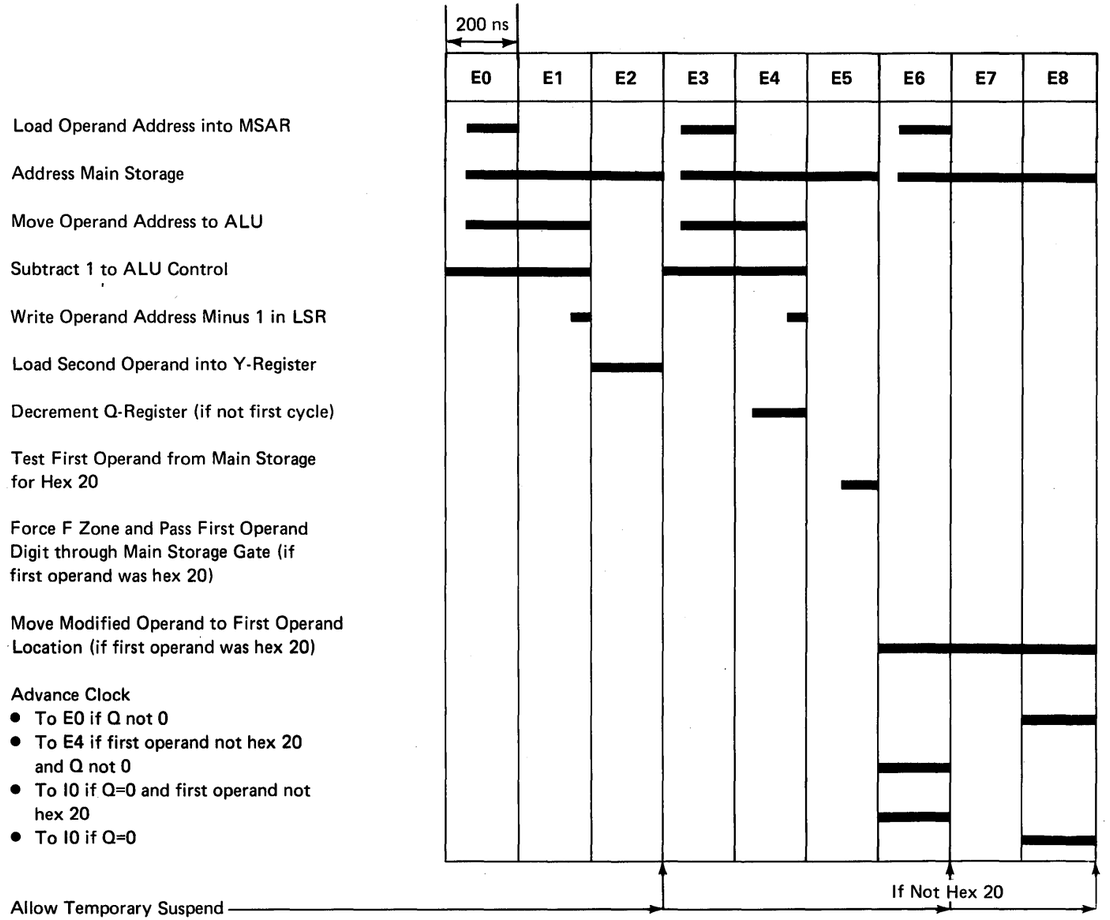
Operands	Op Code (hex)	Q-Byte[1] (hex)	Operand Addresses[2] (hex)			
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
A1(L1),A2	0A	L1-1	Operand 1 address		Operand 2 address	
A1(L1),D2(,R1)	1A	L1-1	Operand 1 address		Op 2 disp from XR3	
A1(L1),D2(,R2)	2A	L1-1	Operand 1 address		Op 2 disp from XR2	
D1(L1,R1),A2	4A	L1-1	Op 1 disp from XR1	Operand 2 address		
D1(L1,R1),D2(,R1)	5A	L1-1	Op 1 disp from XR1	Op 2 disp from XR1		
D1(L1,R1),D2(,R2)	6A	L1-1	Op 1 disp from XR1	Op 2 disp from XR2		
D1(L1,R2),A2	8A	L1-1	Op 1 disp from XR2	Operand 2 address		
D1(L1,R2),D2(,R1)	9A	L1-1	Op 1 disp from XR2	Op 2 disp from XR1		
D1(L1,R2),D2(,R2)	AA	L1-1	Op 1 disp from XR2	Op 2 disp from XR2		

[1]The Q-byte designates the operand length:
L1-1 = the number of bytes in either operand, minus 1.
Operand 2 must contain as many bytes as there are hex 20s in operand 1.

[2]The operands may overlap. Address operands by their rightmost bytes.

Sequence and Timing





Insert and Test Characters (ITC)

The single character at the operand 2 address replaces all the characters to the left of the first significant digit in operand 1. Only decimal digits 1 through 9 are significant.

The operation goes from left to right, filling operand 1 with the character from operand 2. Finding a significant digit in operand 1 ends the operation.

At the end of the operation, the address recall register contains the address of the first significant digit; if no significant digit is found, the address recall register contains the address of the byte to the right of operand 1. This new information remains in the register until the system executes the next decimal add, decimal subtract, zero and add zoned, branch, or insert and test characters instruction.

Program Status Byte Settings

This instruction does not affect the program status register.

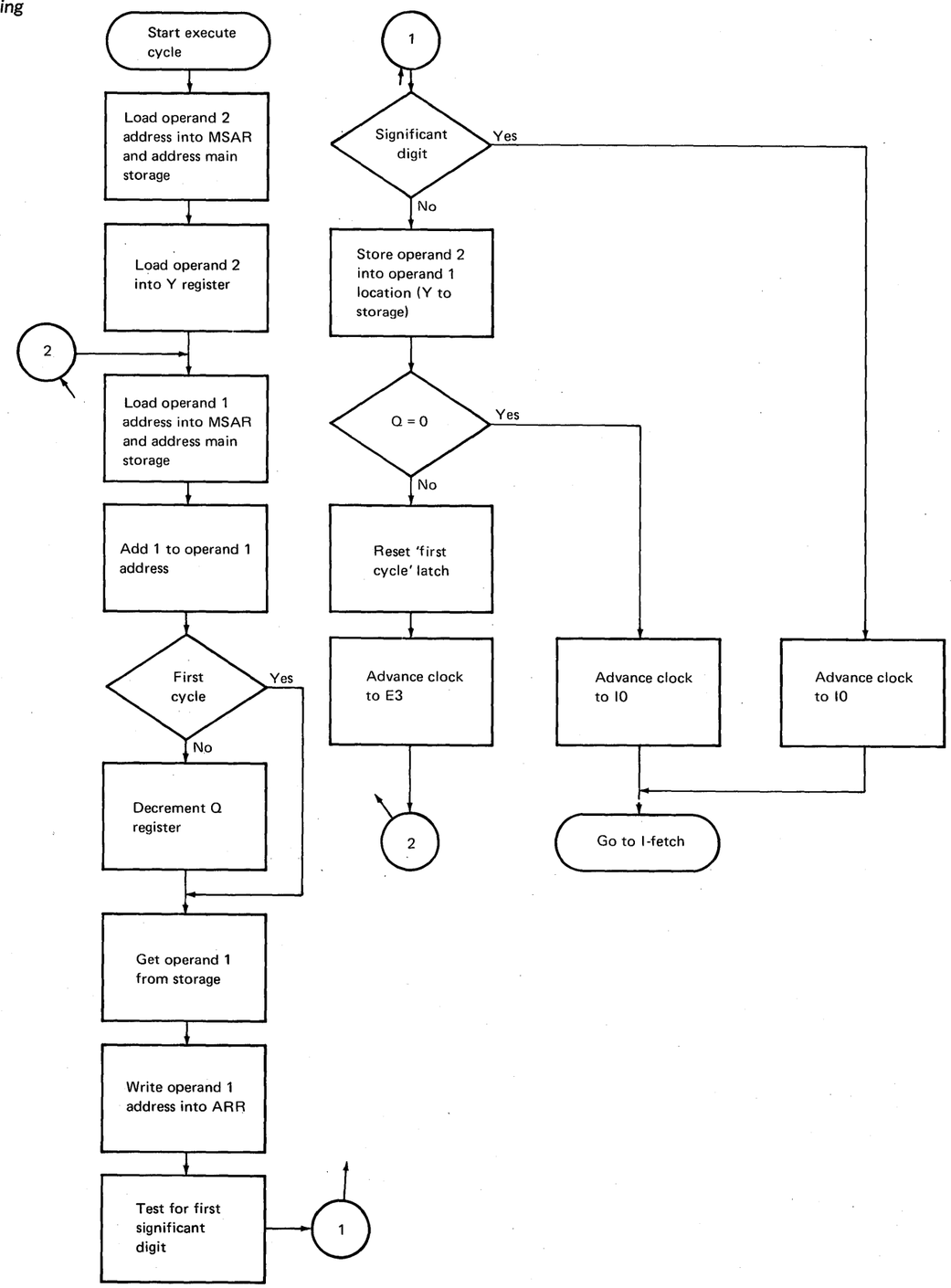
INSERT AND TEST CHARACTERS INSTRUCTION FORMAT

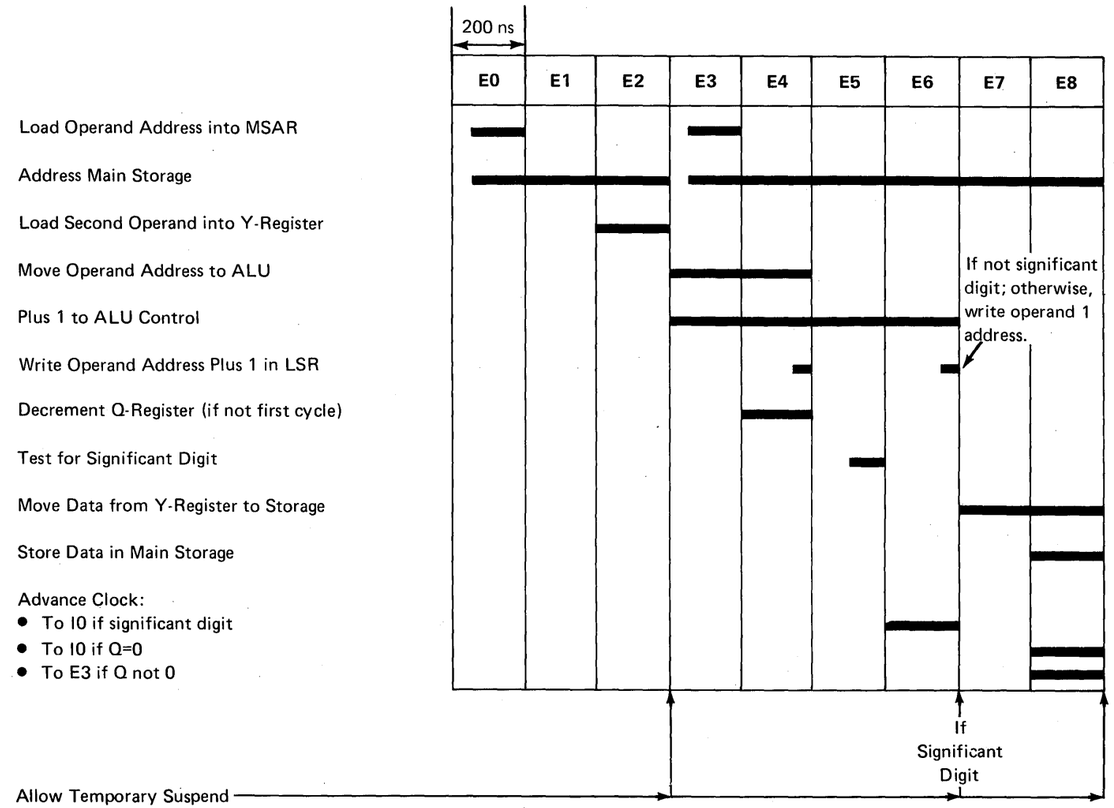
Operands	Op Code (hex)	Q-Byte[1] (hex)	Operand Addresses[2] (hex)			
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
A1(L1),A2	0B	L1-1	Operand 1 address		Operand 2 address	
A1(L1),D2(,R1)	1B	L1-1	Operand 1 address		Op 2 disp from XR1	
A1(L1),D2(,R2)	2B	L1-1	Operand 1 address		Op 2 disp from XR2	
D1(L1,R1),A2	4B	L1-1	Op 1 disp from XR1	Operand 2 address		
D1(L1,R1),D2(,R1)	5B	L1-1	Op 1 disp from XR1	Op 2 disp from XR1		
D1(L1,R1),D2(,R2)	6B	L1-1	Op 1 disp from XR1	Op 2 disp from XR2		
D1(L1,R2),A2	8B	L1-1	Op 1 disp from XR2	Operand 2 address		
D1(L1,R2),D2(,R1)	9B	L1-1	Op 1 disp from XR2	Op 2 disp from XR1		
D1(L1,R2),D2(,R2)	AB	L1-1	Op 1 disp from XR2	Op 2 disp from XR2		

[1]The Q-byte designates the operand length:
L1-1 = the number of bytes in either operand, minus 1.
Operand 2 is a 1-byte field.

[2]Address operand 1 by its leftmost position.

Sequence and Timing





Move Logical Immediate (MVI)

MOVE LOGICAL IMMEDIATE INSTRUCTION FORMAT

Operands	Op Code (hex)	Q-Byte[1] (binary)	Operand Address[2] (hex)	
	Byte 1	Byte 2	Byte 3	Byte 4
A1,I	3C	I	Operand 1 address	
D1(,R1),I	7C	I	Op 1 disp from XR1	
D1(,R2),I	BC	I	Op 1 disp from XR2	

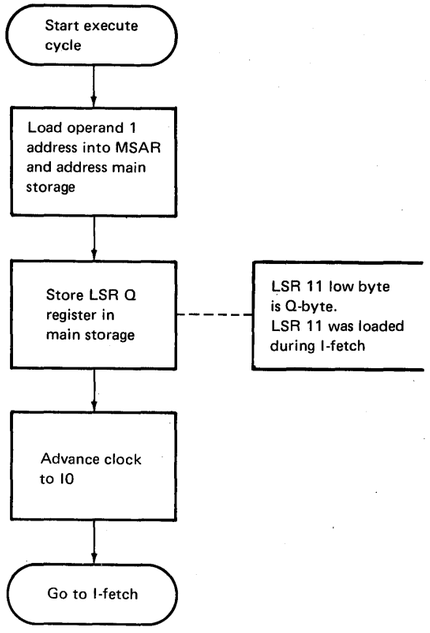
[1]I = 1 byte of immediate data (for example, 1 byte of actual data on a 1-byte mask).
[2]Operand 1 is a 1-byte field; operand 2 is not used.

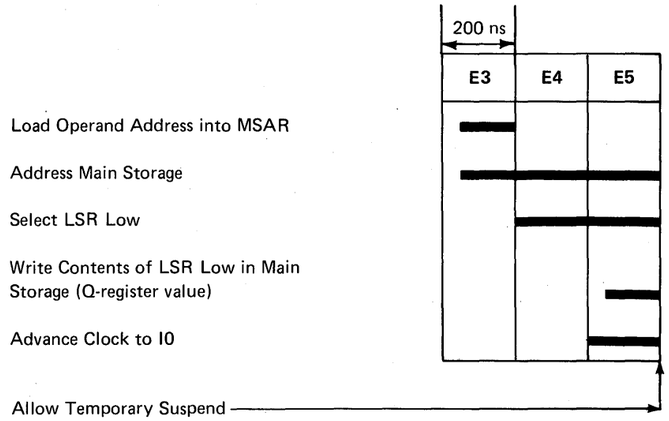
This instruction moves the Q-byte into operand 1.

Program Status Byte Settings

This instruction does not affect the program status register.

Sequence and Timing





Set Bits On Masked (SBN)

The system inspects the Q-byte, bit by bit. If the system finds a binary 1 in the Q-byte, the system sets the comparable bit in the operand byte to 1; if the system finds a binary 0 in the Q-byte, the comparable bit in the operand is not changed.

Program Status Byte Settings

This instruction does not affect the program status register.

SET BITS ON MASKED INSTRUCTION FORMAT

Operands	Op Code (hex)	Q-Byte[1] (binary)	Operand Address[2] (hex)	
	Byte 1	Byte 2	Byte 3	Byte 4
A1,I	3A	xxxx xxxx	Operand 1 address	
D1(,R1),I	7A	xxxx xxxx	Op 1 disp from XR1	
D1(,R2),I	BA	xxxx xxxx	Op 1 disp from XR2	

[1]The Q-byte contains a 1-byte binary mask specifying operand bits to be turned on.

[2]Operand 1 is a 1-byte field; operand 2 is not used.

Set Bits Off Masked (SBF)

The system inspects the Q-byte, bit by bit. If the system finds a binary 1 in the Q-byte, the system sets the comparable bit in the operand byte to 0; if the system finds a binary 0 in the Q-byte, the comparable bit in the operand is not changed.

Program Status Byte Settings

This instruction does not affect the program status register.

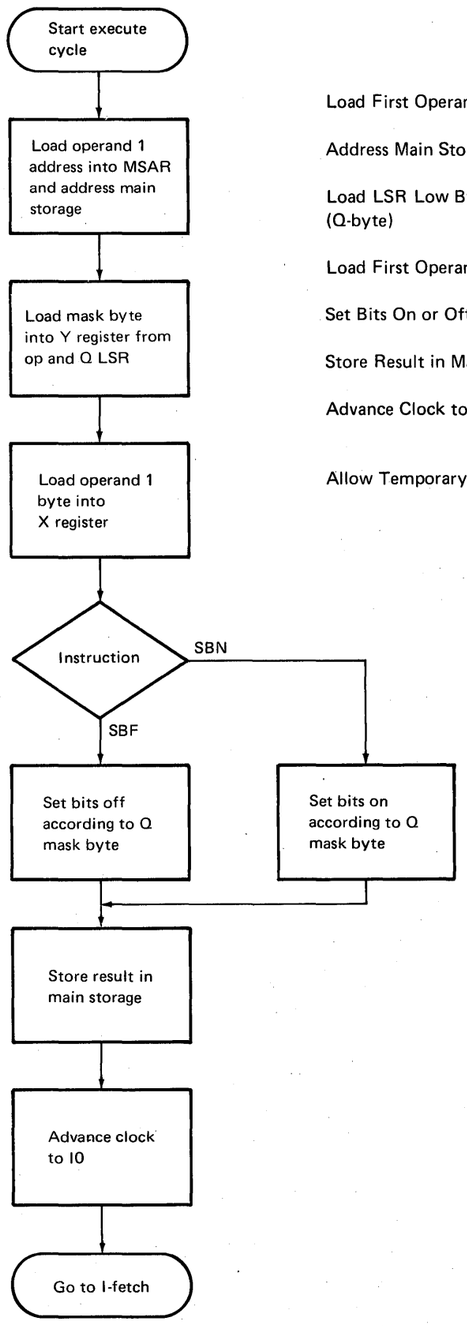
SET BITS OFF MASKED INSTRUCTION FORMAT

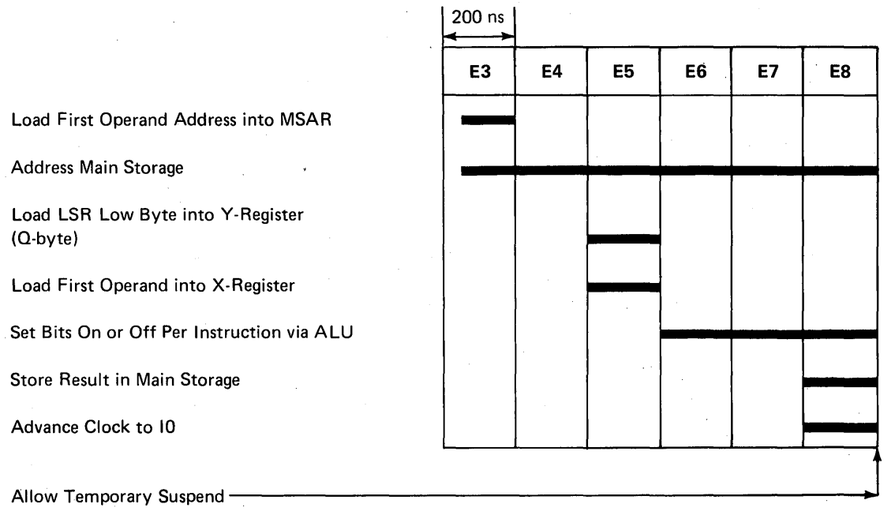
Operands	Op Code (hex)	Q-Byte[1] (binary)	Operand Address[2] (hex)	
	Byte 1	Byte 2	Byte 3	Byte 4
A1,I	3B	xxxx xxxx	Operand 1 address	
D1(,R1),I	7B	xxxx xxxx	Op 1 disp from XR1	
D1(,R2),I	BB	xxxx xxxx	Op 1 disp from XR2	

[1]The Q-byte contains a 1-byte binary mask specifying operand bits to be turned on.

[2]Operand 1 is a 1-byte field; operand 2 is not used.

Sequence and Timing





Store Register (ST)

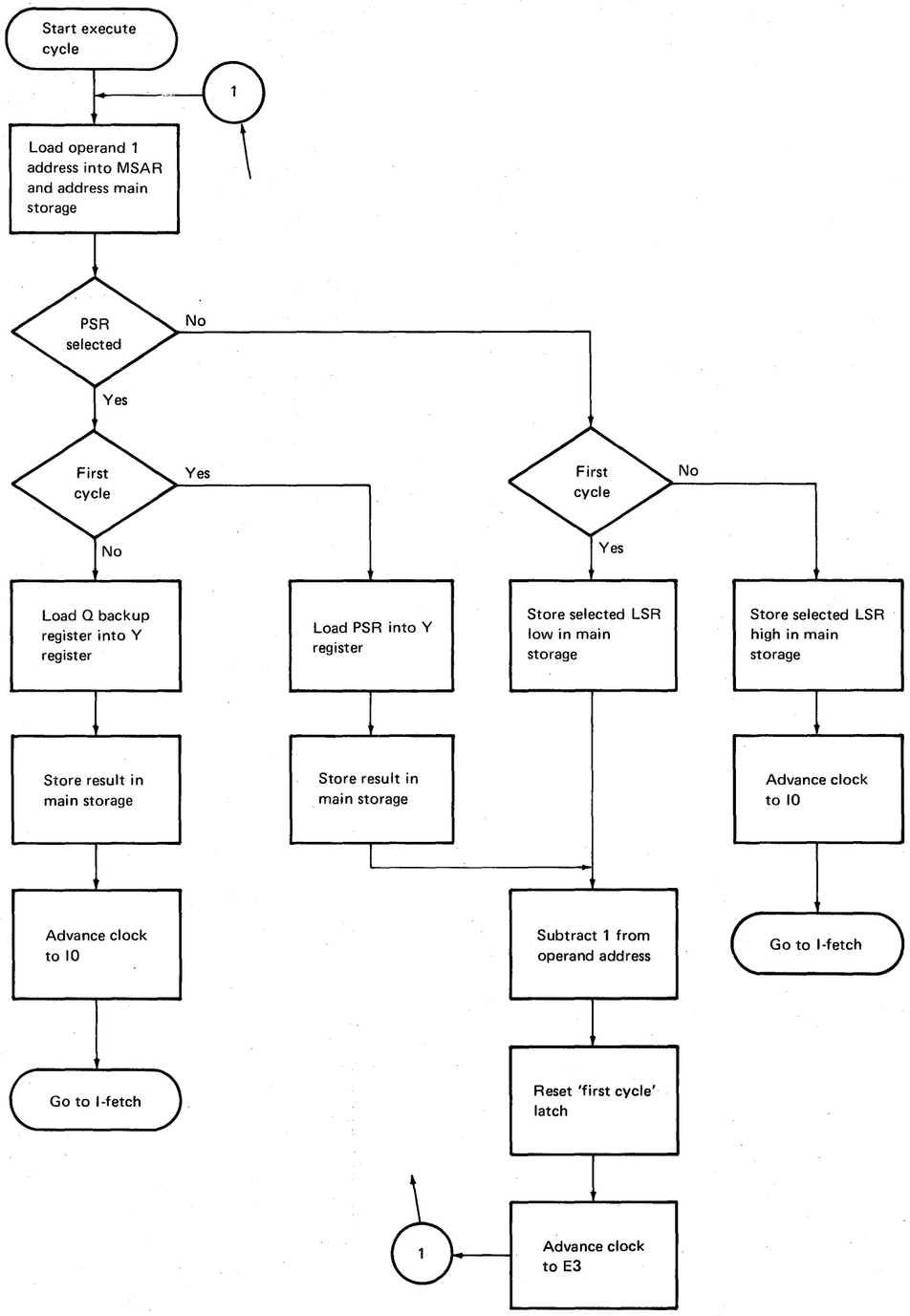
This instruction places the contents of the register specified by the Q-byte into the 2-byte field specified by the operand address. The Q-byte codes are:

Q-Byte (binary)	Q-Byte (hex)	Register Specified
0000 0000	00	None. The system ignores (no-ops) the instruction.
0000 0001	01	XR1.
0000 0010	02	XR2.
0000 0100	04	Program status register.
0000 1000	08	Address recall register.
0001 0000	10	Instruction address register.
0010 0000	20	Instruction address register.
0100 0000	40	Reserved; do not use.
1000 0000	80	Reserved; do not use.

Program Status Byte Settings

This instruction does not affect the program status register.

Sequence and Timing

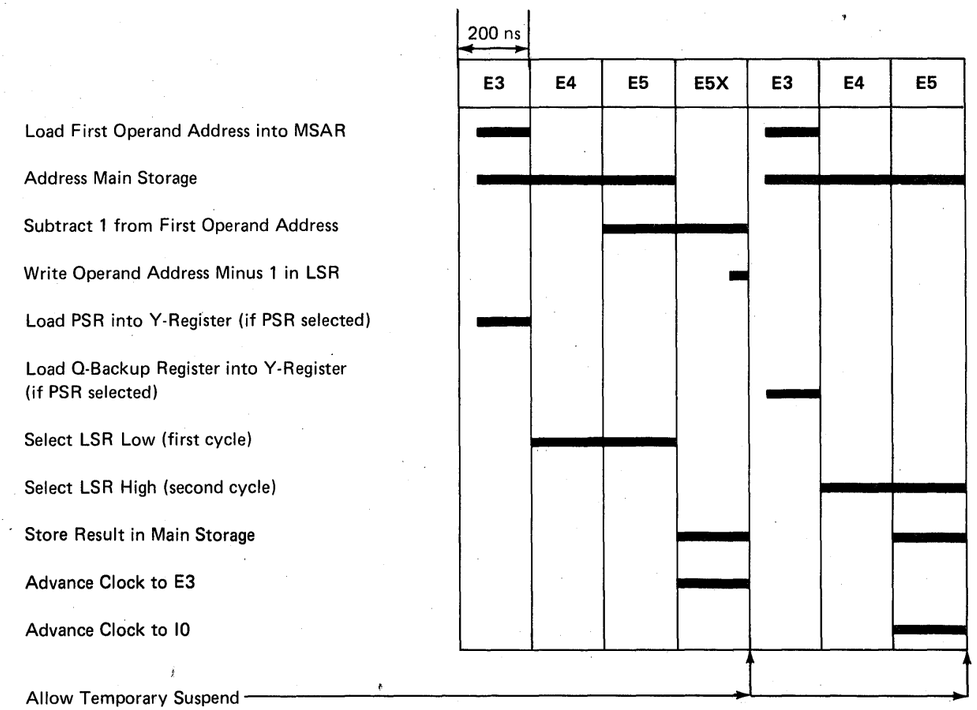


STORE REGISTER INSTRUCTION FORMAT

Operands	Op Code (hex)	Q-Byte[1] (binary)	Operand Address[2] (hex)	
	Byte 1	Byte 2	Byte 3	Byte 4
A1,RX	34	Rx	Operand 1 address	
D1(,R1),RX	74	Rx	Op 1 disp from XR1	
D1(,R2),RX	B4	Rx	Op 1 disp from XR2	

[1]Rx specifies the register whose contents are to be stored.

[2]Operand 1 is a 2-byte field addressed by its rightmost byte; operand 2 is not used.



Load Register (L)

This instruction moves data from the 2-byte field specified by the operand address into the register specified by the Q-byte. The Q-byte codes are:

Q-Byte (binary)	Q-Byte (hex)	Register Specified
0000 0000	00	None. The system ignores (no-ops) the instruction.
0000 0001	01	XR1.
0000 0010	02	XR2.
0000 0100	04	Program status register.
0000 1000	08	Address recall register.
0001 0000	10	Instruction address register.
0010 0000	20	Instruction address register.
0100 0000	40	Reserved; do not use.
1000 0000	80	Reserved; do not use.

Program Status Byte Settings

This instruction does not affect the program status register unless the instruction specifies the program status register.

The 6 rightmost bits (bits 10-15) of the program status register are condition indicators. These bits are named the *program status byte* throughout this manual. The other program status register bits are not used.

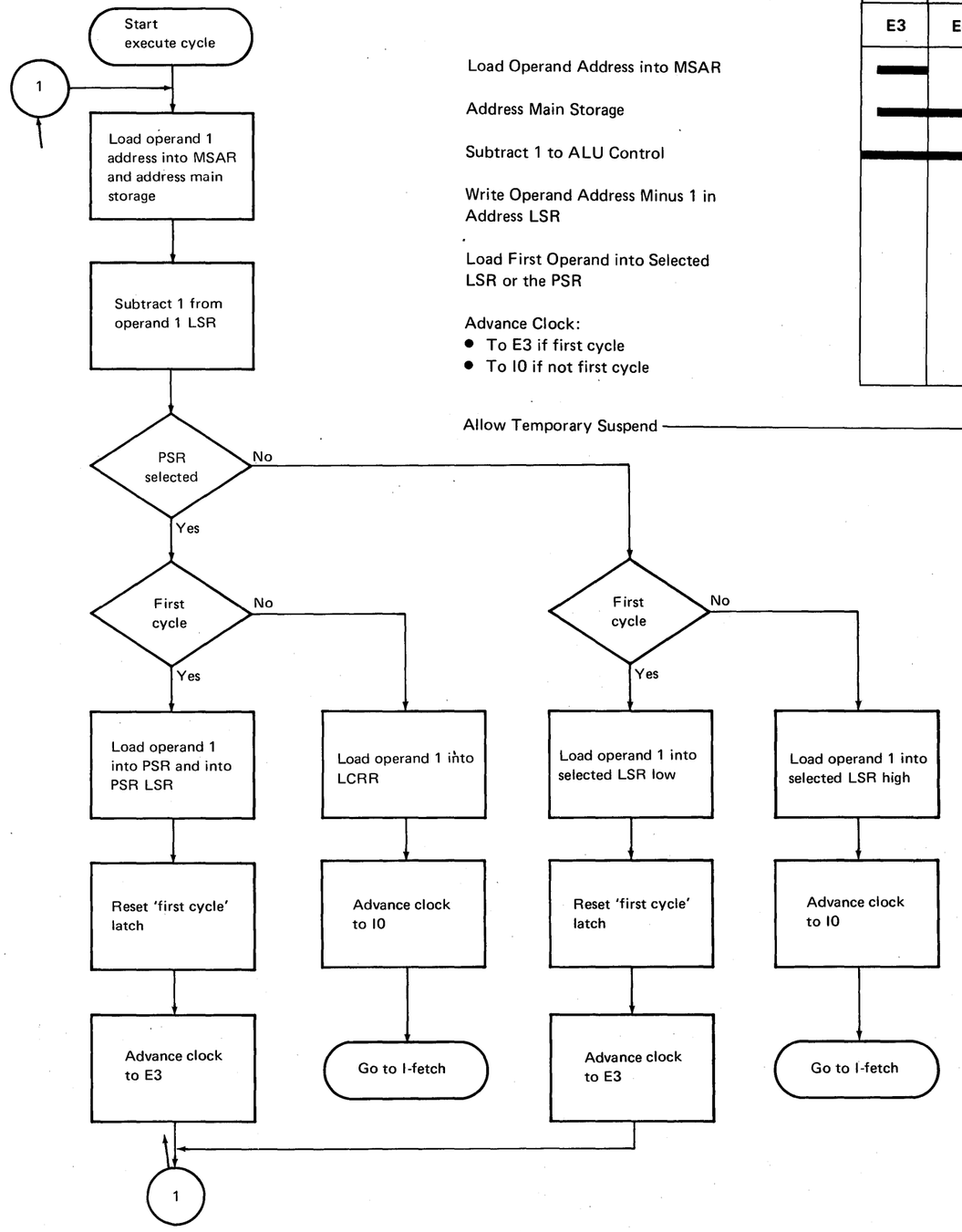
LOAD REGISTER INSTRUCTION FORMAT

Operands	Op Code (hex)	Q-Byte[1] (binary)	Operand Address[2] (hex)	
	Byte 1	Byte 2	Byte 3	Byte 4
A1,RX	35	Rx	Operand 1 address	
D1(,R1),RX	75	Rx	Op 1 disp from XR1	
D1(,R2),RX	B5	Rx	Op 1 disp from XR2	

[1]Rx specifies the register into which data is loaded.

[2]Operand 1 is a 2-byte field addressed by its rightmost byte; operand 2 is not used.

Sequence and Timing



Load Address (LA)

This instruction loads the value specified by instruction byte 3 or instruction bytes 3 and 4 into the index register specified by the Q-byte.

If an attempt is made to load both index registers at the same time (Q-byte bits 6 and 7 are both on), index register 1 is loaded; if neither index register is specified, index register 2 is loaded.

Program Status Byte Settings

This instruction does not affect the program status register.

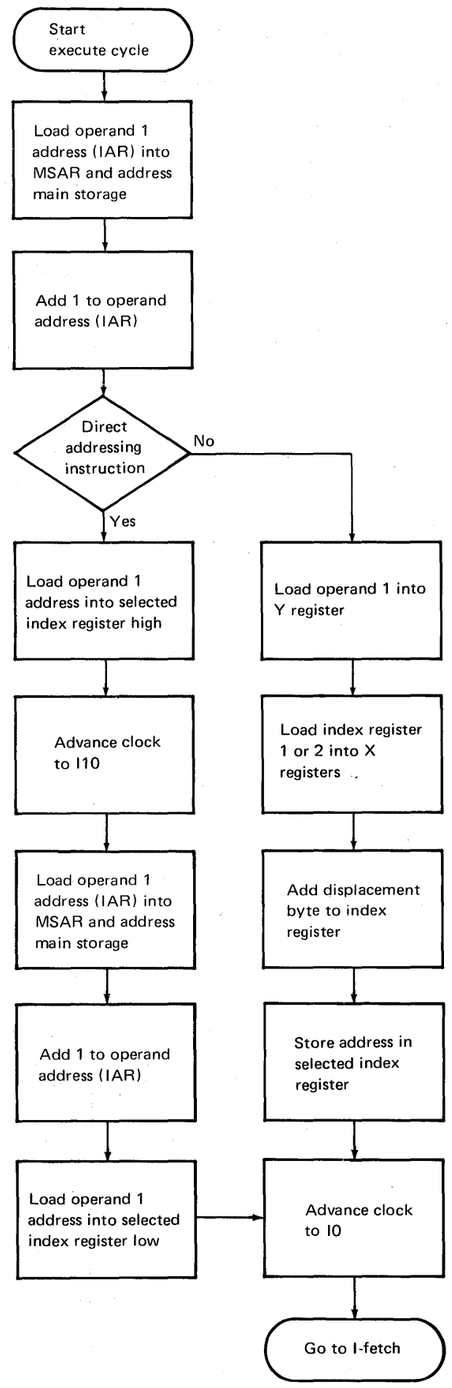
LOAD ADDRESS INSTRUCTION FORMAT

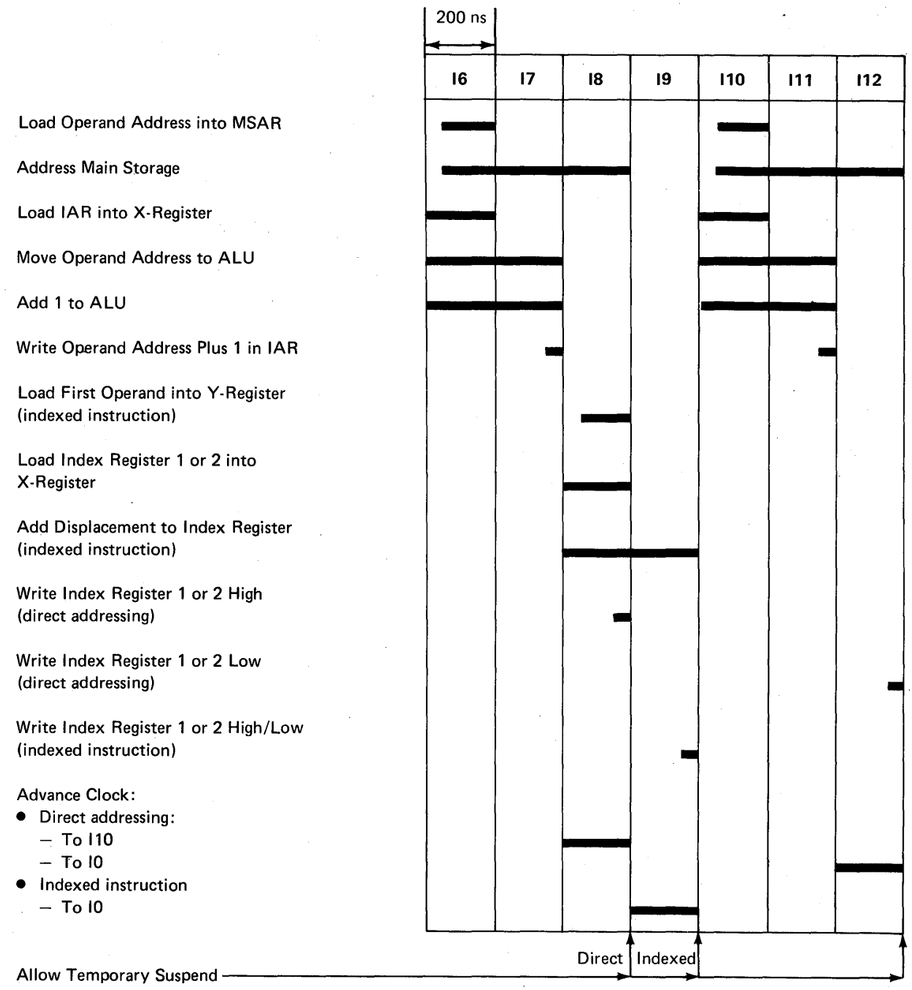
Operands	Op Code (hex)	Q-Byte[1] (binary)	Operand Address[2] (hex)	
	Byte 1	Byte 2	Byte 3	Byte 4
A1,RX	C2	Rx	Direct address	
D1(,R1),RX	D2	Rx	Op 1 disp from XR1	
D1(,R2),RX	E2	Rx	Op 1 disp from XR2	

[1]Rx specifies the index register to be loaded:
XR1 = hex 01 or 03
XR2 = hex 02 or 00

[2]A direct address is loaded when the machine instruction has a C2 op code. When the op code is D2, the system adds the machine instruction byte 3 value to the contents of XR1 and stores the result in the index register specified by the Q-byte. When the op code is E2, the system adds the machine instruction byte 3 value to the contents of XR2 and stores the result in the index register specified by the Q-byte.

Sequence and Timing





Logical Instructions

Compare Logical Immediate (CLI)

This instruction compares all the bits in the Q-byte with all the bits in operand 1 and indicates a high, low, or equal condition by setting a bit in the program status byte. Neither the Q-byte nor operand 1 is changed by this operation.

Program Status Byte Settings

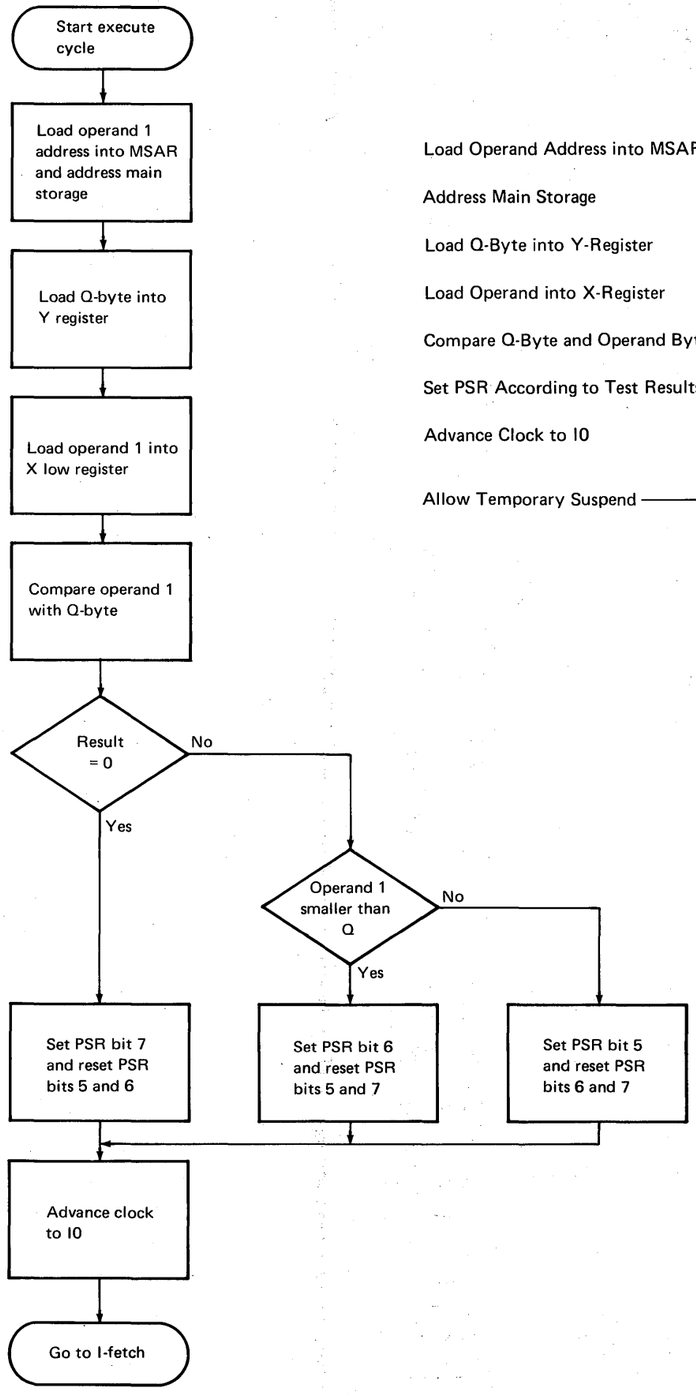
Bit	Name	Condition Indicated
7	Equal	Operand 1 value equal to Q byte value
6	Low	Operand 1 value less than Q byte value
5	High	Operand 1 value greater than Q byte value
4	Decimal overflow	Bit not affected
3	Test false	Bit not affected
2	Binary overflow	Bit not affected

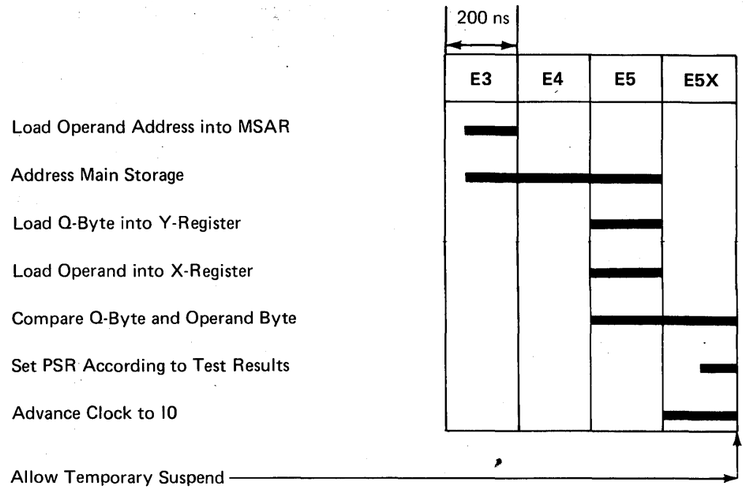
COMPARE LOGICAL IMMEDIATE INSTRUCTION FORMAT

Operands	Op Code (hex)	Q-Byte[1] (binary)	Operand Address[2] (hex)	
	Byte 1	Byte 2	Byte 3	Byte 4
A1,I	3D	I	Operand 1 address	
D1(,R1),I	7D	I	Op 1 disp from XR1	
D1(,R2),I	BD	I	Op 1 disp from XR2	

[1]I = 1 byte of immediate data (that is, 1 byte of actual data that is to be used in binary form).
[2]Operand 1 is a 1-byte field; operand 2 is not used.

Sequence and Timing





Compare Logical Characters (CLC)

This instruction compares operand 1 with operand 2, byte by byte, and sets the program status register to show the compared result. The compare mode uses each operand as a binary quantity; that is, similar bytes from the two operands are compared, bit for bit.

Program Status Byte Settings

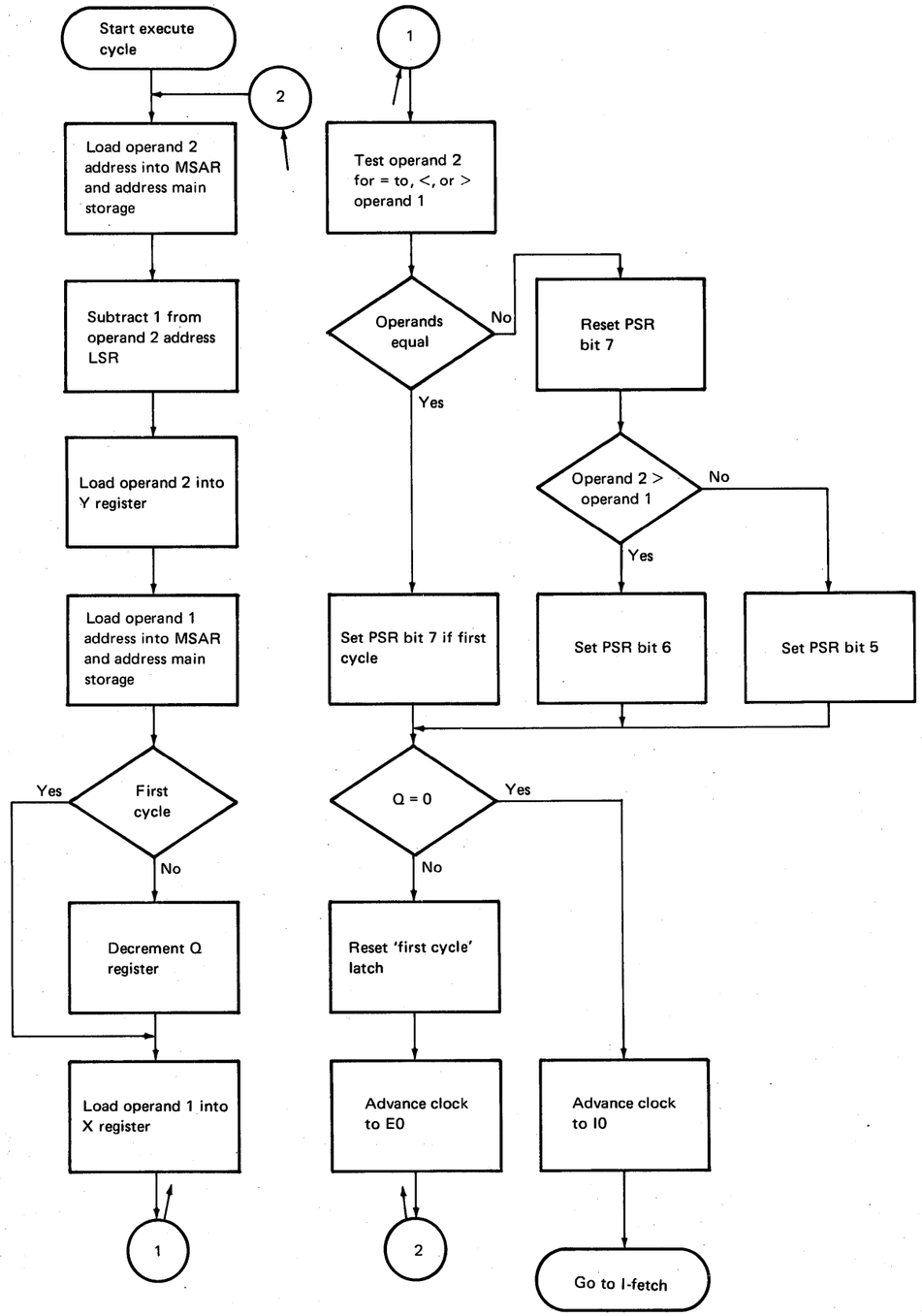
Bit	Name	Condition Indicated
7	Equal	Operand values are equal
6	Low	Operand 1 value smaller than operand 2 value
5	High	Operand 1 value greater than operand 2 value
4	Decimal overflow	Bit not affected
3	Test false	Bit not affected
2	Binary overflow	Bit not affected

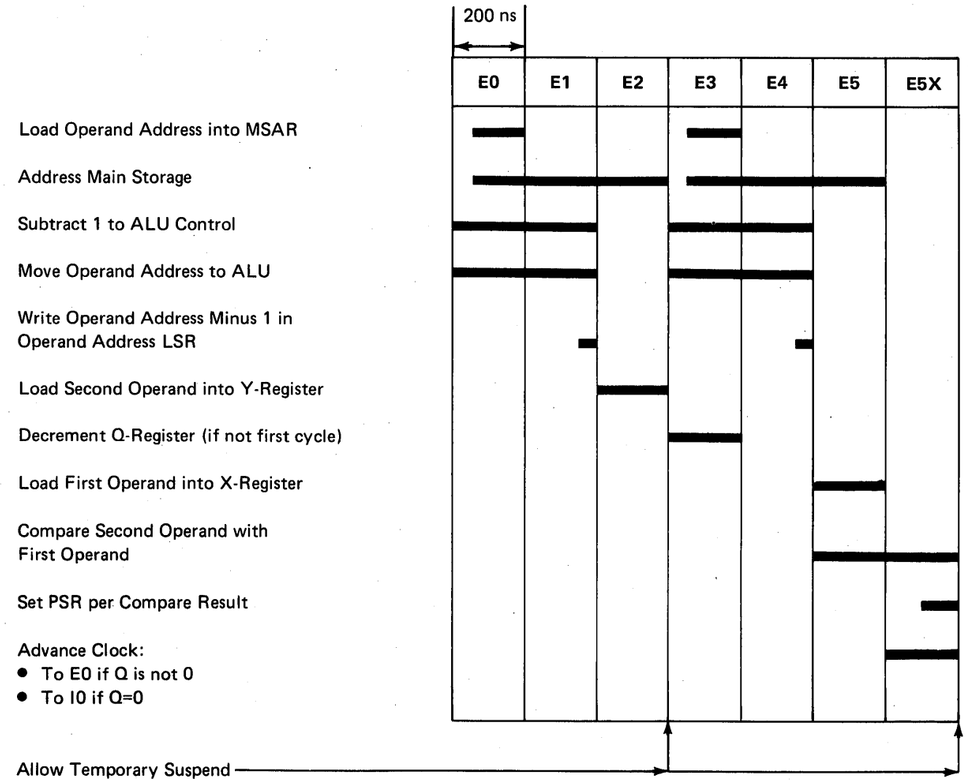
COMPARE LOGICAL CHARACTERS INSTRUCTION FORMAT

Operands	Op Code (hex)	Q Byte[1] (hex)	Operand Addresses[2] (hex)			
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
A1(L1),A2	0D	L1-1	Operand 1 address		Operand 2 address	
A1(L1),D2(,R1)	1D	L1-1	Operand 1 address		Op 2 disp from XR1	
A1(L1),D2(,R2)	2D	L1-1	Operand 1 address		Op 2 disp from XR2	
D1(L1,R1),A2	4D	L1-1	Op 1 disp from XR1	Operand 2 address		
D1(L1,R1),D2(,R1)	5D	L1-1	Op 1 disp from XR1	Op 2 disp from XR1		
D1(L1,R1),D2(,R2)	6D	L1-1	Op 1 disp from XR1	Op 2 disp from XR2		
D1(L1,R2),A2	8D	L1-1	Op 1 disp from XR2	Operand 2 address		
D1(L1,R2),D2(,R1)	9D	L1-1	Op 1 disp from XR2	Op 2 disp from XR1		
D1(L1,R2),D2(,R2)	AD	L1-1	Op 1 disp from XR2	Op 2 disp from XR2		

[1]The Q byte designates the operand length:
L1-1 = the number of bytes in either operand, minus 1.
Maximum length of each operand is 256 bytes; both operands must be the same length.
[2]The operands may overlap. Address operands by their rightmost bytes.

Sequence and Timing





Test Bits On Masked (TBN)

This instruction tests specified bits in the operand byte for a binary 1. For each mask bit (Q-byte bit) on, the system tests the same bit in the operand. If any tested bit is a 0, the system turns the test false indicator (in the program status register) on.

Program Status Byte Settings

Bit	Name	Condition Indicated
7	Equal	Bit not affected
6	Low	Bit not affected
5	High	Bit not affected
4	Decimal overflow	Bit not affected
3	Test false	One of the tested bits not on
2	Binary overflow	Bit not affected

TEST BITS ON MASKED INSTRUCTION FORMAT

Operands	Op Code (hex)	Q-Byte[1] (binary)	Operand Address[2] (hex)	
	Byte 1	Byte 2	Byte 3	Byte 4
A1,I	38	xxxx xxxx	Operand 1 address	
D1(,R1),I	78	xxxx xxxx	Op 1 disp from XR1	
D1(,R2),I	B8	xxxx xxxx	Op 1 disp from XR2	

[1]The Q-byte contains a 1-byte binary mask specifying operand bits for testing.

[2]Operand 1 is a 1-byte field; operand 2 is not used.

Test Bits Off Masked (TBF)

This instruction tests specified bits in the operand byte for a binary 1. For each mask bit (Q-byte bit) that is a 1, the system tests the same bit in the operand. If any tested bit is a 1, the system turns the test false indicator (in the program status register) on.

Program Status Byte Settings

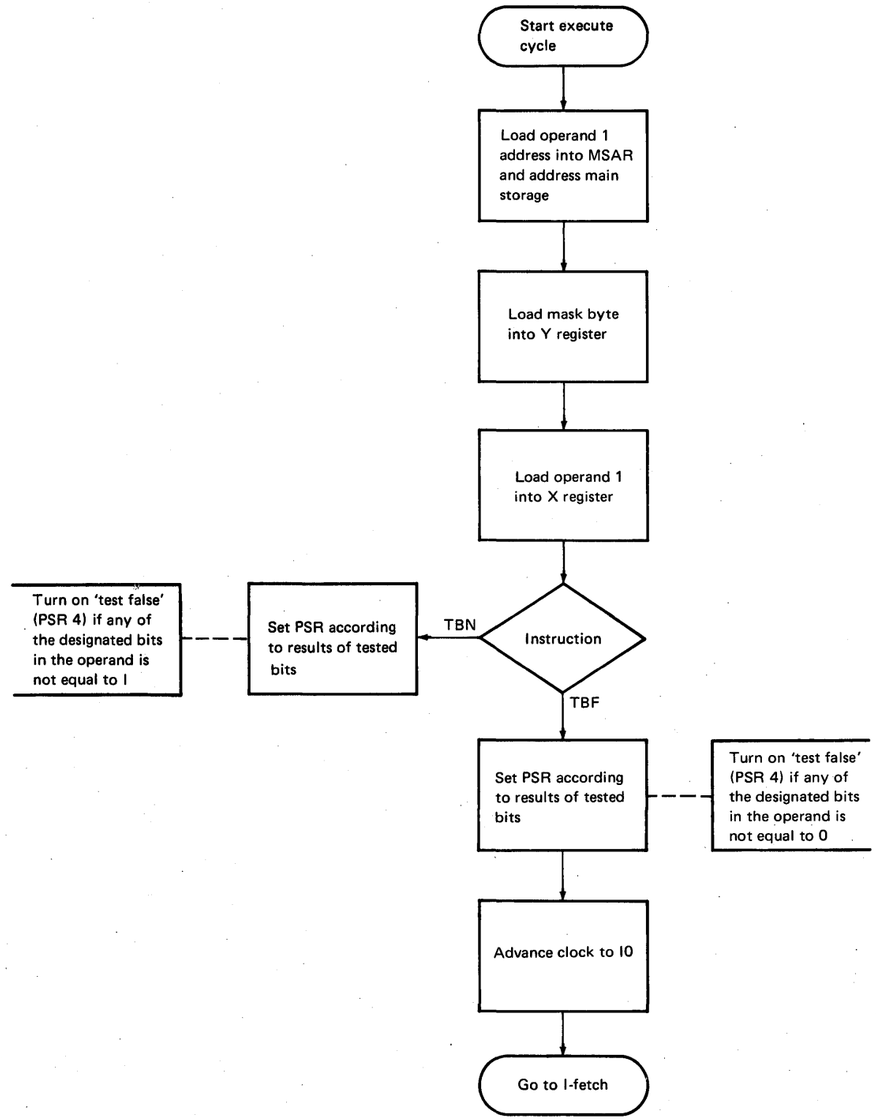
Bit	Name	Condition Indicated
7	Equal	Bit not affected
6	Low	Bit not affected
5	High	Bit not affected
4	Decimal overflow	Bit not affected
3	Test false	One of the tested bits on
2	Binary overflow	Bit not affected

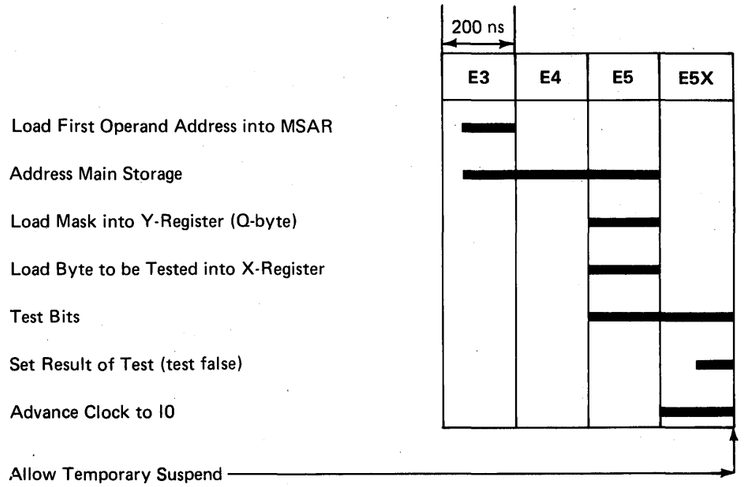
TEST BITS OFF MASKED INSTRUCTION FORMAT

Operands	Op Code (hex)	Q-Byte[1] (binary)	Operand Address[2] (hex)	
	Byte 1	Byte 2	Byte 3	Byte 4
A1,I	39	xxxx xxxx	Operand 1 address	
D1(,R1),I	79	xxxx xxxx	Op 1 disp from XR1	
D1(,R2),I	B9	xxxx xxxx	Op 1 disp from XR2	

[1]The Q-byte contains a 1-byte binary mask specifying operand bits for testing.

[2]Operand 1 is a 1-byte field; operand 2 is not used.


Start execute cycle
Load operand 1 address into MSAR and address main storage
Load mask byte into Y register
Load operand 1 into X register
Instruction
TBN
Set PSR according to results of tested bits
Turn on 'test false' (PSR 4) if any of the designated bits in the operand is not equal to 1
TBF
Set PSR according to results of tested bits
Turn on 'test false' (PSR 4) if any of the designated bits in the operand is not equal to 0
Advance clock to I0
Go to I-fetch


200 ns
E3
E4
E5
E5X
Load First Operand Address into MSAR
Address Main Storage
Load Mask into Y-Register (Q-byte)
Load Byte to be Tested into X-Register
Test Bits
Set Result of Test (test false)
Advance Clock to I0
Allow Temporary Suspend

Branch on Condition (BC)

This instruction, under control of the Q-byte, tests the rightmost byte of the program status register. If the register verifies the condition specified by the Q-byte, the system places the address of the next sequential instruction in the address recall register, places the branch-to address in the instruction address register, and branches to the branch-to address. If the register does not verify at least one condition specified by the Q-byte, the system places the address of the next sequential instruction in the instruction address register, and the program advances to the next sequential instruction.

The Q-byte specifies which conditions are tested and if the branch is to occur on condition true (program status register bit is 1) or condition false (program status register bit is 0).

Bits 2 through 7 of the Q-byte specify which bits of the program status register's byte are to be tested. These bits, and the conditions they represent, are:

Bit	Condition Tested
0	If 1, branch if *any* condition tested is true. If 0, branch if *all* conditions tested are false.
1	None (bit should be set to 0)
2	Binary overflow
3	Test false
4	Decimal overflow
5	High
6	Low
7	Equal

When bit 0 of the Q-byte is 1 (condition true), the branch occurs if any condition tested is 1. When bit 0 of the Q-byte is 0 (condition false), the branch occurs if all conditions tested are 0.

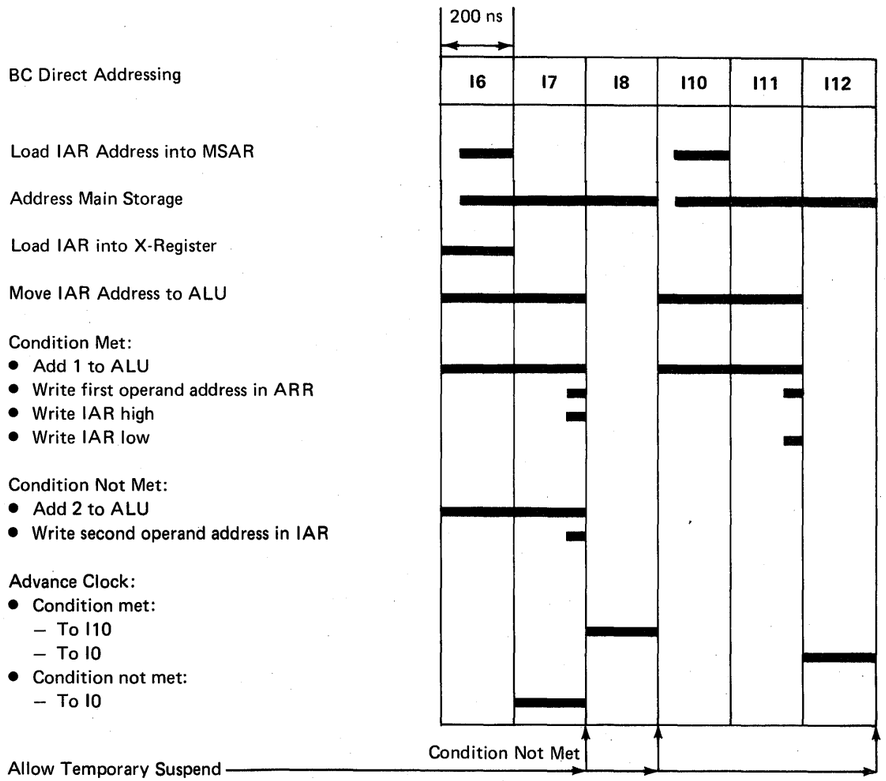
Program Status Byte Settings

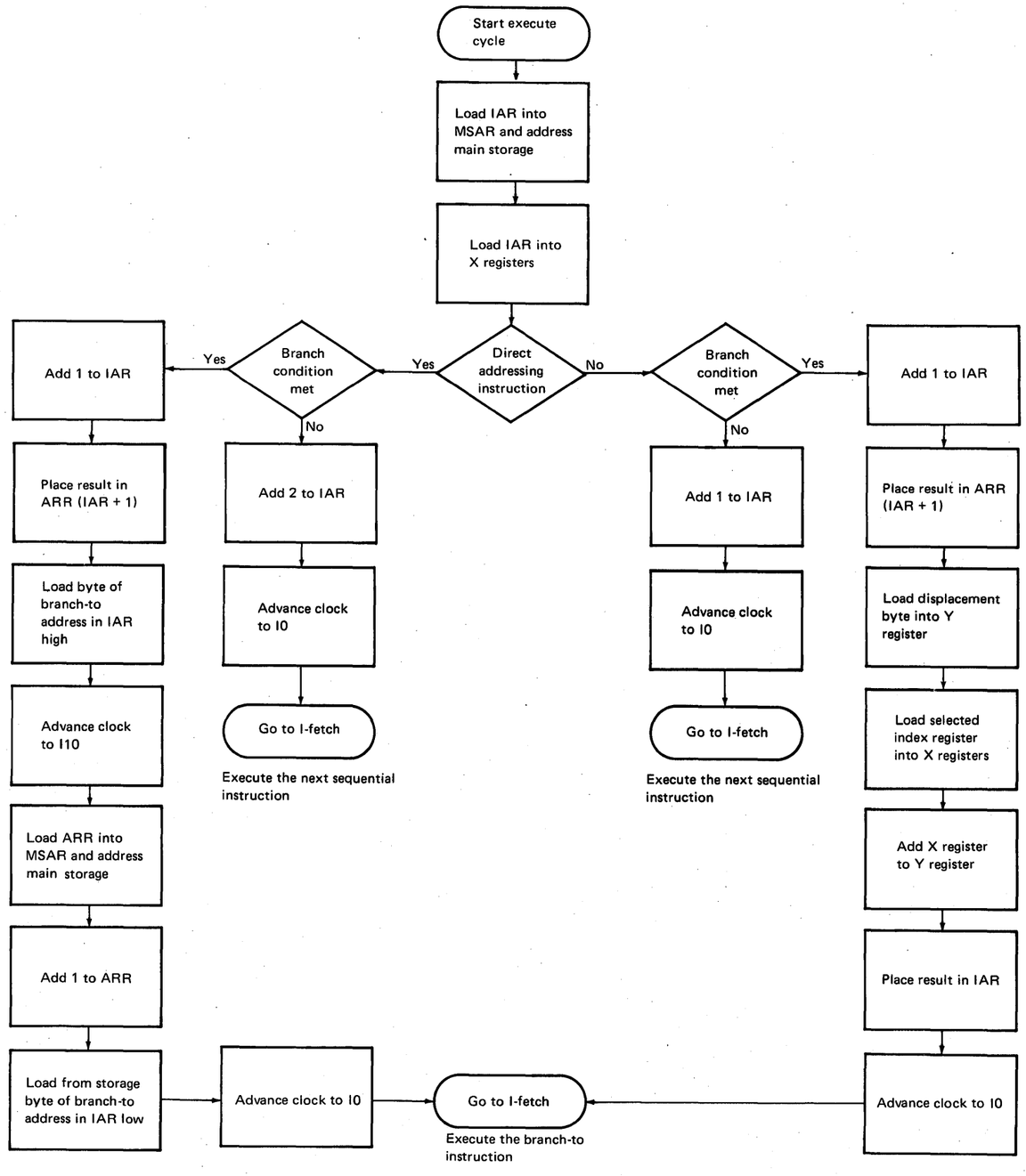
Bit	Name	Condition Indicated
7	Equal	Bit not affected
6	Low	Bit not affected
5	High	Bit not affected
4	Decimal overflow	Turned off if tested; otherwise not affected
3	Test false	Turned off if tested; otherwise not affected
2	Binary overflow	Bit not affected

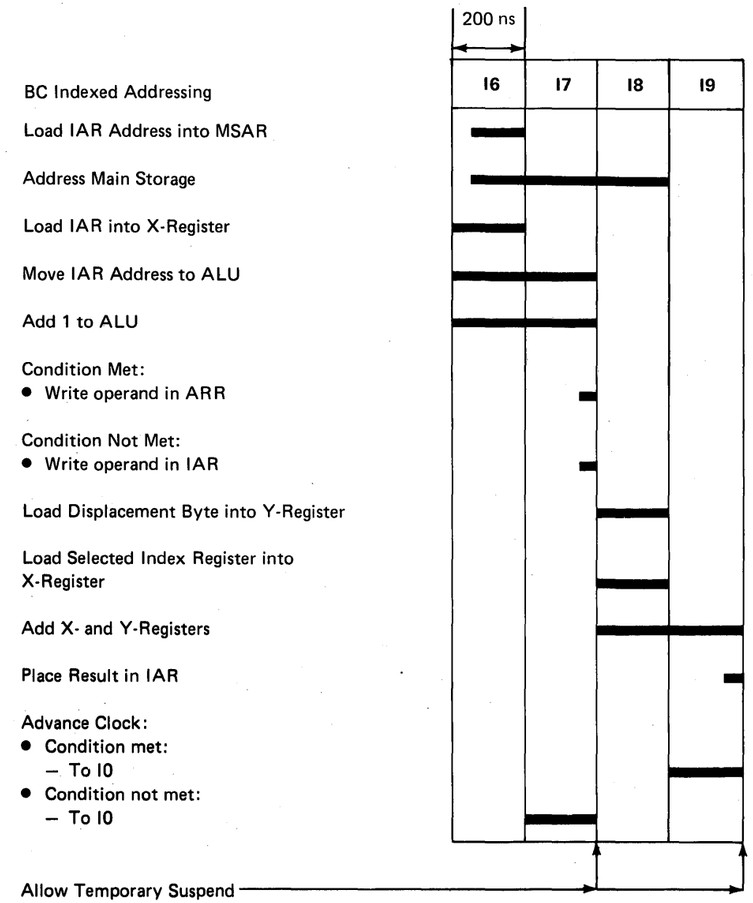
BRANCH ON CONDITION INSTRUCTION FORMAT

Operands	Op Code (hex)	Q Byte[1] (binary)	Branch To Address (hex)	
	Byte 1	Byte 2	Byte 3	Byte 4
A1,I	C0	xxxx xxxx	Direct address	
D1(,R1),I	D0	xxxx xxxx	Disp from XR1	
D1(,R2),I	E0	xxxx xxxx	Disp from XR2	

[1]The Q byte contains a binary mask specifying which program status register positions are tested by the instruction.




Start execute cycle
Load IAR into MSAR and address main storage
Load IAR into X registers
Direct addressing instruction
Yes
No
Branch condition met
Yes
No
Add 1 to IAR
Add 2 to IAR
Advance clock to I0
Go to I-fetch
Execute the next sequential instruction
Branch condition met
Yes
No
Add 1 to IAR
Advance clock to I0
Go to I-fetch
Execute the next sequential instruction
Add 1 to IAR
Place result in ARR (IAR + 1)
Load displacement byte into Y register
Load selected index register into X registers
Add X register to Y register
Place result in IAR
Advance clock to I0
Place result in ARR (IAR + 1)
Load byte of branch-to address in IAR high
Advance clock to I10
Load ARR into MSAR and address main storage
Add 1 to ARR
Load from storage byte of branch-to address in IAR low
Advance clock to I0
Go to I-fetch
Execute the branch-to instruction


200 ns
I6
I7
I8
I9
BC Indexed Addressing
Load IAR Address into MSAR
Address Main Storage
Load IAR into X-Register
Move IAR Address to ALU
Add 1 to ALU
Condition Met:
• Write operand in ARR
Condition Not Met:
• Write operand in IAR
Load Displacement Byte into Y-Register
Load Selected Index Register into X-Register
Add X- and Y-Registers
Place Result in IAR
Advance Clock:
• Condition met:
– To I0
• Condition not met:
– To I0
Allow Temporary Suspend

Command Instructions

Jump on Condition (JC)

JUMP ON CONDITION INSTRUCTION FORMAT

Operand	Op Code (hex)	Q-Byte[1] (hex)	R-Byte[2] (hex)
	Byte 1	Byte 2	Byte 3
A1,I	F2	xxxx xxxx	IAR disp

[1]The Q-byte contains a binary mask that indicates which status register bits (the bits in the rightmost byte of the program status register) are tested by the machine instruction.

[2]The R-byte is a displacement which, when added to the address in the machine instruction address register, provides a jump-to address.

This instruction, under control of the Q-byte, tests the program status register. If the program status register verifies the conditions specified by the Q-byte, the system adds the value stored in byte 3 of the instruction to the contents of the instruction address register and stores the result in the instruction address register. The new address stored in the instruction address register at the end of the jump-on-condition operation fetches the next instruction. If the register does not verify the condition(s) specified by the Q-byte, the system advances to the next sequential instruction in the program. The Q-byte specifies which conditions are tested and if the jump is to occur on condition true (program status register bit is 1) or condition false (program status register bit is 0).

Bits 2 through 7 of the Q-byte specify the program status byte to be tested. These bits, and the conditions they represent, are:

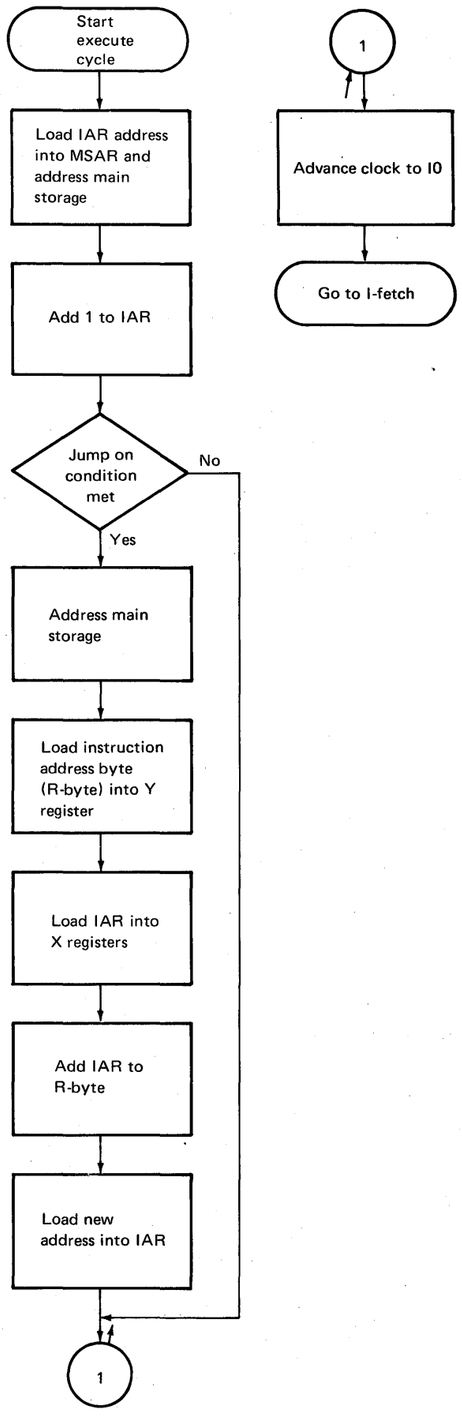
Bit	Condition Tested
1	None (bit should be set to 0)
2	Binary overflow
3	Test false
4	Decimal overflow
5	High
6	Low
7	Equal

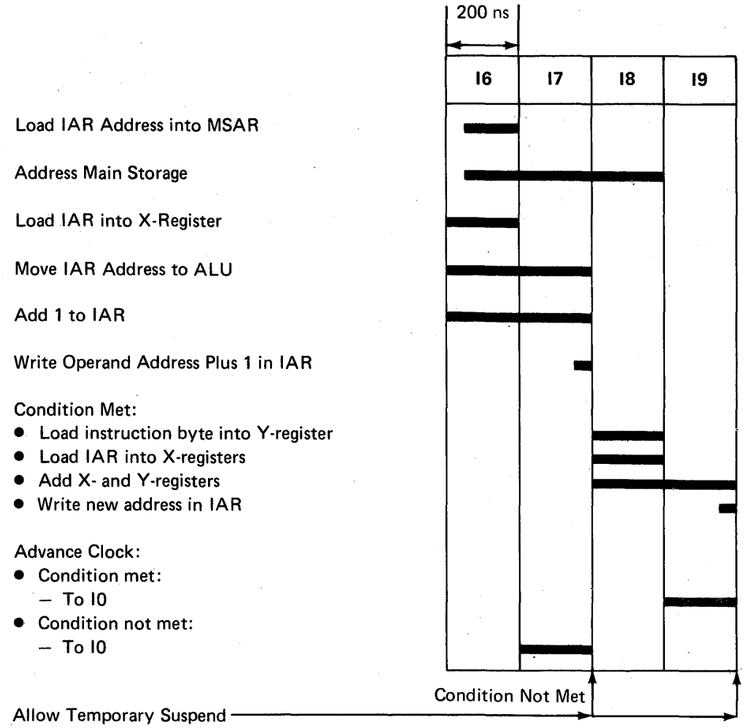
When bit 0 of the Q-byte is 1 (condition true), the jump occurs if any indicator tested is on (associated bit is 1). When bit 0 of the Q-byte is 0 (condition false), the jump occurs if all indicators tested are off (all associated bits are 0).

Program Status Byte Settings

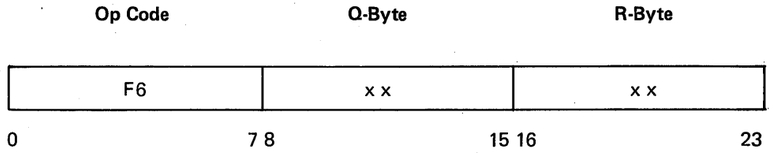
Bit	Name	Condition Indicated
7	Equal	Bit not affected
6	Low	Bit not affected
5	High	Bit not affected
4	Decimal overflow	Turned off if tested; otherwise not affected
3	Test false	Turned off if tested; otherwise not affected
2	Binary overflow	Bit not affected

Sequence and Timing





Load Program Mode Register (LPMR)



The Q-byte must have good parity.

The R-byte is the value to be put into the program mode register.

The main storage processor instruction changes the program mode register with the value of the R-byte. This instruction needs much less time and does not need to wait for service such as an SVC instruction.

Program Status Byte Settings

This instruction does not affect the program status register.

Supervisor Call (SVC)

Operand	Op Code (hex)	Q-Byte (hex)	R-Byte (hex)
	Byte 1	Byte 2	Byte 3
I1,RX	F4	xx	00

This instruction sets control processor interrupt level 5. The main storage processor clock is stopped and remains stopped until started again by the control processor.

Bits in the Q-byte specify how the function requested by the R-byte is to be used. Following is a chart of the Q-byte bits and their operation:

Bit	Operation
0	Control storage supervisor call indicator
1	Not used
2	Hold dispatch indicator
3	Nonquiescing request indicator
4	More than one unit I/O request, or asynchronous error wait request
5	Translate off–Input/output block/parameter list indicator
6	Nonrefreshable supervisor call indicator
7	Wait on this supervisor call indicator; or refresh transient/transfer indicator

The R-byte specifies which control processor function to do. (See the *Control Storage Logic Manual*.) A control processor routine, executed on a level 5 interrupt, checks the information in the Q- and R-bytes to determine if the control processor will (1) execute the function immediately on a level 5 interrupt, or (2) store the information for later processing on the control processor main program level. The R-byte is stored in the length count recall register (LCRR) of the local storage register stack for use by the control processor. A control processor level 5 interrupt is generated during main storage instruction fetch. When the control processor is executing the level 5 interrupt program routine, it looks at the main storage processor status byte 2 bit 0 (nonexecutable bit) to determine if it needs the LCRR information to process the nonexecutable instruction. (See the flowchart under *Instruction Fetch Operation* earlier in this section.)

Two functions of the SVC instruction can be shown here:

1. R-byte = hex 04; transfer control or system transient now has priority.
2. R-byte = hex 0D: Change the program mode register (similar to the LPMR instruction).

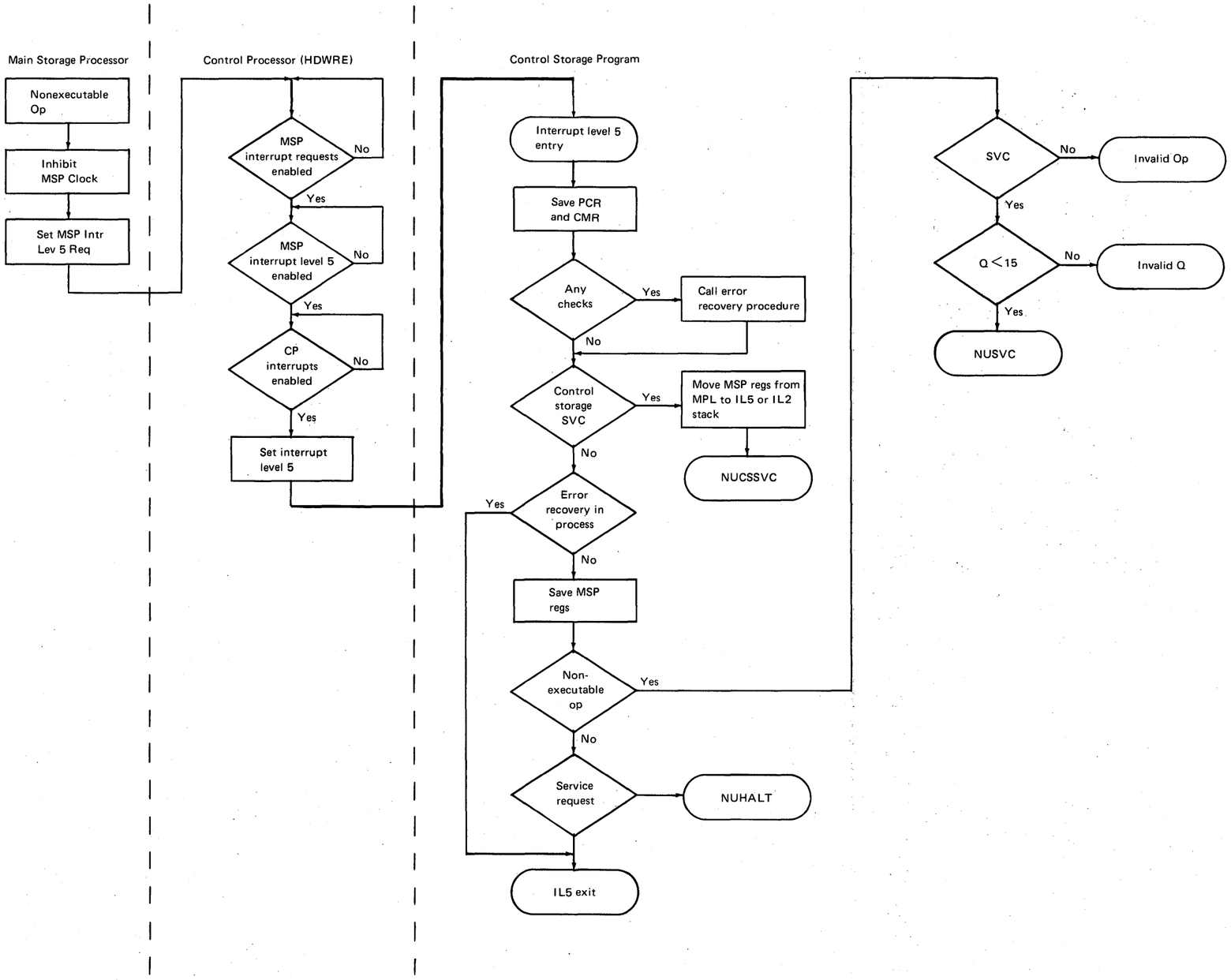
Example of an SVC instruction and use of the R-byte:

SVC

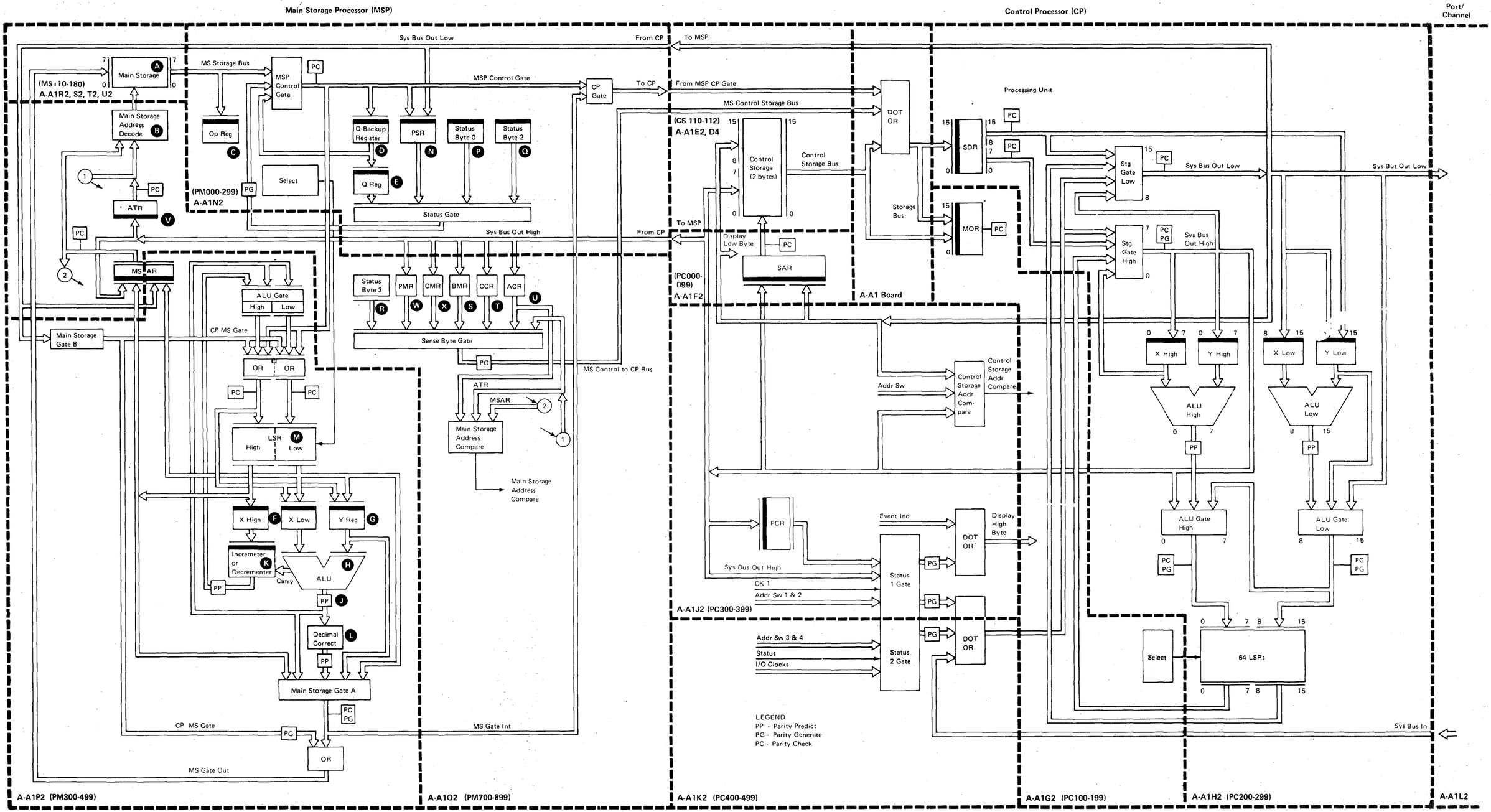
Op Code	Q-Byte	R-Byte	
F4		'01'	
		....	
		'04'	Transfer control/system transient
		....	
(See the *Control Storage Logic Manual*.)		'0D'	Change program mode register
		....	
		....	
		'5F'	

The program mode register can be changed by a main storage processor SVC instruction to the control processor. A control processor interrupt level 5 then causes a control processor program routine to change the program mode register with a load or sense main storage processor register instruction (WMPR or RMPR).

Supervisor Call Operation
(hardware/control storage program)



FUNCTIONAL UNITS



*Data flow bus lines may not pass through FRUs as shown

Main Storage (A)

Main storage contains 32K, 48K, 64K, 96K, or 128K addresses; each address is 1 byte wide. User programs and the SSP programs are loaded in main storage.

Main Storage Address Register (B)

The main storage address register (MSAR) is a 16-bit register that addresses main storage. This register is loaded from one of the main storage local storage registers or from the control processor. The output from the main storage address register goes to the main storage address decode logic. The decode logic selects the addressed byte in main storage. When address translation is needed, the 5 high-order bits are used to select a register from the address translation register stack. The control processor loads MSAR to load or sense the main storage processor register with the WMPR or RMPR instructions.

Operation Register (C)

The operation register (op reg) is an 8-bit register that holds each system instruction as it is fetched from main storage. The output is decoded and then gates the arithmetic and logic unit operations, program status register setting, local storage register selection, and main storage processor clock controls.

Q-Backup Register (D)

The Q-backup register is an 8-bit register that holds the original Q-byte obtained from main storage. The output of this register is gated to the Q-register. The Q-backup register is necessary to perform a recomplement cycle using the zero and add zoned, add zoned decimal, and subtract zoned decimal instructions.

Q-Register (E)

The Q-register is an 8-bit register that specifies the length of the operands for arithmetic and logic unit operations. The Q-register also extends or changes the operation code. During instruction execution, the Q-byte is stored in the Q-register. The Q-register is loaded from the Q-backup register. The decoded output of the Q-register controls instruction execution. The contents of the Q-register are also stored in the Q-byte of the Op-Q local storage register during instruction fetch.

X-Registers (F)

These 8-bit registers are the buffer input to the arithmetic and logic unit (ALU) and the incrementer or decrementer. The X-low register is input to the ALU low; the X-high register is input to the incrementer or decrementer. The input to the X-low register is from the main storage processor storage or from the selected local storage register low. The input to the X-high register is from the selected local storage register high only.

Y-Register (G)

This 8-bit register is the buffer input to the arithmetic and logic unit (ALU) low. The output can bypass the ALU and be gated to main storage. The input is from the main storage processor storage or the local storage registers.

Arithmetic and Logic Unit (H)

The arithmetic and logic unit (ALU) operates on bits 8-15 of a 1- or 2-operand instruction when working with a 1- or 2-byte data field. The ALU performs the following logical operations:

- X OR (not) Y
- AND (not) Y
- AND Y
- OR Y

The ALU also performs the following binary arithmetic operations:

- X-Y carry out
- X-1 carry in
- X+Y carry in
- X+1 carry out

The input to the ALU is from the X-low register and the Y-register. The X-low register contains a 1-byte field or the low-order byte of a 2-byte field for one operand. The Y-register contains a 1-byte field for the other operand to be used in the ALU operation. The output of the ALU is to either the decimal correct logic or the ALU gate. From the ALU gate, the data can be gated into a high- or low-byte position of a selected local storage register. The output from the ALU can also be gated to main storage through main storage gate A.

Arithmetic and Logic Unit Parity Predict (J)

Parity predict circuits calculate the parity of the result of the arithmetic and logic unit operation. This calculated parity is compared against the generated parity. If there is a difference, a parity check occurs.

Incrementer or Decrementer (K)

The incrementer or decrementer operates on bits 0-7 of a single byte during an operation using a 2-byte operand. The incrementer or decrementer is a counter that increases or decreases its contents by 1 when a carry from the ALU occurs. The incrementer or decrementer can also pass a single byte of data. The operation being performed determines if the contents are increased or decreased. Input to the incrementer or decrementer is from the high byte of the selected local storage register through the X-high register; output is to the ALU gate.

Decimal Correct (L)

The decimal correct circuits contain the logic necessary to convert the ALU output of a decimal operation to the correct decimal result. The result can be a value from 0 through 9 or from hexadecimal A through hexadecimal F. Because the hexadecimal system is used to add decimal numbers (binary addition), the result is not always the real value. The result of a decimal digit arithmetic operation may, therefore, be 6 less than the real value. Six is added to the result if a carry occurs after a logical addition of two numbers or if the result is from hexadecimal A through hexadecimal F. For example, if 5 is added to 9, the result from the ALU is hexadecimal E.

```
1111 0101
1111 1001
---------

1111 1110 Result is hexadecimal E
```

In this case, the result is not a decimal number. The result must be converted to a value between 0 and 9. When using the hexadecimal system, the value hexadecimal F is 6 more than 9. To perform the decimal correction, 6 is added to the value in the decimal correct logic and there is a carry of 1. This carry is then added to the next 2 bytes that are added together. Because of the result of the addition, adding 6 gives the correct value as shown:

```
1111 0000 1111 1110
1111 0000 1111 0110
-------------------
1111 0001 1111 0100 = hexadecimal 4
                      with a carry result
                      equals 14
```

The following tables show the output from the decimal correct logic after decimal adding or decimal subtracting. If decimal arithmetic specifies values from hexadecimal A through hexadecimal F, the output from the decimal correction values is also shown. Note that the output from the decimal correct logic is changed by the values of operand 1 and operand 2.

DECIMAL CORRECT FOR DECIMAL ADDING

Op 1 plus (Op 2 or carry in)

Op 2/ carry in	Op 1 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	0c	1c	2c	3c	4c	5c
1	1	2	3	4	5	6	7	8	9	0c	1c	2c	3c	4c	5c	6c
2	2	3	4	5	6	7	8	9	0c	1c	2c	3c	4c	5c	6c	7c
3	3	4	5	6	7	8	9	0c	1c	2c	3c	4c	5c	6c	7c	8c
4	4	5	6	7	8	9	0c	1c	2c	3c	4c	5c	6c	7c	8c	9c
5	5	6	7	8	9	0c	1c	2c	3c	4c	5c	6c	7c	8c	9c	Ac
6	6	7	8	9	0c	1c	2c	3c	4c	5c	6c	7c	8c	9c	Ac	Bc
7	7	8	9	0c	1c	2c	3c	4c	5c	6c	7c	8c	9c	Ac	Bc	Cc
8	8	9	0c	1c	2c	3c	4c	5c	6c	7c	8c	9c	Ac	Bc	Cc	Dc
9	9	0c	1c	2c	3c	4c	5c	6c	7c	8c	9c	Ac	Bc	Cc	Dc	Ec
A	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9
B	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	0c
C	C	D	E	F	0	1	2	3	4	5	6	7	8	9	0c	1c
D	D	E	F	0	1	2	3	4	5	6	7	8	9	0c	1c	2c
E	E	F	0	1	2	3	4	5	6	7	8	9	0c	1c	2c	3c
F	F	0	1	2	3	4	5	6	7	8	9	0c	1c	2c	3c	4c
F + 1	0	1	2	3	4	5	6	7	8	9	0c	1c	2c	3c	4c	5c

DECIMAL CORRECT FOR DECIMAL SUBTRACTING

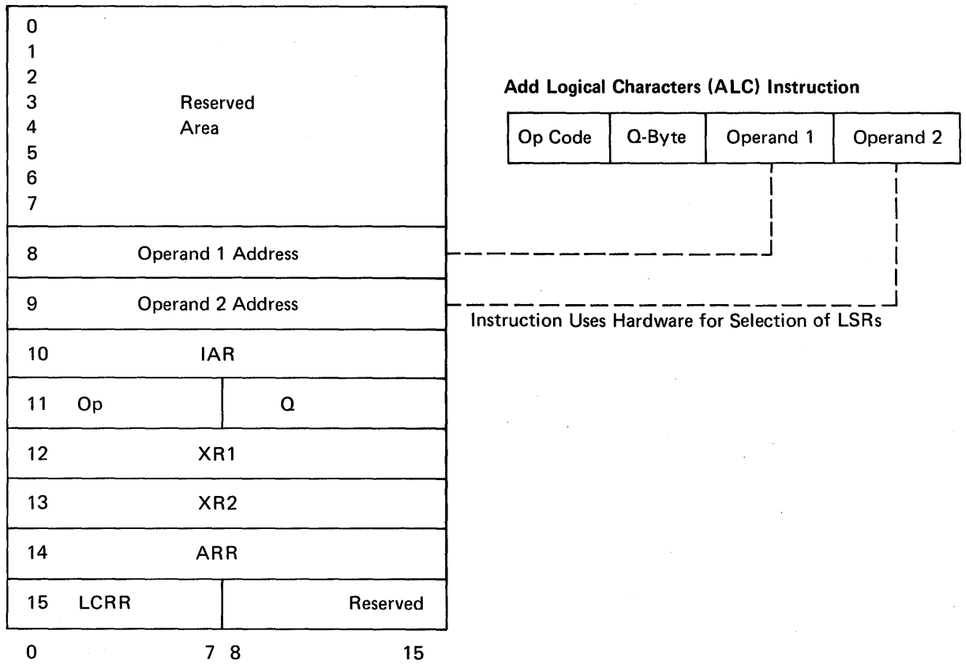
Op 1 minus (Op 2 or borrow in)

Op 2/ borrow in	Op 1 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	9b	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
2	8b	9b	0	1	2	3	4	5	6	7	8	9	A	B	C	D
3	7b	8b	9b	0	1	2	3	4	5	6	7	8	9	A	B	C
4	6b	7b	8b	9b	0	1	2	3	4	5	6	7	8	9	A	B
5	5b	6b	7b	8b	9b	0	1	2	3	4	5	6	7	8	9	A
6	4b	5b	6b	7b	8b	9b	0	1	2	3	4	5	6	7	8	9
7	3b	4b	5b	6b	7b	8b	9b	0	1	2	3	4	5	6	7	8
8	2b	3b	4b	5b	6b	7b	8b	9b	0	1	2	3	4	5	6	7
9	1b	2b	3b	4b	5b	6b	7b	8b	9b	0	1	2	3	4	5	6
A	0b	1b	2b	3b	4b	5b	6b	7b	8b	9b	0	1	2	3	4	5
B	Fb	0b	1b	2b	3b	4b	5b	6b	7b	8b	9b	0	1	2	3	4
C	Eb	Fb	0b	1b	2b	3b	4b	5b	6b	7b	8b	9b	0	1	2	3
D	Db	Eb	Fb	0b	1b	2b	3b	4b	5b	6b	7b	8b	9b	0	1	2
E	Cb	Db	Eb	Fb	0b	1b	2b	3b	4b	5b	6b	7b	8b	9b	0	1
F	Bb	Cb	Db	Eb	Fb	0b	1b	2b	3b	4b	5b	6b	7b	8b	9b	0
F + 1	Ab	Bb	Cb	Db	Eb	Fb	0b	1b	2b	3b	4b	5b	6b	7b	8b	9b

Local Storage Register (M)

The local storage register stack contains sixteen 2-byte local storage registers (LSRs); the first eight are reserved and the last eight are used as data buffers and address registers for main storage. Main storage processor addressing is controlled by both hardware and instructions. Selecting a register from the local storage register stack is under direct hardware control as a function of the instruction format, as shown in the following example:

Main Storage Processor Local Storage Register



Program Status Register (N)

The program status register (PSR) contains the main storage processor conditions that are tested by the branch-on-condition (BC) and jump-on-condition (JC) instructions. The contents of the program status register can be changed by:

- A system reset
- A load register (L) instruction or an add to register (A) instruction
- An instruction that changes bits

Program status register bits 0 and 1 are not assigned and are always 0. Only one of bits 5, 6, and 7 (high, low, equal) can be set by the load register instruction. If bit 7 is set to 1, hardware forces bits 5 and 6 to 0. If bit 5 is set to 1, bits 6 and 7 are forced to 0. If bit 6 is set to 1, bits 5 and 7 are forced to 0. Program status register bits are assigned as follows:

Bit	Contents
0	Not used
1	Not used
2	Binary overflow
3	Test false
4	Decimal overflow
5	High
6	Low
7	Equal

Program Status Register Setting

		Binary Overflow	Test False	Decimal Overflow	High	Low	Equal
Zero and Add Zoned Decimal	Set				Result positive	Result negative	Result zero
	Reset				Operand not positive	Operand not negative	Operand not equal zero
Add and Subtract Zoned Decimal	Set			Result overflows	Result positive	Result negative	Result zero
	Reset				Result negative or result zero	Result positive or result zero	Result not zero
Edit[1]	Set				Operand 2 positive	Operand 2 negative	Operand 2 zero
	Reset				Operand 2 not positive	Operand 2 not negative	Operand 2 not zero
Compare Logical Characters	Set				Operand 1 greater than operand 2	Operand 1 less than operand 2	Operand 1 equal to operand 2
	Reset				Operand 1 not greater than operand 2	Operand 1 not less than operand 2	Operand 1 and 2 not equal
Compare Logical Immediate	Set				Operand 1 greater than immediate data	Operand 1 less than immediate data	Operand 1 equal to immediate data
	Reset				Operand 1 not greater than immediate data	Operand 1 not less than immediate data	Operand 1 not equal to immediate data
Add Logical Characters	Set	Carry out			Carry out and result not zero	No carry and result not zero	Result equals zero
	Reset	Reset at end of instruction execution			No carry or result zero	Carry out or result zero	Result not zero
Subtract Logical Characters	Set				Operand 1 greater than operand 2	Operand 1 less than operand 2	Result zero
	Reset				Operand 1 not greater than operand 2	Operand 1 not less than operand 2	Result not zero
Add to Register	Set	Carry out			Carry out and result not zero	No carry and result not zero	Result equals zero
	Reset	Reset at end of instruction			No carry or result zero	Carry out or result zero	Result not zero
Test Bits On	Set		Test bits are not all ones				
	Reset						
Test Bits Off	Set		Test bits are not all zeros				
	Reset						
Branch or Jump On Condition	Set						
	Reset		Reset if tested	Reset if tested			
(PSR) Load Register	Set	Set if loaded; bit 10 is on.	Set if loaded; bit 11 is on.	Set if loaded; bit 12 is on.	Set if loaded; bit 13 is on.	Set if loaded; bit 14 is on.	Set if loaded; bit 15 is on.
	Reset	Reset if loaded; bit 10 is off.	Reset if loaded; bit 11 is off.	Reset if loaded; bit 12 is off.	Reset if loaded; bit 14/15 is on.	Reset if loaded; bit 14 is off or 15 is on.	Reset if loaded; bit 15 is off.
System Reset	Set						
	Reset	Binary overflow reset	Test false reset	Decimal overflow reset	High reset	Low reset	Set

[1] The program status byte settings will be as shown only if one of the following conditions exists:
1. The program status byte bit 7 was set before edit is executed.
2. The rightmost byte of operand 1 was hex 20.
3. Operand 2 is not zero.

Status Byte Registers

Status Byte 0 (Sense Only) (P)

Status byte 0 is sensed by the control processor to determine the main storage processor major and minor clock cycle times. This status byte is encoded to show these clock times:

Bits								Encoded Next Major Clock Time
0	1	2	3	4	5	6	7	
0	0							Not used
0	1							Complement latch set
		0	0	0	0			Op time
		0	0	0	1			Q time
		0	0	1	0			IH1/IX1 time
		0	0	1	1			IL1 time
		0	1	0	0			IH2/IX2 time
		0	1	0	1			IL2 time
		1	0	0	0			EA time
		1	0	1	0			EB time
		1	1	0	0			EC time
						1	1	Minor A time
						1	0	Minor B time
						0	0	Minor C time
						0	1	Minor D time

Status Byte 1 (Load Only)

Status byte 1 is loaded by the control processor. Status byte 1 is not gated out and cannot be displayed. Status byte 1 controls setting and resetting the following in the main storage processor:

Bit	Function
0	Set or reset the 'step mode' latch
1	Set or reset the 'clock run' latch
2	Issue a check reset to main storage processor checks
3	Not used
4	Not used
5	Not used
6	Not used
7	Set or reset the 'carry' trigger

Status Byte 2 (Q)

Status byte 2 is sensed by the control processor for main storage processor check conditions and control information. Bits are assigned as follows:

Bit	Function
0	An instruction that cannot be executed (nonexecutable instruction)
1	Control gate check
2	Local storage register gate check
3	Main storage gate check
4	First cycle
5	Recomplement cycle
6	Main storage processor address check
7	'Carry' trigger set

Status Byte 3 (R)

Status byte 3 is sensed and reset by the control processor. Six of the 8 bits are not used (bits 0-5). Bit 6 indicates a main storage not valid address check. Bit 7 is set by hardware to indicate that a main storage exception check occurred during a main storage processor operation. Bits 6 and 7 are both set to indicate that a main storage address register parity check occurred.

Backup Mode Register (S)

The backup mode register (BMR), which is loaded and sensed by the control processor, controls hardware switching of main storage card selection. Storage is tested by diagnostic routines during the control storage initial program load (CSIPL) procedure. If a storage failure is found in the first 8K bytes of main storage, bit 6 of the BMR is activated to cause an electronic card switch to select a different card. In order to decrease the programming needs, this switch bit is set only during the control storage initial program load sequence. Therefore, if a storage failure occurs inside the nonrelocatable storage area (first 8K bytes) during normal system operation, the system operator must perform an IPL to recover. If a main storage operation occurs with the BMR bit 6 activated, and if the physical address is between 0K and 16K, the hardware selects the same relative address in the 16K-to-32K block. In reverse, if the physical address is between 16K and 32K, the hardware selects the same relative address in the 0K-to-16K block.

Configuration Control Register (T)

The configuration control register (CCR) is an 8-bit register that:

- Selects the main storage address compare function to be performed (bits 0-3)
- Stores the main storage configuration information for input to the address check logic (bits 4-7)

Address Compare Register (U)

The address compare register (ACR) is a 17-bit register that can be loaded and sensed by the control processor. This register, along with the configuration control register (CCR), results in a main storage address compare condition that is used for synching or stopping during hardware or program error analysis. These two registers are loaded by the (SSP) alter storage/display storage routine when the main storage address compare is desired. The address compare register (ACR) contents are compared with logical addresses in the main storage address register (MSAR), or real addresses supplied by the address translation register (ATR) and/or main storage address register (MSAR), under control of the configuration control register (CCR). If the selected condition compares, and the main storage position and address stop positions are selected by switches on the CE panel, an interrupt level 5 is generated and the main storage program stops at the end of the instruction.

Address Translation

Address translation permits the System Support Program Product to load a program or blocks of a program into any 2K area of main storage. The program addresses are then translated and the program is executed as if it were located in the area specified by the link-edited addresses in the program. The three types of registers used with address translation are:

- Address translation registers
- Program mode register
- Control mode register

Address Translation Registers (V)

The 64 address translation registers (ATRs) are 1-byte registers that control address translation. Thirty-two of the registers are used for user program address translation. The remaining 32 registers are used for I/O address translation.

The 6 low-order bits of an address translation register are used to address one of sixty-four 2K-byte pages inside main storage. The 2 high-order bits are used for storage protection. To protect a 2K-byte area in storage, a hexadecimal FF is loaded into the address translation register. Any attempt to address a protected storage area by use of the address translation registers is inhibited and a storage exception is generated; this causes a processor check if the operation is an I/O operation, or an interrupt level 5 request if the operation is a main storage processor operation.

The address translation registers are used with the main storage address register to convert the logical address specified in the program to the translated real main storage address. The address translation register is selected by the 5 high-order bits in the main storage address register. A translated address is made by having the selected address translation register contents link with the 11 low-order bits from the main storage address register.

Address translation must be used to get access to any real address in main storage from 64K to 128K.

Program Mode Register (W)

The program mode register (PMR) is an 8-bit register that controls main storage address translation and protection during main storage processor operations. This register is loaded or sensed by the control processor using a register control instruction: load main storage processor register (WMPR), or sense main storage processor register (RMPR). The main storage processor can change the program mode register in either of two ways: (1) a supervisor call (SVC) instruction with an R-byte equal to hex 0D can permit the control processor to run an interrupt level 5 program routine using the load or sense main storage processor register instruction (WMPR or RMPR); or (2) a load program mode register (LPMR) instruction that enables the main storage processor to load the program mode register directly if bit 7 = 0 (nonprivileged mode bit in the program mode register). If bit 7 = 1 when the instruction is executed, the program mode register does not change and a storage exception is generated.

Program mode register bits are assigned as follows:

Bit(s)	Setting	Function
0	1	Disable task dispatching switch.
	0	Enable task dispatching switch.
1-3		Not used.
4	1	Instruction address register is translated.
	0	Instruction address register is not translated.
5	1	Operand 2 addresses are translated.
	0	Operand 2 addresses are not translated.
6	1	Operand 1 addresses are translated.
	0	Operand 1 addresses are not translated.
7	1	Task not privileged. A nonprivileged mode operation that cannot change the first 8K physical addresses in the main storage processor. The program mode register cannot be changed
	0	Privileged mode. Permits address translation in the first 8K physical addresses by the main storage processor

Control Mode Register (X)

The control mode register (CMR) is an 8-bit register that is used by the control processor to control main storage addressing. Control storage processor instructions are used to load and sense the control mode register. Bits for the control mode register are assigned as follows:

Bit(s)	Setting	Function
0-5		Not used
6	0	The address register selects the task address translation register.
	1	The address register selects the I/O address translation register.
7	0	Does not translate the address in the main storage address register (address is a real address).
	1	Translates the address in the main storage address register (address is a logical address).

ERROR CONDITIONS

When the main storage processor finds an error condition, it stops processing and sets a control processor interrupt level 5. The control processor then attempts to correct the error by determining the failing instruction and error type. Error correction and error recording are done by control storage transients that are loaded into control storage and executed after the error occurs.

Processor Error Byte (Display Byte 0)

Bit	Error	Cause
0	Storage data register parity check	Parity in the storage data register is not correct.
1	Micro-operation register parity check	Parity in the micro-operation register is not correct.
2	Storage gate parity check	Parity at the output of the storage gate is not correct.
3	ALU gate parity check	The parity expected does not match the parity generated at the ALU gate.
4	Illegal control storage address/ storage address register	Control storage was addressed outside its limits. Bits 4 and 5 both on indicates that parity in the storage address register is not correct.
5	Control storage program check/storage address register	The control storage program remained in a loop for more than 7 seconds. Bits 4 and 5 both on indicates that parity in the storage address register is not correct.
6	Illegal main storage address/main storage address register	The real or translated main storage address used by the control storage program is greater than the main storage size of the system. Bits 6 and 7 both on indicates that parity in the main storage address register is not correct.
7	Storage exception/ main storage address register	The control storage program addressed a not valid address translation register; that is, an address translation register containing hexadecimal FF. Bits 6 and 7 both on indicates that parity in the main storage address register is not correct.

Decode of Bits 6 and 7

Bits 6 7	CMR Bit 7	PMR Bit 7	Cause
1 0	0	*	Invalid main storage address (real)
1 0	1	*	Invalid main storage address (translate)
0 1	1	*	Storage protect
0 1	*	1	MSP tried to alter PMR while PMR bit 7 = 1
1 1	*	*	MSAR parity check
1 1	1	*	ATR parity check

Legend: * = don't care

Main Storage Processor Checks

The following data is recorded for each error:

- The operation code
- The contents of the Q-byte register
- The contents of the instruction address register and the address translation register used by the instruction address register
- The contents of index register 1
- The contents of index register 2
- The contents of the address recall register
- The contents of operand 1 and the address translation register used by operand 1
- The contents of operand 2 and the address translation register used by operand 2
- The contents of the program status register
- The contents of the program mode register
- The main storage processor status bytes
- The physical (real) failing address (storage read errors)
- The time and date

Examples of the error history tables for the main storage processor and the control processor can be found under *Error Indications* in the *Control Processor* section of this manual. Error correction procedures are determined by the type of error. The main storage processor error stops the main storage processor program (except when in check run mode) and sets the control processor interrupt level 5.

The control storage program analyzes the cause of the error. If the error is caused by a storage exception (not a valid main storage address or the address translation register contained hexadecimal FF), the control storage program terminates the main storage task that caused the check condition (user error). This type of error is not recorded. If the error is a hardware error, the control storage program analyzes the error condition.

If the error can be corrected, the transients, which are put into control storage, perform error correction, record the error information in the main storage processor error recording area, and restart the main storage processor at the point the program error was found. When correction is possible, the error is transparent to the program except for the time needed to execute the error correction function. If error correction is not possible, the error information is recorded in the main storage recording area and on disk. The task that caused the main storage processor error is terminated. If the error occurs when the control processor is in control, a software processor check halt is issued.