IBM Field Engineering

Theory of Operation

5410 Processing Unit

SY31-0207-1

IBM Field Engineering

Theory of Operation

5410 Processing Unit

Preface

This manual, SY31-0207, describes the operation of the IBM 5410 Processing Unit. The manual gives an explanation of the logical functions of a circuit and the major circuit objectives. With this information, the CE can interpret the operation of circuits illustrated in the companion Field Engineering Maintenance Diagrams (FEMDs).

Other manuals necessary for the CE to understand and service the 5410 are:

1. For instructional and maintenance diagrams, flowcharts, and timing charts:
   IBM 5410 Processing Unit Field Engineering Maintenance Diagrams.

2. For maintenance procedures:
   IBM 5410 Processing Unit Field Engineering Maintenance Manual.

Second Edition

This manual is a complete revision of, and obsoletes, SY31-0207-0.

Some illustrations in this manual have a code number in the lower corner. This is a publishing control number and is not related to the subject matter.

Changes are continually made to the specifications herein; any such change will be reported in subsequent revisions or FE Supplements.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Product Publications, Department 245, Rochester, Minnesota 55901.

List of Abbreviations

AAR	A Address Register
ALD	Automated Logic Diagram
ALU	Arithmetic and Logic Unit
ARR	Address Recall Register
ASCII	American National Standard Code for Information Interchange
BAR	B Address Register
BSM	Basic Storage Module
BSCA	Binary Synchronous Communications Adapter
CPU	Central Processing Unit
CR	Condition Register
CRR	Condition Recall Register
DA	Device Address
DBI	Data Bus In
DBO	Data Bus Out
DFDR	Disk File Data Address Register
DFCR	Disk File Control Address Register
DPF	Dual Program Feature
DRR	Data Recall Register
EBCDIC	Extended Binary Coded Decimal Interchange Code
Hz	Hertz
IAR	Instruction Address Register
I/O	Input/Output
IPL	Initial Program Load
K	Thousand
LCR	Length Count Register
LCRR	Length Count Recall Register
LPDAR	Line Printer Data Address Register
LPIAR	Line Printer Image Address Register
LSR	Local Storage Register
MAP	Maintenance Analysis Procedure
MFCU	Multi-Function Card Unit
MPCAR	MFCU Punch Data Address Register
MPTAR	MFCU Print Data Address Register
MPDAR	MFCU Read Data Address Register
MST	Monolithic System Technology
OV/OC	Overvoltage/Overcurrent
PC	Parity Check
PG	Parity Generate
POR	Power On Reset
PSR	Program Status Register
SAR	Storage Address Register
SCR	Silicon Controlled Rectifier
SDR	Storage Data Register
SIOC	Serial Input/Output Channel
SMS	Standard Modular System
S/Z	Sense/Inhibit
UC	Undercurrent
XR	Index Register

The IBM System/3 is a compact, high-speed data processing system designed to use the 96 column card. The IBM 5410 Processing Unit (CPU), the control unit for the System/3, is supplemented with table-top I/O units arranged for convenient operation by a single operator.

This manual describes the central processing unit for the IBM System/3. Figure 1-1 shows the system configuration.

Compact packaging of the IBM System/3 is made possible through the use of miniaturized monolithic systems technology (MST) and the small, increased capacity card. The minimum system configuration (CPU, MFCU, printer) provides for complete unit record type functions including card reading, punching, interpreting, collating, reproducing, summary punching, computing, and printed reports all under program control. The addition of disk storage drives, with removable disk cartridges, offers practically unlimited data storage growth.

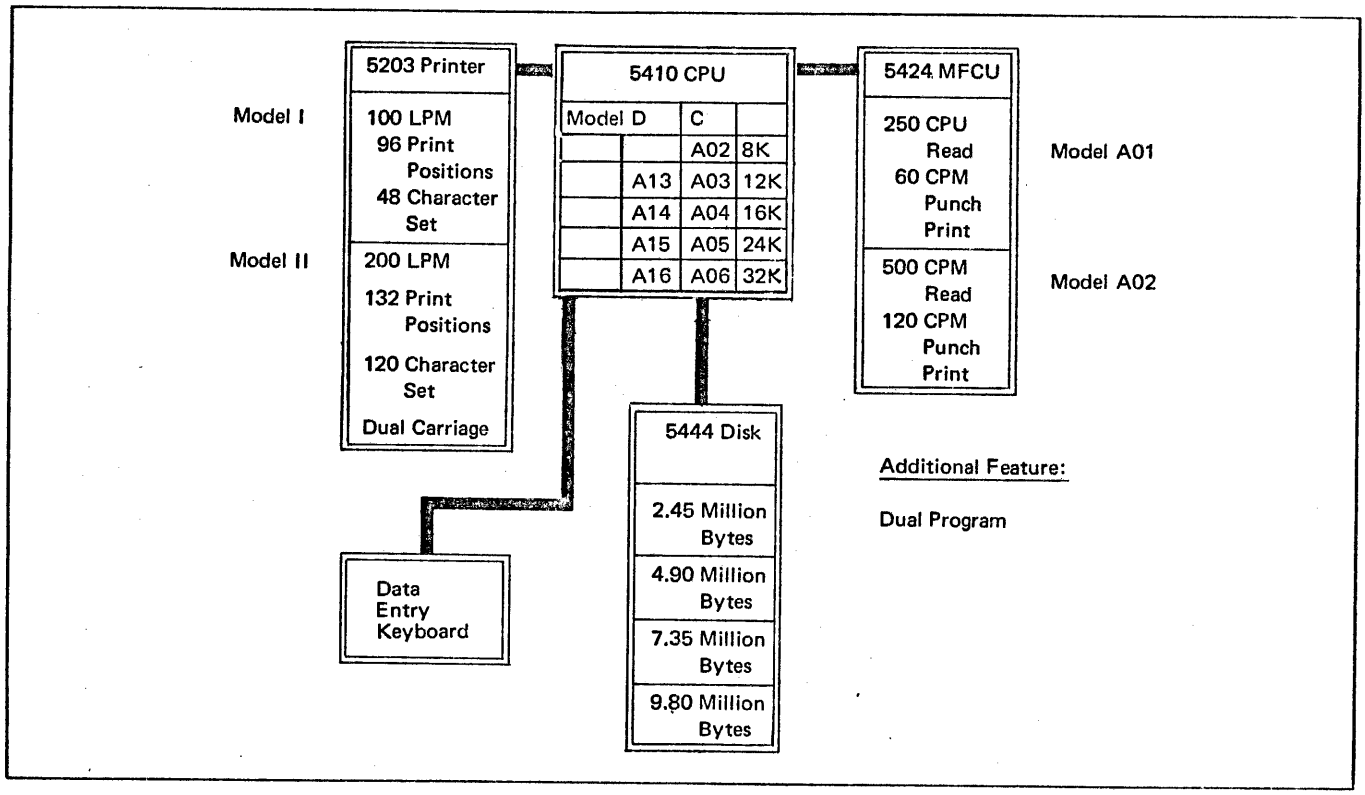


Figure 1-1. System/3 Configuration

SYSTEM COMPONENTS

5410 Central Processing Unit (CPU)

The IBM 5410 CPU (Figure 1-2) contains the facilities for addressing storage, arithmetic and logical processing of data, sequencing instructions, and controlling the transfer of data between core storage and attached input/output devices. The basic unit of information is the byte which represents one alphabetic, numeric, or special character. In arithmetic operations, a byte contains one numeric character and a zone. The low order byte contains the sign in the zone portion. Bytes may be handled separately or grouped together to form fields.

The CPU contains 8192 (8K) positions of core storage which may be increased to 32,768 (32K) positions. The storage locations are numbered consecutively (0, 1, 2, . .) Each number corresponds to the address of an individual byte.

The 5410 Model C is the base system CPU. A02 through A06 correspond to core size (Figure 1-1). Model D (disk system) starts with A13 (12K storage) and continues to A16.

The CPU core storage unit has a read/write cycle time of 1.2 us., with a data access time of 465 ns from the start of read. A calculation time is inserted between the read/write time to provide a basic machine cycle (read/compute/write) time of 1.52 us.

Addressing for CPU functions is maintained in local storage registers (LSRs). These registers contain the core addresses necessary for instruction sequencing as well as data manipulation: both internally, and to and from I/O devices. In addition, the LSRs are used as temporary storage for data while the CPU is performing instructions.

Step-by-step data processing is controlled by registers (op register, Q register, and condition register) which contain the operation code for the instruction being performed and the additional information required to execute the instruction.

Calculations within the CPU are performed in the arithmetic and logical unit (ALU). All data to be processed within the CPU is routed through the ALU which is capable of performing the action required to arrive at the desired result.

The CPU has direct control over all the I/O devices attached to it. I/O operations are initiated and tested by program instructions which determine what operation is performed (read, write, etc.,) and which unit is to be used.

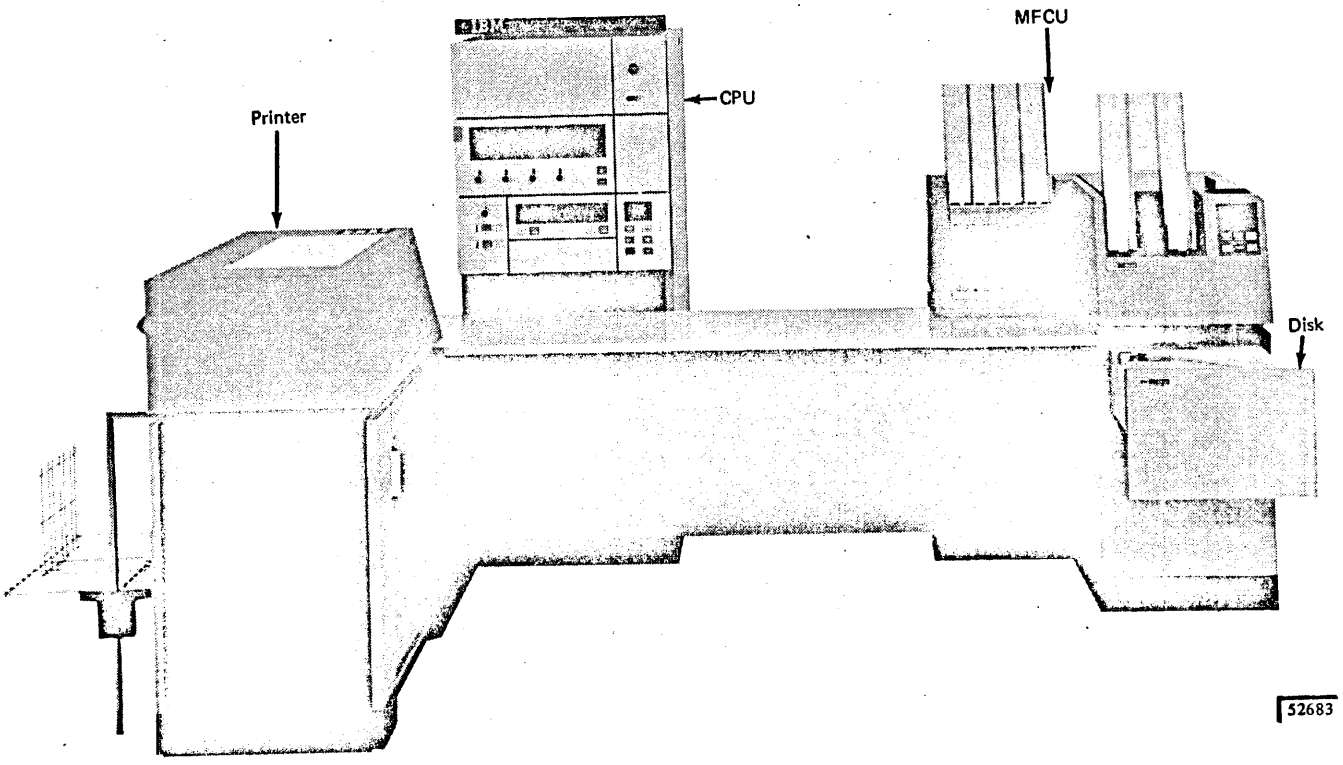


Figure 1-2. IBM System/3

An automatic interrupt is provided to allow the system to make optimum use of the I/O devices. An interrupt originates at an I/O device which requests special attention from the CPU. Generally, an interrupt means that the CPU must interrupt a current instruction sequence, perform an intervening instruction sequence, and return to the interrupted program.

In addition, the CPU is available for processing during most of each I/O operation even though many devices may be functioning simultaneously. This overlap of I/O operations and CPU processing is made possible by a cycle steal capability by which an I/O device, while performing an I/O operation, breaks into the main program and uses enough cycles to transmit the bytes which are immediately available. For instance, when reading a row of data from a card, that row of data is placed in the proper core location with cycle steal cycles and the main program then continues until the next row of data is available. Thus, the cycle steal capability provides the benefit of a buffer without sacrificing storage capacity or requiring a separate buffer.

In the CPU, odd parity is provided for all bytes of data to provide a means of checking for the validity of data. As the data is transferred throughout the CPU, each register is parity checked to determine if the data transferred correctly. As data is changed within the ALU, the parity of the resultant byte is determined by a second ALU (check ALU) and the generated parity is then checked at the ALU latched output.

In addition to parity checking, as the program is executed each operation code is checked to ensure that it contains a valid instruction.

5424 MFCU

The IBM 5424 Multi-Function Card Unit (MFCU) provides the IBM System/3 with the capabilities of many single unit accounting machines. With two hoppers, a phototransistor read station, a common punching station, a printing station, and four selective stackers, the MFCU offers full card file maintenance abilities plus three or four lines of card document printing.

Cards from both the primary and secondary hopper can be read, punched, printed, and selected into any one of the four stackers, regardless of the hopper origin. The traditional unit record functions of reproducing, gang-punching, summary punching, interpreting, collating, and sorting can be performed on the MFCU under complete control of the stored program.

5203 Printer

The IBM 5203 Printer provides printed report output for the IBM System/3. The alphabetic, numeric, and special characters are assembled on a chain and the printing format is controlled by the system's stored program. As the chain travels in a horizontal plane, each character is printed as it is positioned opposite a magnet driven hammer that drives the form against the chain.

An interchangeable chain cartridge permits the operator to change type fonts and character sets. Spacing and skipping is performed by a tapeless carriage under control of the CPU stored program.

5444 Disk Drive

The IBM 5444 Disk Drive provides System/3 Model D with dual disk capabilities on a single disk spindle. One disk is mounted permanently in a container at the base of the spindle; the other is mounted at the top of the spindle and is removable.

Each disk contains 2.45 million bytes of storage. Therefore, the addition of a second disk drive provides online disk capacity of 9.80 million bytes.

MACHINE LANGUAGE

Number Systems

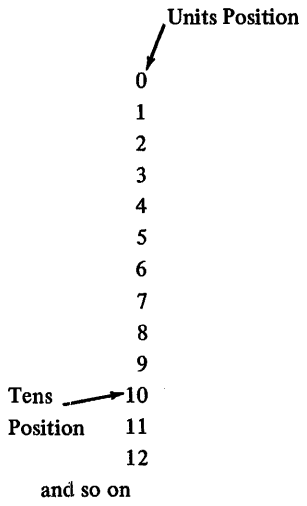
To understand the operation of the CPU, it is necessary to become familiar with the number systems and character codes used. Accordingly, the following topics discuss the decimal, binary, and hexadecimal number systems.

Decimal Number System

- System has ten symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.
- Base of system is 10.

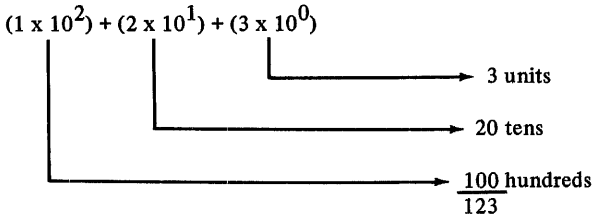
The decimal number system has ten symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. Counting starts in the units position with 0 and proceeds through the next nine symbols. When 9 is reached, there are no more symbols; therefore, a 1 is

placed in the position to the left (tens position) and the count resumes with a 0 in the original position:



Continuing the count, it takes 101 numbers (count began with zero) before a third position (hundreds position) is required to express a 3-digit number. Similarly, it takes 1001 numbers before a fourth position (thousands position) is required to express a 4-digit number. Because of the role that the powers of 10 play in the representation of a number, (10 unique symbols), 10 is said to be the base of the decimal system.

A number is made up of terms corresponding to the number of positions required to express the number. Each term consists of a product of a power of 10 and some number from 0 to 9. For example, the number 123 breaks down as follows:



Binary Number System

- System has two symbols: 0 and 1.
- Base of system is 2.

Current digital computers use binary circuits and, therefore, binary mathematics. The binary, or base 2, system uses two symbols, 0 and 1, to represent all quantities. Counting is started in the same manner as in the decimal system, with 0 for zero and 1 for one. At this point, there are no more symbols to be used. It is therefore necessary to express a 2 by placing a 1 in the next position to the left and starting again with 0 in the original position. Thus binary 10 is equivalent to 2 in the decimal system. Counting continues with a carry to the next higher order every time a 2 is reached instead of every time a 10 is reached. Counting in the binary system is as follows:

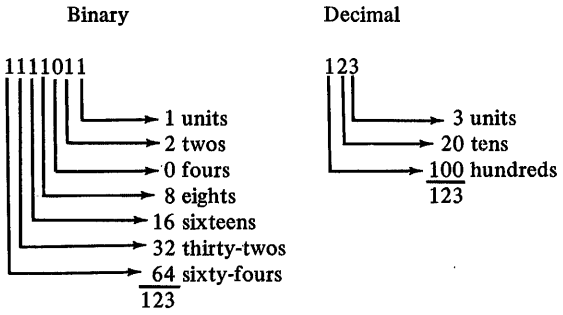
Binary	Decimal	Binary	Decimal
0	0	110	6
1	1	111	7
10	2	1000	8
11	3	1001	9
100	4	1010	10
101	5	1011	11

and so on

The 1s and 0s of a binary number represent the coefficients of the ascending powers of 2. To illustrate, assume the binary number 1111011; the number is expressed as:

$$(1 \times 2^6) + (1 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$

The various terms do not have the meanings of units, tens, hundreds, thousands, etc., as in the decimal system, but signify units, twos, fours, eights, sixteens, etc. Thus the binary number breaks down as follows (compared with decimal equivalent):



Hexadecimal Number System

- System has 16 symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.
- Base of system is 16.
- System is shorthand notation for binary numbers.
- Four binary bits are represented by one hexadecimal symbol.
- Byte is represented by two hexadecimal symbols.

Binary numbers have approximately 3.3 times as many terms as their decimal counterparts. This increased length presents a problem when talking or writing about binary numbers. A long string of 1's and 0's cannot be effectively spoken or read. A shorthand system is necessary, one that has a simple relationship to the binary system and that is compatible with the basic 8-bit byte used in the CPU. The hexadecimal number system meets these requirements.

The hexadecimal system has sixteen symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. Counting is performed as in the decimal and binary systems. When the last symbol (F) is reached, a 1 is placed in the next position to the left and counting resumes with a 0 in the original position, as follows:

0	10	20	A0
1	11	21	A1
2	12	22	A2
3	13	23	↓
4	14	↓	
5	15		
6	16		and so on
7	17		
8	18		
9	19		
A	1A	9A	
B	1B	9B	
C	1C	9C	
D	1D	9D	
E	1E	9E	
F	1F	9F	

One hexadecimal symbol can represent four binary bits. Thus the 8-bit binary byte, in turn, can be represented by two hexadecimal symbols. The relationship between the hexadecimal, binary, and decimal systems is as follows:

Hexadecimal	Binary	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

The important relationship to remember is that four binary positions are equivalent to one hexadecimal position.

Hexadecimal numbers are represented in the same manner as decimal and binary numbers, except that the base is 16. The terms of the number represent the coefficients of the ascending powers of 16. For example, consider the hexadecimal number 257 (decimal equivalent equals 599):

$$257 = (2 \times 16^2) + (5 \times 16^1) + (7 \times 16^0)$$
$$= (2 \times 256) + (5 \times 16) + (7 \times 1)$$
$$= 512 + 80 + 7$$
$$= 599$$

Number Conversions

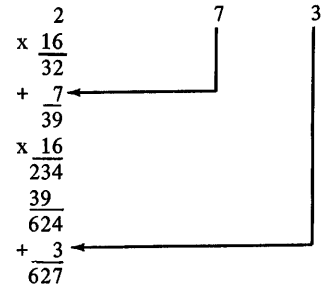
The preceding topics derived decimal equivalents by multiplying the terms by the coefficients of the ascending powers of the base. Large numbers are difficult to convert with this method. The following topics outline simpler methods for converting hexadecimal to decimal and back, and hexadecimal to binary and back.

Hexadecimal to Decimal

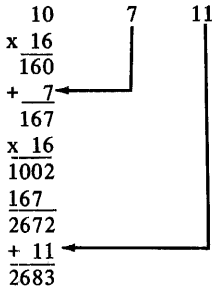
To convert a hexadecimal number to its decimal equivalent:

1. Convert any term represented by a letter symbol to its decimal equivalent.
2. Multiply the high-order term by 16.
3. Add the next lower order term to the product obtained in step 2.
4. Multiply the result obtained in step 3 by 16.
5. Add the next lower order term to the product obtained in step 4.
6. Continue multiplying and adding until the low order term has been added to the answer, at which time the conversion is complete.

As an example, convert the hexadecimal number 273 to its decimal equivalent.



As a second example, convert the hexadecimal number A7B. Converting letter symbols to decimal equivalents yields 10 7 11.



Decimal to Hexadecimal

To convert a decimal number to its hexadecimal equivalent:

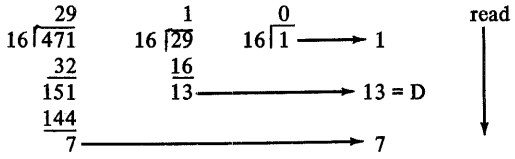
1. Divide the decimal number by 16; the remainder of this first division becomes the low order term of the final answer.

2. Divide the quotient (received from the first division) by 16; again the remainder becomes a part of the final answer (next higher order term).

3. Repeat steps 1 and 2 until the quotient is less than 16. This final quotient is the high order term of the final answer.

4. Convert any term between 10 and 15 to its hexadecimal letter-symbol equivalent.

For example, convert the decimal number, 471 to its hexadecimal equivalent:



answer = 1D7

Hexadecimal to Binary and Binary to Hexadecimal

Hexadecimal terms 0 through F, which have the values of 0 through 15, respectively, are represented in the binary system by four binary bits. To convert a hexadecimal number to its binary equivalent, express each term in its equivalent 4-bit binary group. To convert binary numbers to hexadecimal numbers, reverse the process.

For example:	Hexadecimal:	3	7	B	1
	Binary:	0011	0111	1011	0001
Another example:	Hexadecimal:	A	6	5	F
	Binary:	1010	0110	0101	1111

Data Formats

The basic unit of information in the CPU is the byte. Each byte is 8 bits or two hexadecimal characters in length. An additional bit (P bit) is added to each 8-bit byte to maintain odd parity. Figure 1-3 shows the bit structure of a byte.

Each main storage address location contains one byte of information. Therefore, each time main storage is addressed a full byte is read from storage to be processed.

To represent data, each byte is divided into two parts. Bits 4 to 7 represent the numeric portion of a character and bits 0 to 3 represent the zone portion. Therefore, a byte can represent numeric, alphabetic, or special characters. These characters are expressed in EBCDIC (Extended Binary Coded Decimal Interchange Code).

When used as a numeric quantity, each byte contains one numeric digit in bits 4 to 7 with the sign of the entire field contained in the zone portion of the low order byte. The zone portion of the rest of the bytes in the field contain the EBCDIC for a numeric digit (F in hexadecimal). The EBCDIC for plus is the hexadecimal F and for minus hexadecimal D. Internally, the CPU also recognizes the ASCII-8 (American National Standard Code for Information Interchange) for minus (B in hexadecimal) but changes it to EBCDIC in the result field of a decimal operation. Any other zone combination is considered to be plus.

A minus field is entered into the CPU from a card by punching a B zone over the units position of the field. A plus field contains no zone punch. Figure 1-4 contains a conversion chart for EBCDIC and card code.

The maximum length of a source field is 16 digits and the maximum length of a result field is 31 digits.

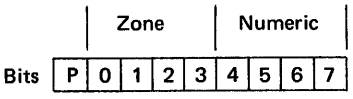


Figure 1-3. Byte Structure

EBCDIC Bits 0123 / 4567 (Zone, Digit Pch, Punch)	1100	1101	1110	1111
	BA	B	A	
0000				*ϕ
0001	1	A	J	1
0010	2	B	K	S	2
0011	21	C	L	T	3
0100	4	D	M	U	4
0101	4 1	E	N	V	5
0110	42	F	O	W	6
0111	421	G	P	X	7
1000	8	H	Q	Y	8
1001	8 1	I	R	Z	9

*Card Code for Numeric Zero is A Only

Figure 1-4. Card Code and EBCDIC Conversion Chart

CPU OPERATION

Bridge Basic Storage Module

The bridge basic storage module (BSM) is a ferrite core storage unit, available in three basic sizes:

1. 8,192 byte (9 bit) readout.
2. 16,384 byte (9 bit) readout.
3. 32,768 byte (9 bit) readout.

These capacities are commonly called 8K 9 bit, 16K 9 bit, and 32K 9 bit. The 24K 9 bit or 32K 9 bit capacities can be obtained by chaining two BSMs together or by using one 32K 9 bit BSM. Check circuitry causes an invalid address for addresses above the rated capacity.

Physically, the BSM is a separately packaged unit that mounts in the space provided for an MST-1 board. It contains a core array, timing and control circuits, a drive system, and a sense/inhibit system. The BSM also includes the storage data register (SDR), but does *not* contain a storage address register (SAR).

Communication between the system and storage is via interface lines which transfer address information, input data, output data, and control signals. Interface circuits are compatible with MST-1 circuit technology. Within the BSM some SLT circuits are used.

Each storage cycle (read cycle or write cycle) takes approximately 1.2 us. Once it has been started by a read call/write call signal, the BSM executes one cycle, either a read or a write cycle, depending upon which was last completed. Since the core storage has a destructive readout, a write cycle must always follow a read cycle in order to allow the data to be regenerated. Interlocks are provided to ensure that read and write cycles alternate properly. However, a system reset resets these interlocks so that the first storage cycle is a read cycle.

Access time, from read call until data is available at the interface, is approximately 465 ns.

Basic Data Flow

The system storage address register (SAR) addresses the storage unit (see 'Addressing System' for details). When the system signals the storage unit with read call/write call, a read cycle is started. The data located at the specified address is read out to the storage data register (SDR) and placed on data lines to the system. The read cycle is complete and the storage clock stops after approximately 600 ns. Data flow is shown in Figure 1-5.

The data read out may be placed back into the addressed location, or new data may be placed into that storage position. When storing new information, 'store new' causes the data latches to be cleared prior to their setting with new data. Read call/write call starts a write cycle and the information in the data latches is transferred to core storage. The write cycle is complete and the storage clock stops after approximately 600 ns.

Instruction and Execute Cycle

- I-cycles read out instructions from storage.
- A cycles and B cycles execute the instruction.

There are two types of machine cycles used in the internal operation of the CPU. These are instruction cycles (I-cycles) and execute cycles.

I-cycles are used to move the instructions from storage to the various registers required to execute the instruction. If the instruction does not require additional use of main storage after the completion of I-cycles, such as a branch instruction, the operation is completed without execute cycles. However, most operations require the use of data from one or two main storage fields. Execute cycles are used to manipulate this data to perform the operation.

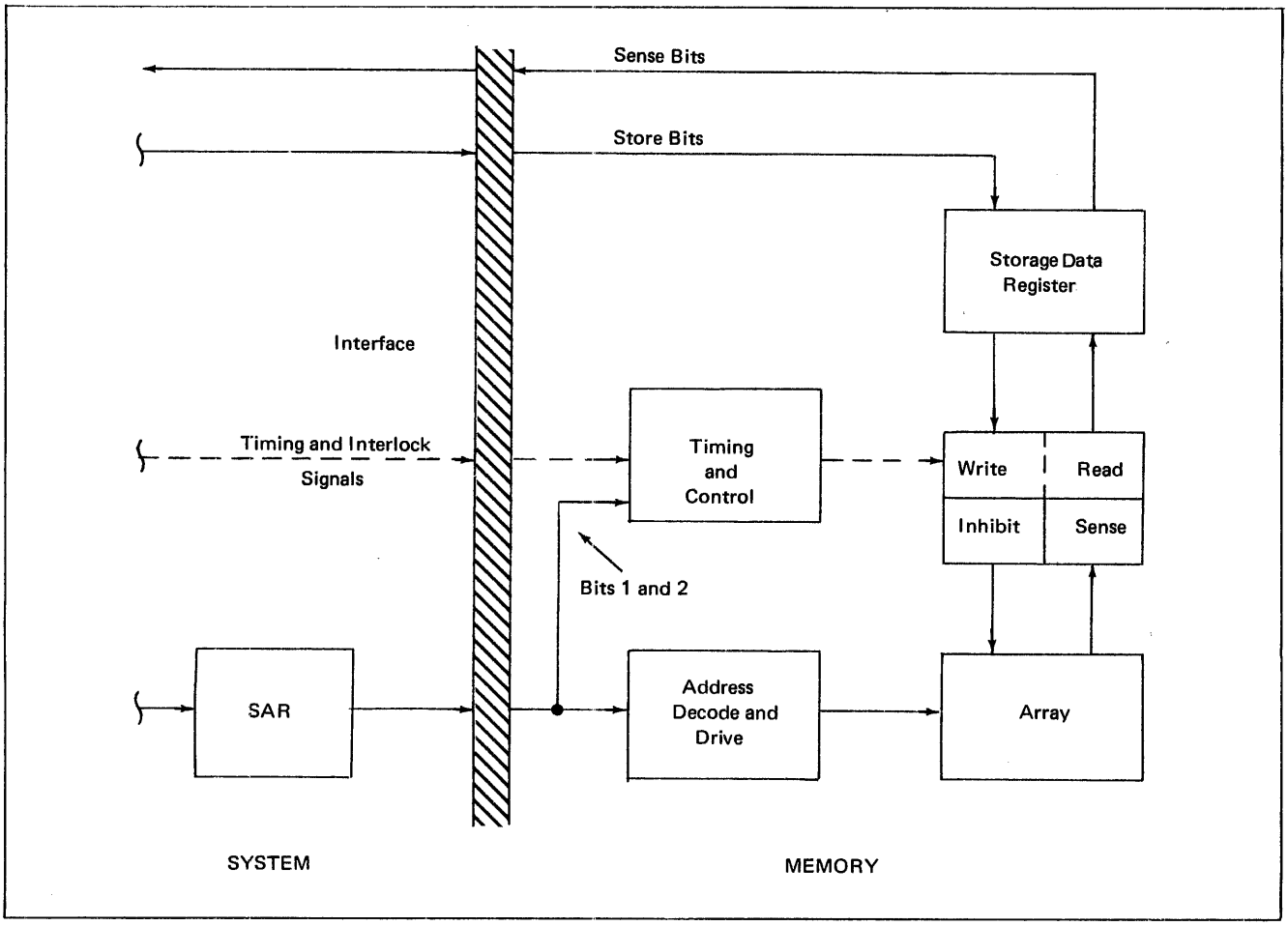


Figure 1-5. System-Memory Data Flow

Two types of execute cycles are used by the CPU. A-cycles are used to address main storage for the source fields and B-cycles are used to address main storage for the result field. If only one field is involved with the instruction, B-cycles are used to address main storage.

Sequential Instruction Execution

- Instructions are located in consecutive, ascending main storage locations.
- Instruction Address Register (IAR) is incremented by 1 each instruction cycle.

The CPU performs computations in a step-by-step procedure similar to manual accounting. However, while many steps may be combined in a manual operation, the CPU requires a separate operation for each step. For instance, when figuring net pay in a manual payroll operation, all the deductions could be added together in one step. The CPU must add each deduction into the deduction total in a separate operation (Figure 1-6).

The example shown would probably require three steps in the CPU. Since it would probably be desirable to retain the federal tax figure, the total deductions would be calculated in a separate location. Therefore, the first step would be to reset the total deduction field to zeros and add the federal tax into the total deduction field (zero and add zoned). Then in separate steps the state tax and United Fund fields would be added to the total (add zoned decimal).

Manual		CPU	
Federal Tax	17.50	Federal Tax	17.50
State Tax	6.30	State Tax	6.30
United Fund	1.50	Total	23.80
Total Deductions	25.30	United Fund	1.50
		Total Deductions	25.30

Figure 1-6. Step-By-Step Processing

Because of this step-by-step method of processing, the instructions are arranged in ascending storage locations in the CPU (Figure 1-7). Instruction sequence is maintained by retaining the address of the storage location in the instruction address register (IAR). The IAR is a two byte LSR and is incremented each cycle so the next ascending storage location can be addressed.

In the sample program shown in Figure 1-7 the IAR would contain the storage location 1000. Storage location 1000 is then addressed during an I-cycle and the operation code is read from storage and loaded into the operation register (op register). The quantity 1 is then added to the IAR so storage location 1001 can be addressed during the next cycle. This process continues until the storage location 1005 has been addressed. The instruction is then executed. After the instruction has been completed, the IAR is again used to address storage location 1006 (op code of next instruction).

Branching

- If branch condition is met, CPU branches to address given in branch instruction for location of next instruction to be performed.

Branching is used by the CPU to alter the instruction sequence under certain conditions and thus provides flexibility within a given CPU program. By branching to a different storage location and skipping certain steps the results of the stored program are altered. Figure 1-8 shows a sample program that contains a branching operation. In this example, the company has a stock option plan which permits employees who earn $80.00 or more to purchase stock. If they do not earn $80.00 or more, the stock deduction is bypassed.

During the subtract zoned decimal operation the condition register (CR) is set to high, low, or equal depending upon the result. This is a function of all arithmetic and compare operations, with the CR setting varying with the operation performed. For the subtract zoned decimal operation, the CR is set to low if the result is negative. When subtracting $80.00 from the salary field, any salary of less than $80.00 will result in a minus total.

The Q code bit structure determines the branch condition for a branch on condition instruction. In the example shown, the branching condition is a CR low setting (Figure 1-8). If the CR is set at high or equal, the next sequential instruction (storage location 1028) is performed. However, if the CR is set at low, the branch-to address (storage location 1034) replaced the next sequential address and the next instruction is bypassed.

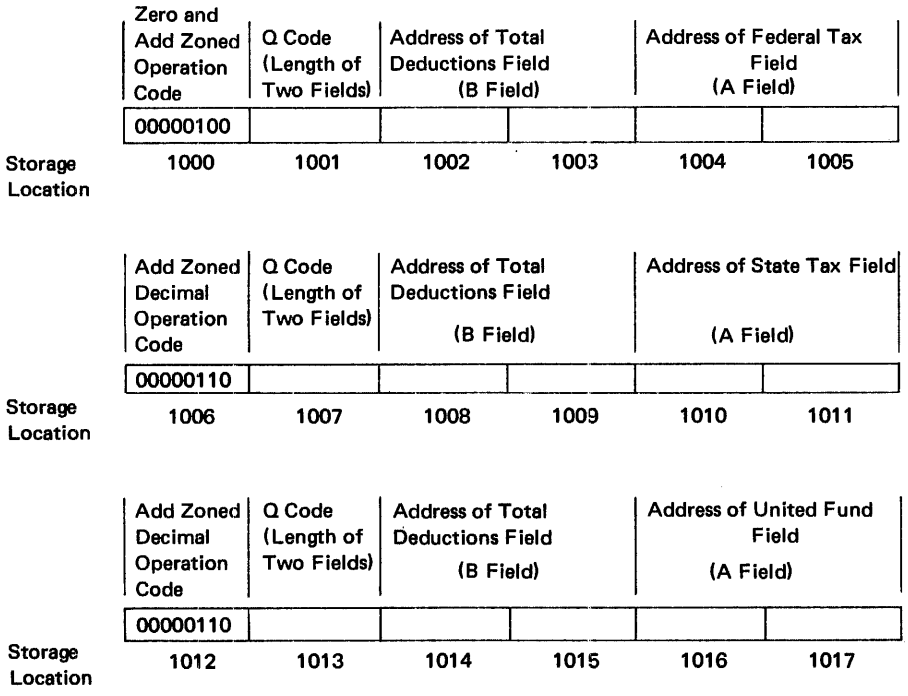


Figure 1-7. Sequential Instruction Execution

I/O Data Transfer

The CPU issues a start I/O (SIO) instruction which starts the needed mechanical motion. Whenever the I/O device reaches a point in its mechanical operation where it needs data from storage (write, print, punch) or has data to send to storage (read) the device requests an I/O cycle. The CPU uses two methods for transferring data to and from the various I/O devices: cycle steal and interrupt.

Cycle Steal

The 5424 is one I/O device which uses the cycle steal method of transferring its data. An I/O cycle request can occur during any cycle and is granted before the next CPU cycle. The attachment then has complete control of data flow, LSR selection, and storage during that cycle.

More than one device attachment may request a cycle at a time so each device is assigned a cycle steal priority.

Interrupt

Some I/O attachments operate by means of an interrupt routine. An interrupt differs from a cycle steal by interrupting the main program with a separate program routine. For this reason, an interrupt can occur only at the completion of an instruction.

I/O attachments transferring data during interrupt routines are assigned priorities, as in cycle steal. The highest interrupt level device attachment takes precedence over lower level devices. It is possible for one interrupt routine to interrupt another interrupt routine of lower priority.

Addressing

The CPU uses two types of addressing for the various fields when executing instructions: direct addressing and indexed addressing.

Direct Addressing

- Two byte address is contained in instruction.

The sample programs used in the two previous topics are examples of direct addressing. Direct addressing requires two bytes for each field or location used by the instruction. The first address which follows the Q code in the instruction, is the address of the result or destination field (B field). For two address instructions, the second address is the source field (A field). The B field address is maintained in the B address register (BAR) and the A field address is maintained in the A address register (AAR).

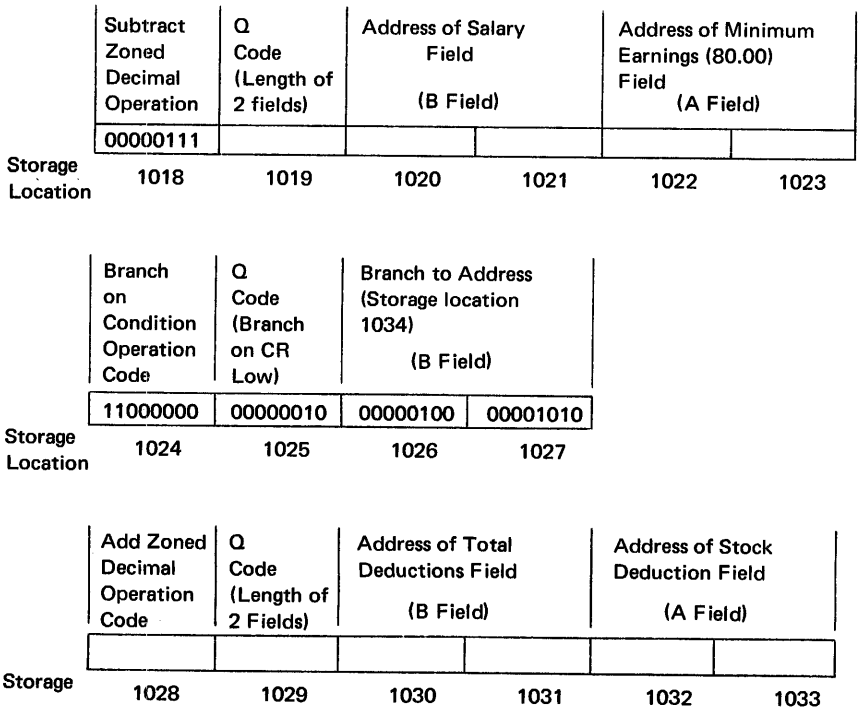


Figure 1-8. Branch On Condition

Most addresses given in the instructions are for the location of the low order or right hand digit of the field. Therefore, as the instruction is executed and as each digit position is processed, the BAR and AAR are decremented to address core in descending order. An exception is the insert and test character operation which is executed from high order to low order digits. In this case the BAR is incremented in the same manner the IAR is incremented during instruction cycles.

Indexing

- Two byte index register is added to one byte from instruction to create new address.

Indexing provides the programmer with a means of changing addresses within a program without changing the instruction. An indexed address consists of a single byte within the instruction. This single byte is added to the contents of a two byte index register to form a new address. The indexed address is then loaded into the BAR or AAR depending upon the address being indexed.

Indexing is used in; (1) performing an instruction with an indexed address, (2) adding a constant to the index register, and (3) branching to an address to perform the instruction at a different core storage location. Thus it is possible to perform an instruction or series of instructions many times without wasting main storage by repeating the instruction.

Either of two index registers (XR-1 or XR-2) can be selected for indexing. The recognition of indexed addresses and the selection of each index register is covered under 'Instruction Formats.'

Instruction Formats

- Instruction length is three to six bytes.
- Bits 0-3 of op code determine type of instruction and addressing.

The CPU performs three types of instructions. They are:

- two address instructions
- one address instructions
- command instructions

Two address instructions are those instructions which involve two separate fields within main storage and therefore contain two addresses. Most one address instructions involve only one field within main storage and therefore contain one address (the load address instruction contains the needed data rather than an address). Command instructions are those instructions which do not involve main storage at all and therefore contain no addresses.

Each instruction consists of an operation code and a Q code (Figure 1-9). These are followed by either a control code, or one or two addresses. Thus, the length of the instruction varies from three to six bytes depending upon the type of instruction and the type of addressing.

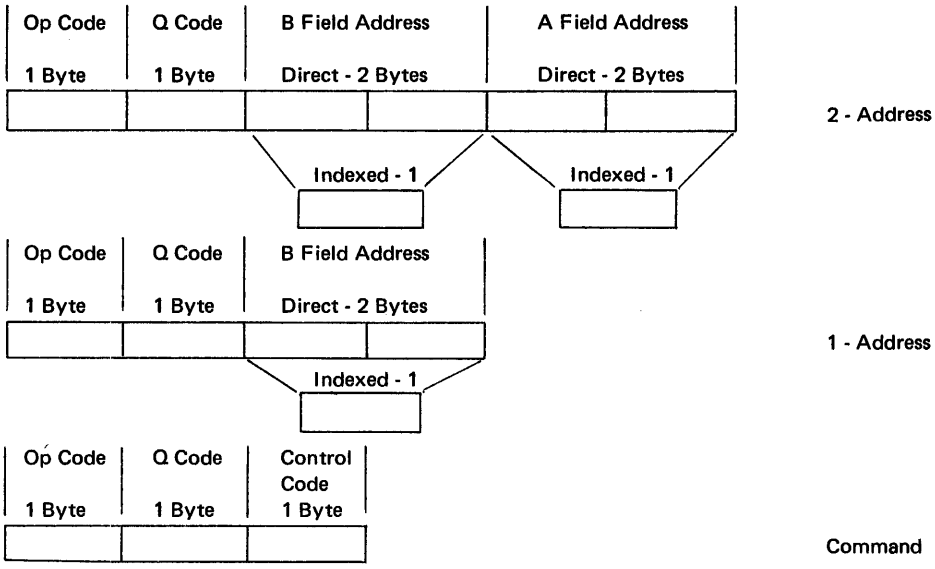


Figure 1-9. Instruction Format

The first half byte of the op code (bits 0-3) determines the format of the instruction performed (one address, two address, etc.) and the method of addressing used (Figure 1-10). If all four bits are present, the instruction is a command instruction. From there the bits are broken into two groups (bits 0-1 and bits 2-3). If both bits in either group are present, the instruction is a one address instruction; if neither group has both bits present, the instruction is a two address instruction. If a bit is present in either of the groups in a two address instruction or in the bit-absent group of a one address instruction, the address is indexed (Figure 1-10).

Number of Bytes in Address	B Field Address	Op Code Bits 01	Op Code Bits 23	A Field Address
2	Direct	00	00	Direct
1	Indexed XR1	01	01	Indexed XR1
1	Indexed XR2	10	10	Indexed XR2
	No address	11	11	No address

Figure 1-10. Determining Instruction Format

Instructions

The second half byte (bits 4-7) of the op code determines the actual operation performed. Use of the Q code and control code depend upon the operation being performed. The complete instruction set performed by the CPU is shown in Figure 1-11.

Operation Type	Mnemonic and Operation		Q Code Use	Control Code Use
2 Address	ZAZ	Zero and Add Zoned	Field Length	
	AZ	Add Zoned Decimal		
	SZ	Subtract Zoned Decimal		
	MVC	Move Characters		
	ALC	Add Logical Characters		
	SLC	Subtract Logical Characters		
	CLC	Compare Logical Characters		
	ED	Edit		
	ITC	Insert and Test Characters		
	MVC	Move Hex Character	Half-byte Selection	
1 Address	MVI	Move Logical Immediate	Immediate Data	
	CLI	Compare Logical Immediate		
	SBN	Set Bits On Masked	Bit Selection	
	SBF	Set Bits Off Masked		
	TBN	Test Bits On Masked		
	TBF	Test Bits Off Masked		
	ST	Store Register	Register Selection	
	L	Load Register		
	A	Add to Register		
	LA	Load Address		
	BC	Branch on Condition	Branch Condition	
	TIO	Test I/O and Branch		
	SNS	Sense I/O	Device Address and Data Selection	
	LIO	Load I/O		
Command	HPL	Halt Program Level	Halt Identifier (tens)	Halt Identifier (units)
	APL	Advance Program Level	Advance Condition	Not Used
	JC	Jump on Condition	Jump Condition	Address Modifier
	SIO	Start I/O	Device Address and Unit	Stacker select, spacing, etc.

Figure 1-11. Instructions

DATA FLOW

Data flow for System/3 is shown in Figure 1-12. Data flows through the machine in 8-bit bytes plus one parity bit. It flows serially by byte through the arithmetic and logic unit (ALU) and is distributed to the remaining functional units of the machine.

The ALU receives two bytes of data and combines them in parallel into one byte. It is able to perform decimal add and subtract, binary add and subtract, and logical AND and OR operations. All data to the ALU comes from the A and B registers, and the output is the contents of B modified by the contents of A.

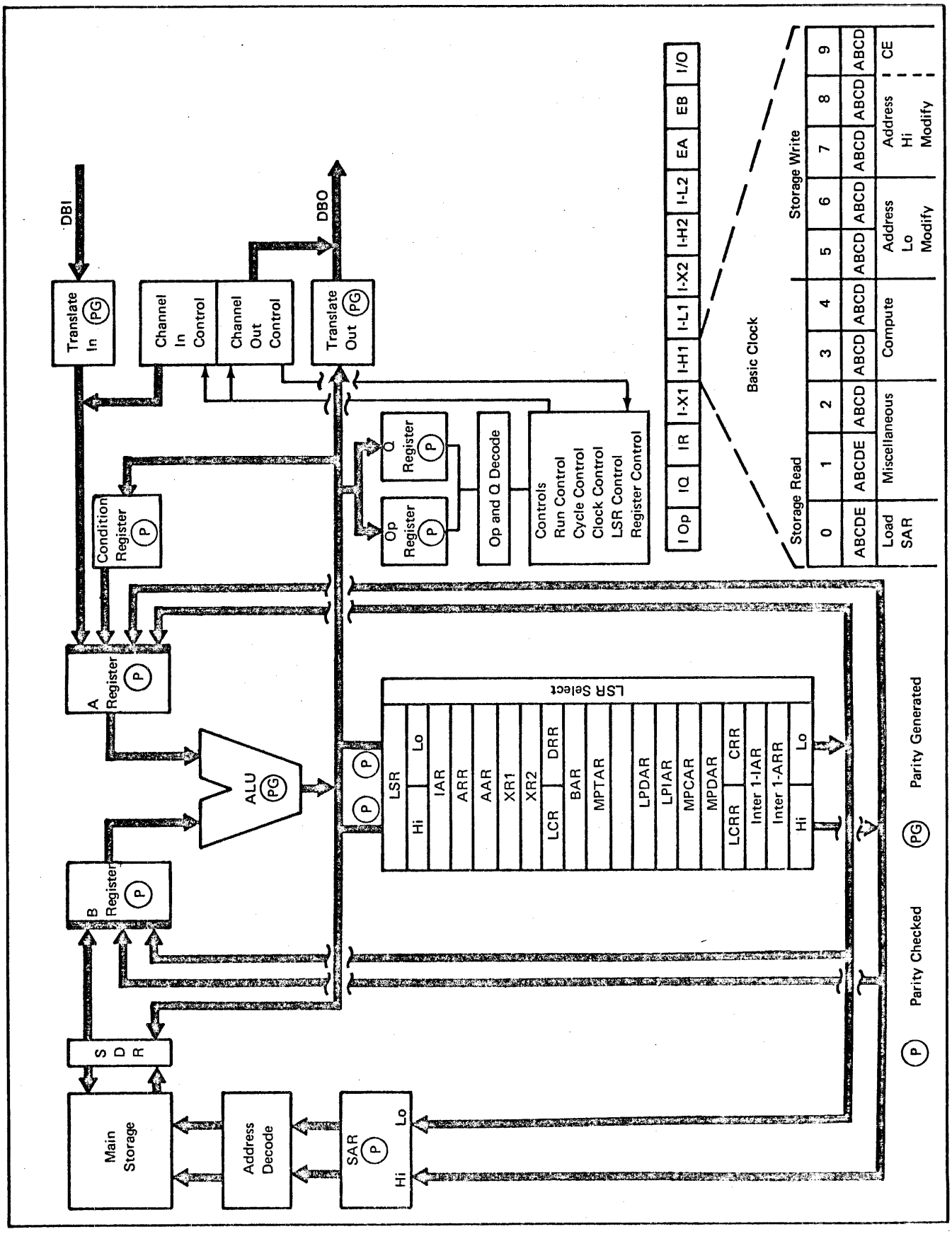
Output of the ALU is available to the I/O attachments on the data bus out (DBO) in one of two forms; either translated from EBCDIC (Extended Binary Coded Decimal Interchange Code) to System/3 card code, or straight from the ALU. The ALU output is also available to the storage data register (SDR) for entry into main storage, to the op and Q registers for instruction decode, to the condition register (CR), and to the local storage registers (LSR) for temporary storage.

Parity Checking

A check for dropped or extra bits during data transfer is accomplished by checking for an odd number of bits after the transfer. Figure 1-12 shows the parity checking (P) and parity generating (PG) points. Chapter 2 of the FEMD shows error checking circuits, and Chapter 2 of the FEMM gives a detail list of all error conditions and their causes.

Check ALU

Correct output parity is generated in the check ALU since the parity of the ALU does not stay constant with the inputs. After the data leaves the A and B registers, it can be altered by the decimal and binary complement circuits, the ALU, the decimal correct circuits, and the sign control circuits. The parity changes caused by all these must be considered when determining the parity of the ALU output.



- Figure 1-12. CPU Data Flow

CPU Timing

The basic unit of time in the CPU is a 1.52 us machine cycle, and based on the speed of the main storage unit. Each machine cycle contains a storage read and a storage write time.

The machine cycle is divided into nine clock times known as clock 0 through clock 8 (Figure 1-13). Clock 0 and clock 1 are each 200 ns long and all of the remaining clock times are 160 ns in length. Each clock time is divided into 40 ns phases. A clock 9 time consisting of four phase pulses is taken each cycle during system reset, step mode, alter SAR, and alter/display storage with the storage test switch in its step position.

Each machine cycle is divided into five functional time periods. The first, clock 0, is address time. This is the time when the address LSR is selected and sent to the SAR. During I/O cycles the I/O device attachment provides the selection of the LSR assigned as its address register.

The second functional time is known as the miscellaneous period. After the CPU has addressed main storage for a read operation, there is a delay before the output is available. This delay provides time to process other data through the ALU. The data processed is determined by the purpose of the machine cycle and as the name implies, there is no specific function during this period.

The third functional time is the compute period and is the time in which the main storage contents become available to the CPU in the SDR. The SDR at this time is sent to the B register unless it is blocked and is processed through the ALU with the contents of the A register. The A register may be loaded with zero if the SDR content is to be transferred through the ALU unchanged, or the A register may contain a modifier if the data from storage is to be modified or tested. The results of the modification are available on the ALU output for transfer into the SDR for storage. This time is also used to transfer data through the A register, ALU, and into the SDR.

The fourth and fifth functional time periods are address modification periods known as modify lo (clock 5-6) and modify hi (clock 7-8). During modify lo, the low half of the selected LSR is gated into the B register and into the ALU to be modified by the contents of the A register. During modify hi, the same thing is repeated for the high half of the selected LSR. During I/O cycles, the attachment must supply the modifier on the data bus in.

During clock 1 and 5 the CPU sends the read and write pulse to main storage.

Basic Clock

0	1	2	3	4	5	6	7	8	9
ABCDE	ABCDE	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD
Load SAR	Misc		Compute		Address Lo Modify		Address Hi Modify		CE

Figure 1-13. Clock Timing

CPU CLOCK

This is a group of binary triggers driven by a 25 MHz ocillator (OSC) to supply the basic timing pulses for the system. This clock provides timing pulses for both the CPU internal operations and the I/O attachments. These pulses (Figure 2-1) are nine (0-8) basic clock pulses (the CE can force a clock nine times for diagnostics). Clocks 0 and 1 are each 200 ns in length while the remaining clock times are 160 ns each in length. These clock times are further divided into 40 ns pulses called phase pulses. Clocks 0 and 1 have five phases: A through E; while the other clock times have only four: A through D.

Refer to FEMD 4-020 for a circuit description.

CYCLE CONTROLS

These are a group of triggers, controlled by the contents of the op code, which determine what machine cycles are needed to execute an operation.

Refer to FEMD 4-030 for a circuit description.

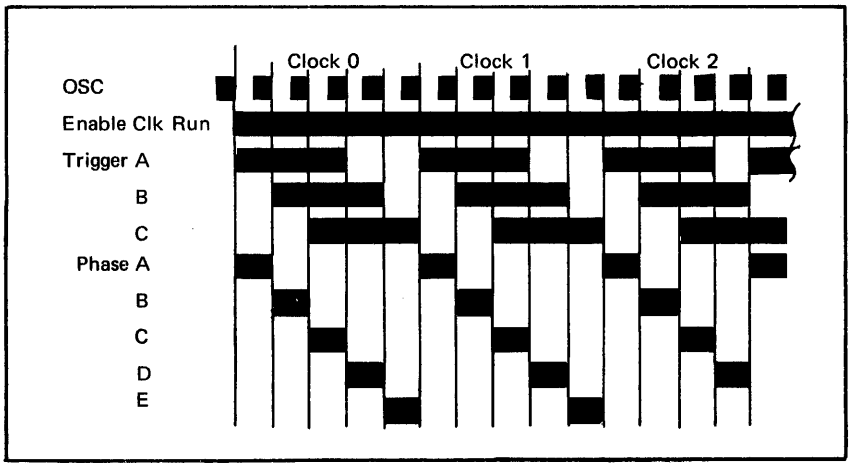


Figure 2-1. CPU Clock Timing

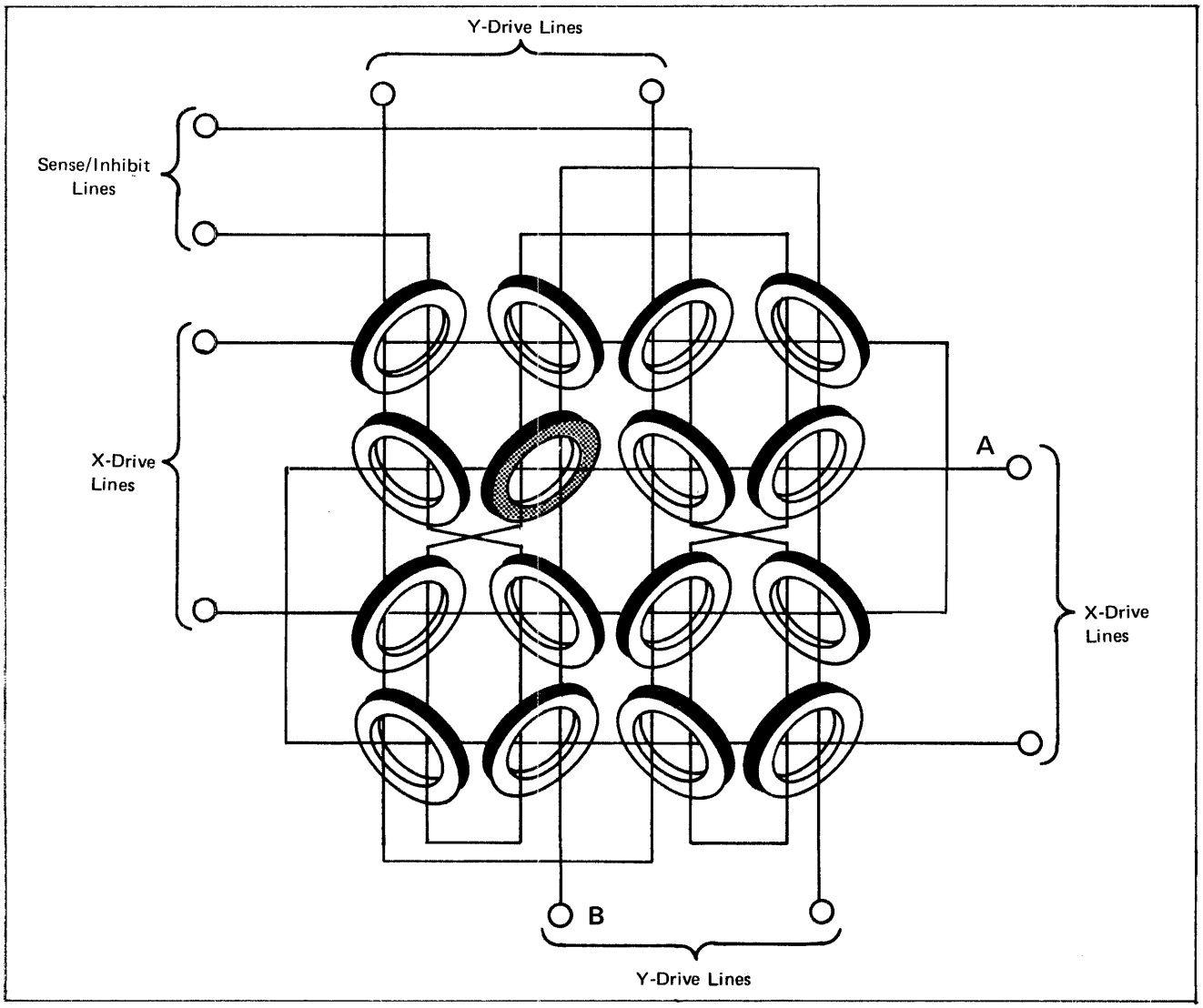
BRIDGE BASIC STORAGE MODULE

Storage Principles

The IBM 5410 CPU uses magnetic core storage. A magnetic core is a tiny ring of a special magnetic material. This core can be magnetized clockwise or counterclockwise by passing and electric current through a wire which is passed through the hole in the core. The core is magnetized clockwise by passing a current in one direction and counterclockwise by passing a current in the opposite direction. By assigning the values of 1 and 0 (*bit* and *no-bit*) to the two states of this core, we can store one bit of information.

For a useful storage device, this idea is expanded to include many cores. Figure 2-2 shows several cores in a core storage arrangement. The problem is to store a bit of information in only one of these cores. A specific amount of current is required to change a core from one state to the other; therefore, if two wires are used (X- and Y-drive lines) with half the required current in each, we can select the core to be set.

Figure 2-2 shows a 4 by 4 core plane. The X-drive lines run horizontally and the Y-drive lines run vertically. If half the required current passes through the X-drive line labeled A and half through the Y-drive line labeled B, only the shaded core is set (it is the only core receiving full current). Current through these two lines in one direction sets the core to bit status (write current). Current through the same two lines in the opposite direction resets the core to no-bit status (read current).



- Figure 2-2. Core Selection

When a current passes through these lines in the read direction and the core is already reset (no-bit), it will not change. However, if the core is set (bit), it will change status. We must be able to detect this change of status to know whether the core contained a bit or no-bit. This is done by passing one wire (sense line) through all the cores. When a core changes status, it induces a voltage in the sense line and this voltage is sensed as a bit.

The 5410 arrays contain either five, nine, or 18 core planes. Every plane has two separate inhibit windings. An inhibit winding passes through every core in the plane. When write current passes through X- and Y-drive lines, it passes through the selected cores of all planes. If we do not want to write a bit in a specific plane, current passes through the inhibit winding. The inhibit current is equal to and in the opposite direction of the current in the X-drive line. This cancels the effect of the current in the X-drive line and no-bit is written in that plane.

In the 5410, the same wire handles the sense and inhibit functions.

5410 BSMs

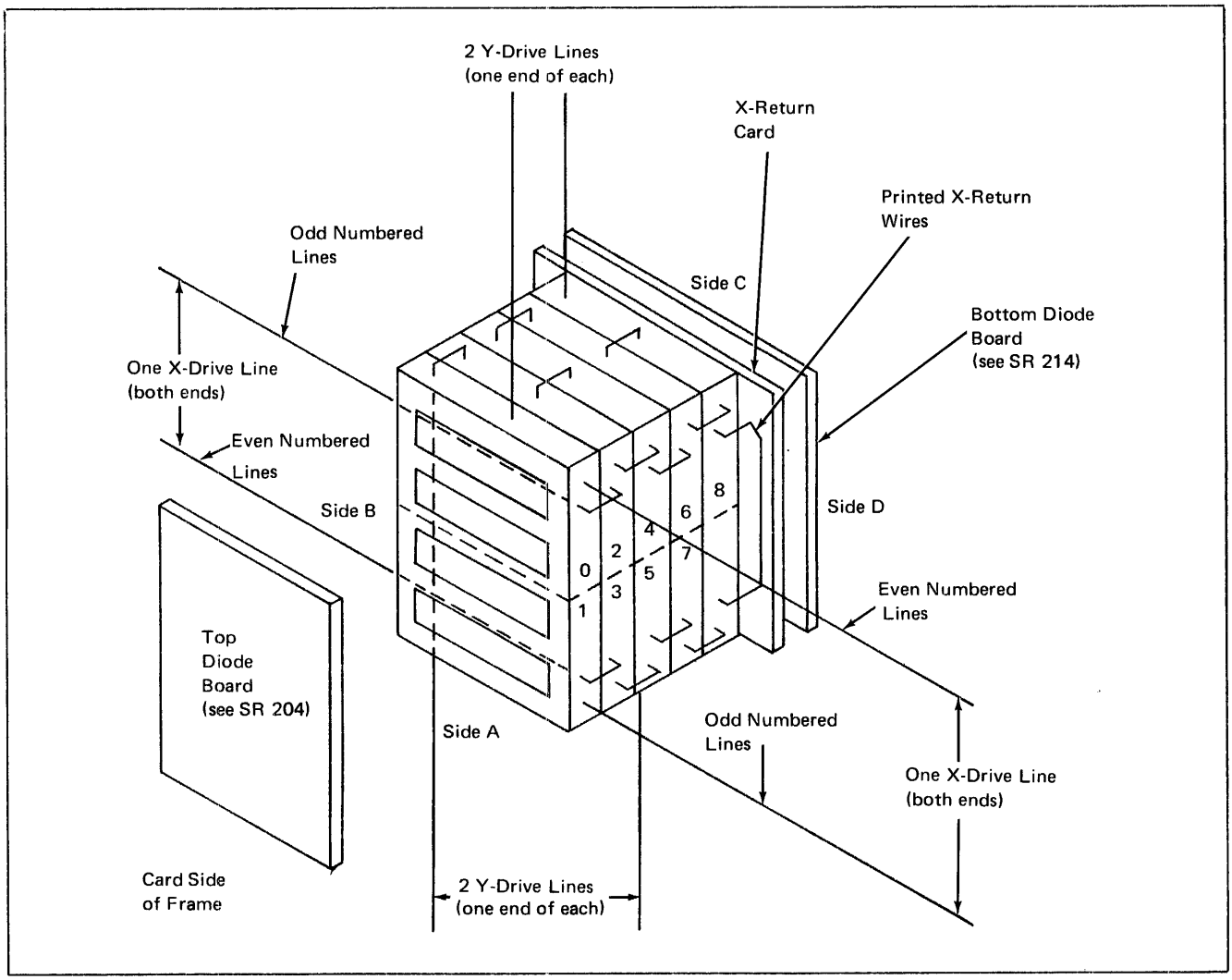
Several core planes comprise a core array. Arrays with their drivers, control circuits, diodes, sense data latches (which are also called the storage data register–SDR), and all associated wiring attached are called basic storage modules (BSMs).

If 8K, 12K, or 16K of storage is installed in the CPU, it is located in one 8K or 16K BSM at 01A1-B4. If 24K or 32K of storage is installed in the CPU, three types of BSMs can be used (Figure 2-2A). For early systems, 24K of storage uses one 8K BSM and one 16K BSM chained together; 32K of storage uses two BSMs chained together. For later systems, a 32K BSM is used for both 24K and 32K of storage. When two 16K BSMs are chained (dual BSMs), they are at 01A1-B4 and 01A1-A4 of the CPU. When one 32K BSM is used, it is at 01A1-B4 of the CPU.

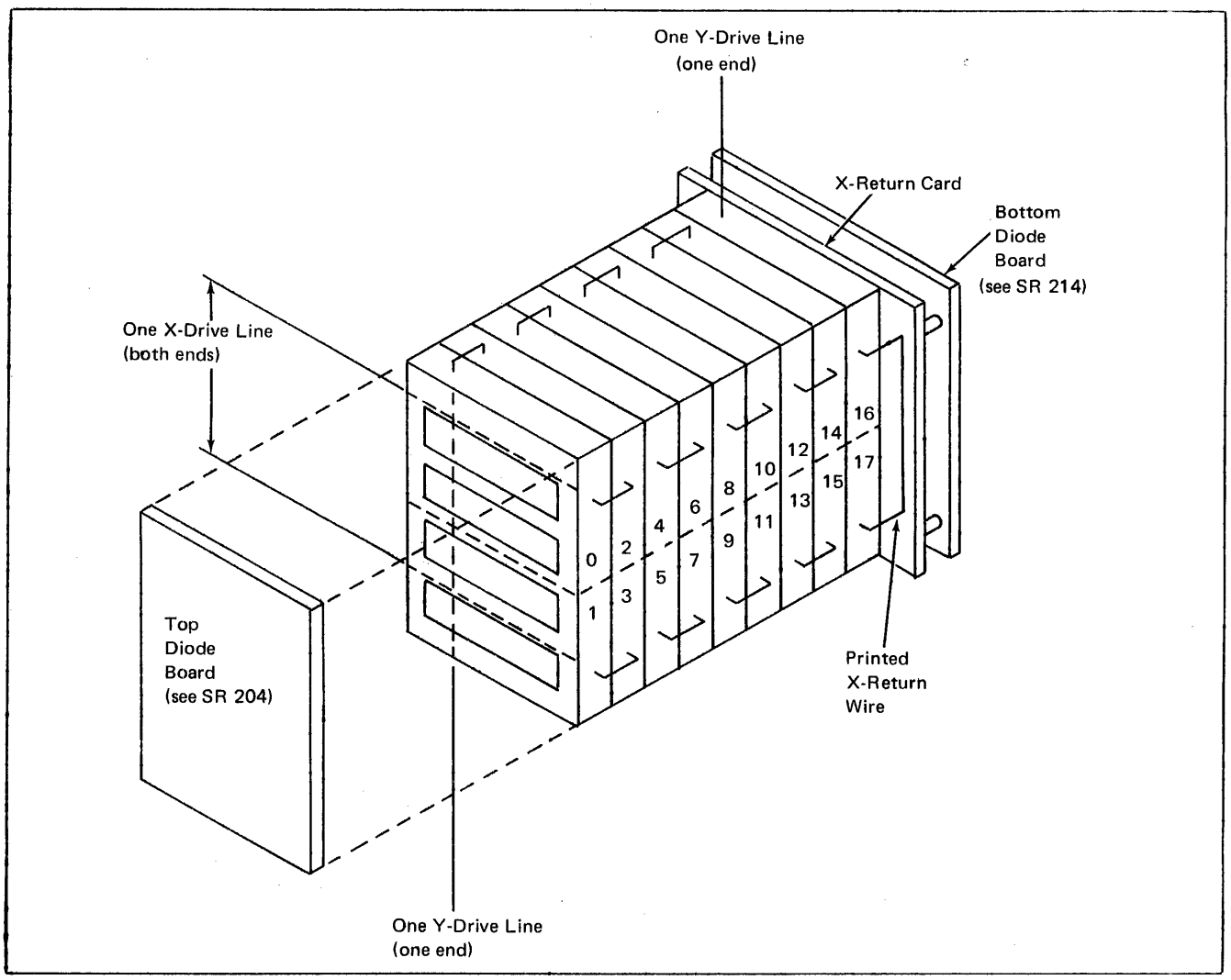
Storage Capacity	BSM Sizes Required		
	8K	16K	32K
8K	1		
12K		1	
16K		1	
24K	1 and	1 or	1
32K		2 or	1

● Figure 2-2A. Storage Capacity–BSM Requirements

Each core plane contains 16,384 (16K) cores at the intersection of 128 X-drive lines and 128 Y-drive lines. The core plane also has a sense/inhibit wire (S/Z winding) which runs parallel to the X-drive line. Thus, three wires go through each core: one X-drive line, one Y-drive line, and one S/Z winding. Each plane contains two S/Z segments (separate windings), each representing one data bit. For 8K and 16K BSMs, each X-drive line passes through both S/Z segments via the X-return card at the bottom of the array (Figures 2-3 and 2-4). Thus, when one X-drive line and one Y-drive line are active, two cores on the plane experience coincident currents (one in each segment). This means that each plane contains two bits for each of the possible X/Y addresses. Since there are 16K cores on the plane, they can all be addressed by 8K unique addresses provided by SAR bits 3-15.



- Figure 2-3. 8K Array Assembly



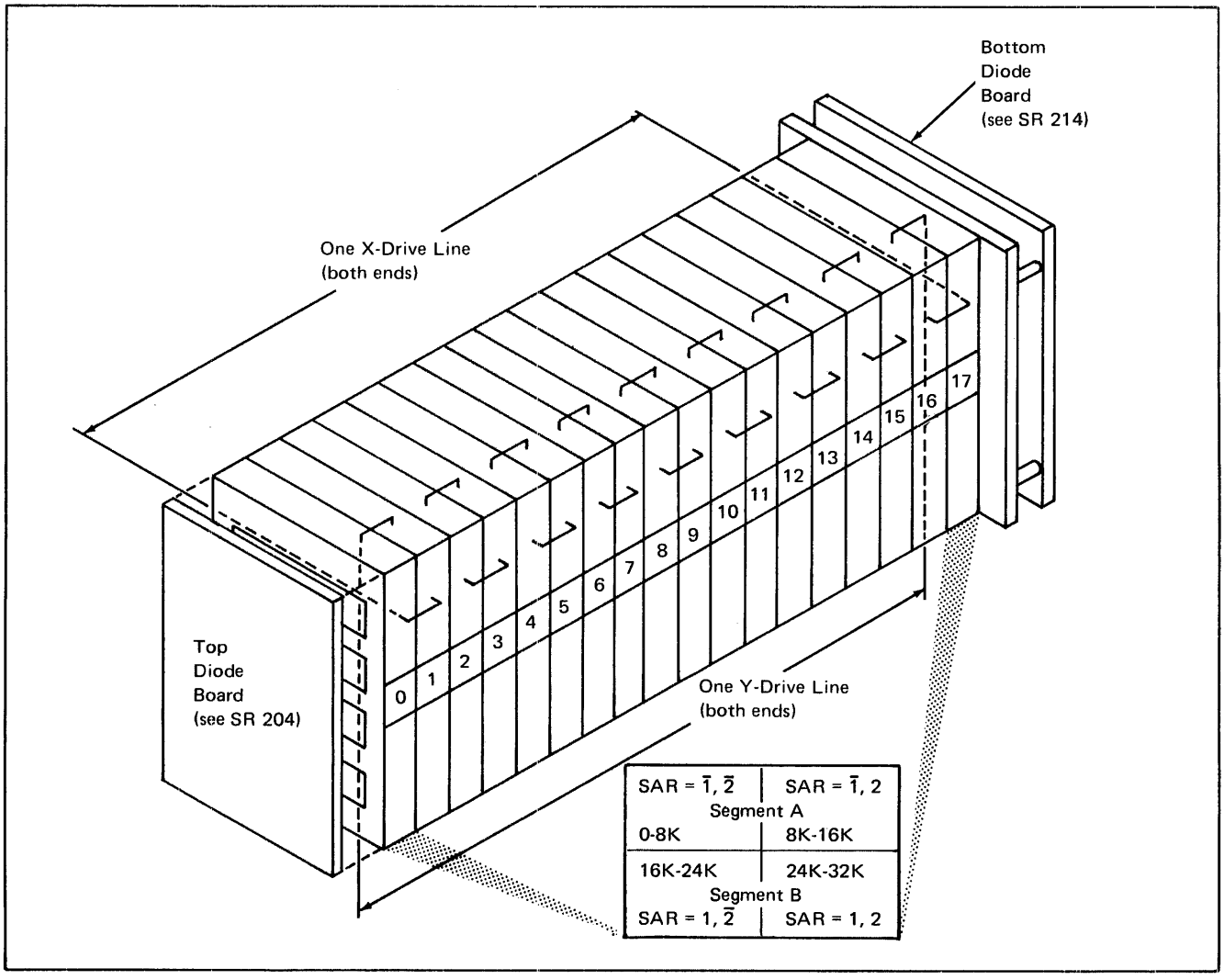
- Figure 2-4. 16K Array Assembly

Figure 2-4A shows a 32K BSM. Each X-drive line passes through one S/Z segment (there is no X-return card). When one X-drive line and one Y-drive line are active, one of 16K cores on the plane has coincident current. This means that each plane has one bit for each of the possible X/Y addresses. SAR bits 1 and 3-15 provide 16K unique addresses.

8K Basic Storage Module

An 8K byte BSM has five core planes (Figure 2-3). The first plane forms bits 0-1, the second plane forms bits 2-3, the third plane forms bits 4-5, the fourth plane forms bits 6-7, and half of the fifth plane forms bit 8. The lower half of the fifth plane is not used in an 8K byte BSM.

Each X-drive line travels through five planes before crossing to the second half of the core planes via an X-return card at the end of the array. This X-return card has printed lines that carry X-drive line current from one end of the X-drive line in the lower half of the array to the other end of the X-drive line in the upper half of the array. The winding pattern of the array is such that alternate X-drive lines are driven from opposite ends of the core plane (Figure 2-3).



- Figure 2-4A. 32K Array Assembly

A Y-drive line starts at either the top or bottom of the first core plane and is wound through each plane. It leaves the last plane on the opposite side from which it entered. Thus, if drive current flows through one X-drive line and one Y-drive line, ten cores will experience the coincident drive current necessary to affect the cores. The tenth core, in the lower half of the last plane, experiences coincident drive drive current, but this core output is not sensed because the S/Z segment is not used.

Two cores are addressed in every plane during a read operation and during a write operation. During a read operation, nine bits are sent to the sense data latches (which are also called the storage data register–SDR). During a write operation, either the same nine bits or nine different bits (storing new data) from the ALU are written back into the core planes.

Each drive line (X or Y) is connected to the array through diodes on the top diode board or the bottom diode board (Figure 2-5). These diodes are part of the gate and selection system described under "X/Y Drive System" in this chapter.

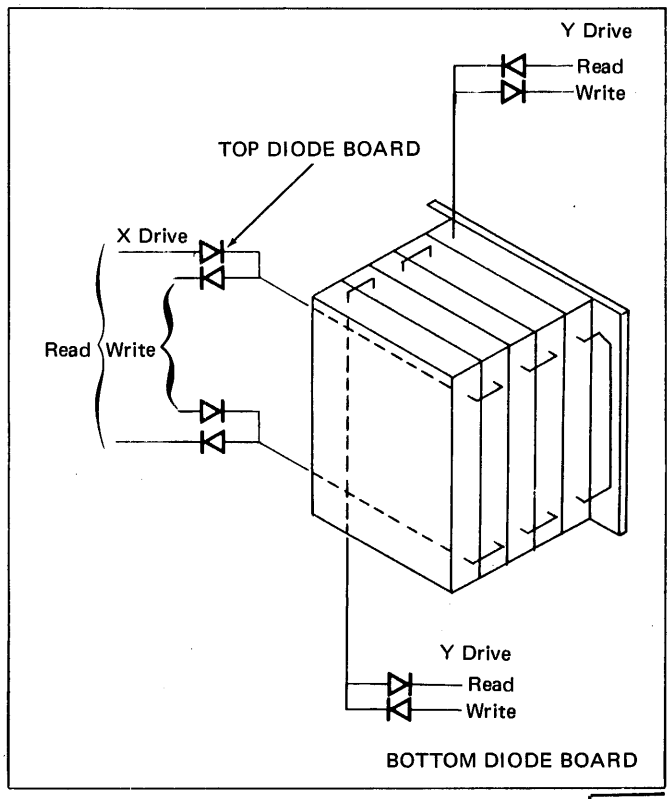


Figure 2-5. Diodes X- and Y-Drive Lines

16K Basic Storage Module

A 16K byte BSM has nine core planes (Figure 2-4). Two cores are addressed in every plane during a read operation and during a write operation.

All S/Z segments are used, providing 18 bits of data at each address. As in the 8K BSM, there are 8K unique addresses available. With the addition of one more address bit (SAR bit 2), 16K total addresses become available.

During a read operation, 18 bits are sent to the sense data latches (SDR). SAR bit 2 is used in byte control circuits which causes either the first nine bits or the second nine bits to be gated out from the SDR to the B register during the read operation.

During a write operation if 'store new' is active, SAR bit 2 determines which nine of the 18 bits are replaced with nine new bits from the ALU. On the other hand, during a write operation if 'store new' is not active, all 18 bits from the SDR are written back into storage.

32K Basic Storage Module

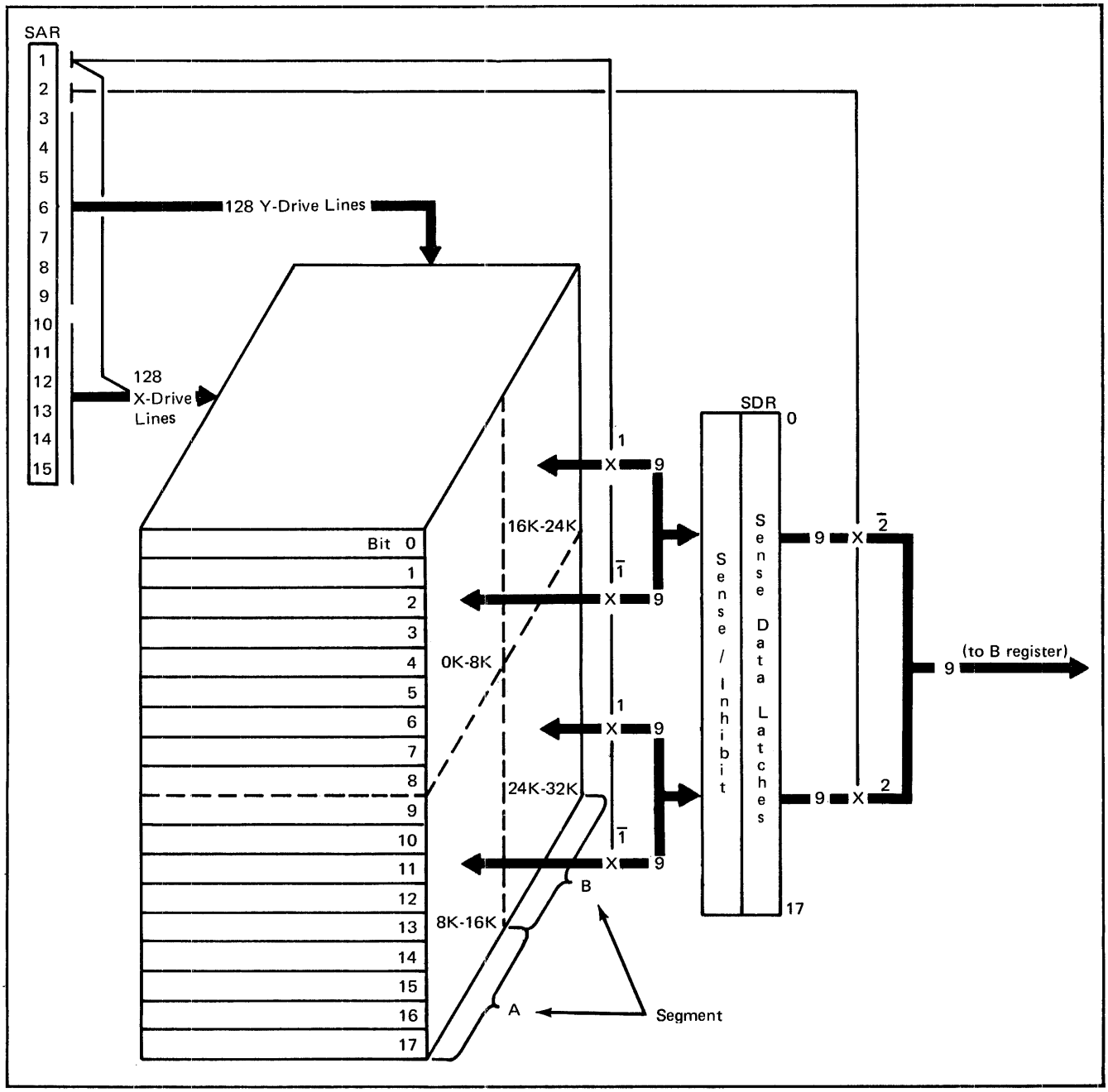
A 32K BSM has 18 core planes (Figures 2-4A and 2-5A). One core is addressed in every plane during a read operation and during a write operation. Each plane is divided into two segments: segment A and segment B. SAR bit 1 is used for segment control which selects either segment A or segment B to be read out of or written into. During a read operation, 18 bits are sent to the sense data latches (SDR). Sense data latches 0-8 contain byte 1; sense data latches 9-17 contain byte 2. SAR bit 2 is used for byte control which selects either byte 1 or byte 2 to be gated from the sense data latches to the B register during a read operation.

During a write operation if 'store new' is active, SAR bit 2 determines which nine of the 18 bits are replaced with nine new bits from the ALU. On the other hand, during a write operation if 'store new' is not active, all 18 bits from the SDR are written back into storage.

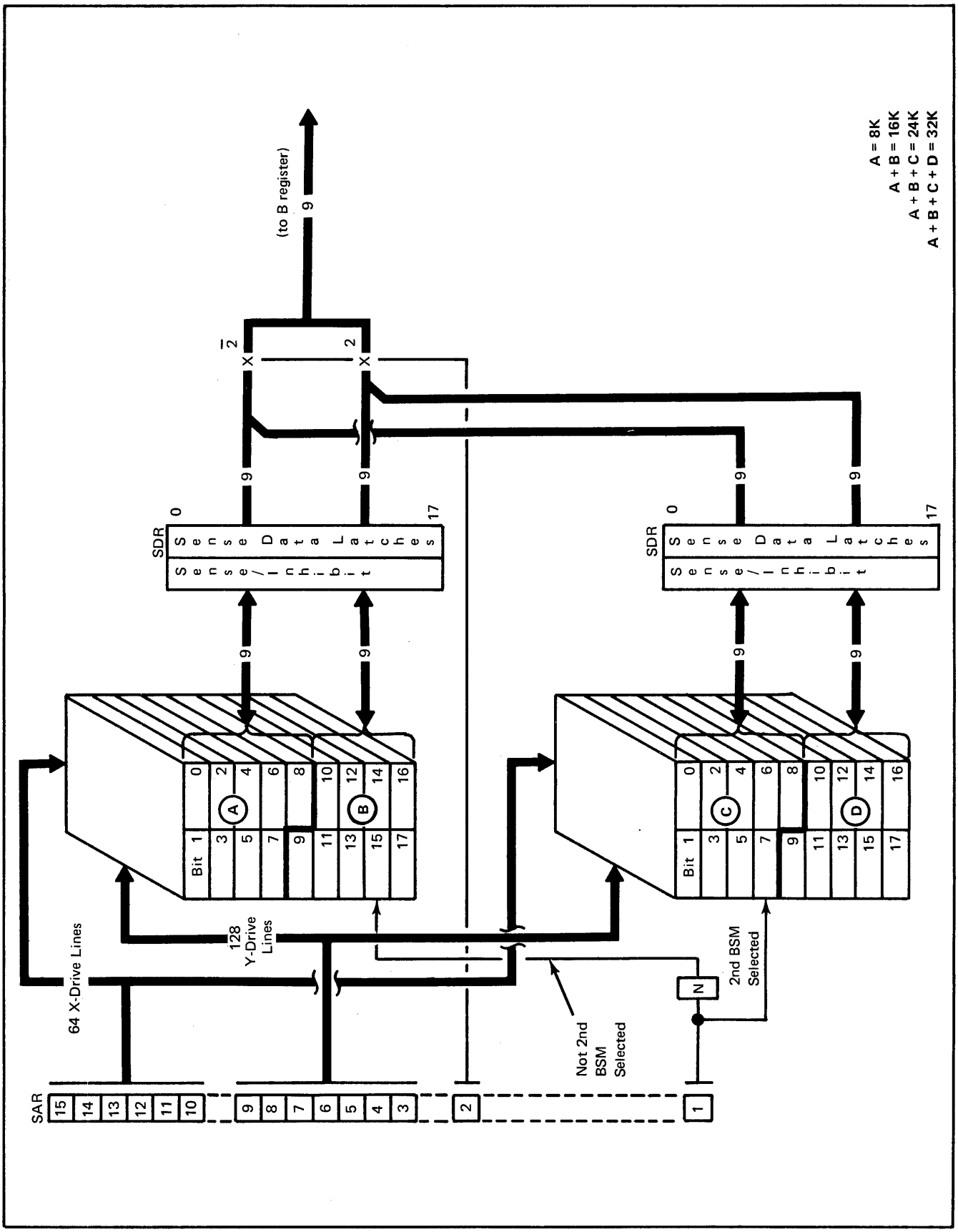
Chained BSMs

For early systems, 24K of storage has one 8K BSM chained to one 16K BSM, and 32K of storage has two 16K BSMs chained together (Figure 2-5B).

Chaining involves cabling a second BSM to the first BSM, adding or removing terminator cards, and removing the byte control card from the second BSM.



- Figure 2-5A. 32K BSM



- Figure 2-5B. Two 16K BSMs (Chained)

Addressing System

The storage address register (SAR) in the CPU, provides storage address bits. The address bit lines must be held active throughout the read and write operations.

The 8K BSM uses SAR bits 3-15 to access all the addresses in the 8K of storage (Figure 2-6).

The 16K BSM requires the use of SAR bit 2 as byte control. Byte control circuits determine which sense data latches (0-8 or 9-17) to use during a read operation and 'store new' operation.

When two BSMs are chained (Figure 2-5B), another address bit is required: SAR bit 1 (called '2nd BSM selected'). This bit determines if the storage timer is started in the low order or the high order BSM. The low order BSM is storage HEX address 0000-1FFF in CPU 01A-B4. The high order BSM is storage HEX address 2000-3FFF in CPU 01A-A4.

The output of data latches 0-8 in the high order BSM are dot-ORed with data latches 0-8 in the low order BSM. The output of data latches 9-17 in the high order BSM are dot-ORed with data latches 9-17 in the low order BSM. The low order BSM using SAR bit 2 performs byte control; this BSM controls the gating out of data for both BSMs to the B register.

The 32K BSM uses SAR bits 1 and 3-15 to access 16K unique addresses (Figures 2-4A and 2-6). SAR bit 1 is also used for segment control which gates the 18 bits to the sense data latches during a read operation. SAR bit 2 is used for byte control. Byte control circuits determine which sense data latches (0-8 or 9-17) are sent during a read operation to the B register.

During a write operation if 'store new' is active, SAR bit 2 determines which nine of the 18 bits are replaced with nine new bits from the ALU.

X/Y Drive System

The X/Y drive system uses:

- Current sources
- Gate and selection system

The X/Y drive system selects and passes current through one X-drive line and one Y-drive line. Each of these currents is one-half the current necessary to cause a core to flip. Consequently, cores at the intersection of the selected lines are the only cores which can be affected. Also, during the read cycle, cores in the logical one state are the only one which can flip. A third wire in each core, the S/Z winding, senses the cores which flip from the one state to the zero state.

Since all the logical ones at the addressed location are changed to logical zeros on a read cycle, this is called destructive readout. These bits are now in the SDR and may be used to restore the cores by reversing the X- and Y-drive currents at the same addressed location during the write cycle. (New data may be placed in the SDR prior to the write cycle if this is a store new operation.)

Zeros in the SDR are used to activate inhibit (Z) drivers which pass current through the S/Z winding in a direction which opposes the X-drive line current and prevents the corresponding cores from flipping to the one state.

Capacity	Version	SAR Bits 1	2	3 - 15
8K	(1 BSM)			3 - 15
12K	(1 BSM)		*	3 - 15
16K	(1 BSM)		*	3 - 15
24K-32K	(2 BSMs chained)	●	*	3 - 15
24K-32K	(1 BSM)	□	*	3 - 15

● 2nd BSM Selected
* Byte Control
□ Selects either X Decode Hi A or X Decode Hi B

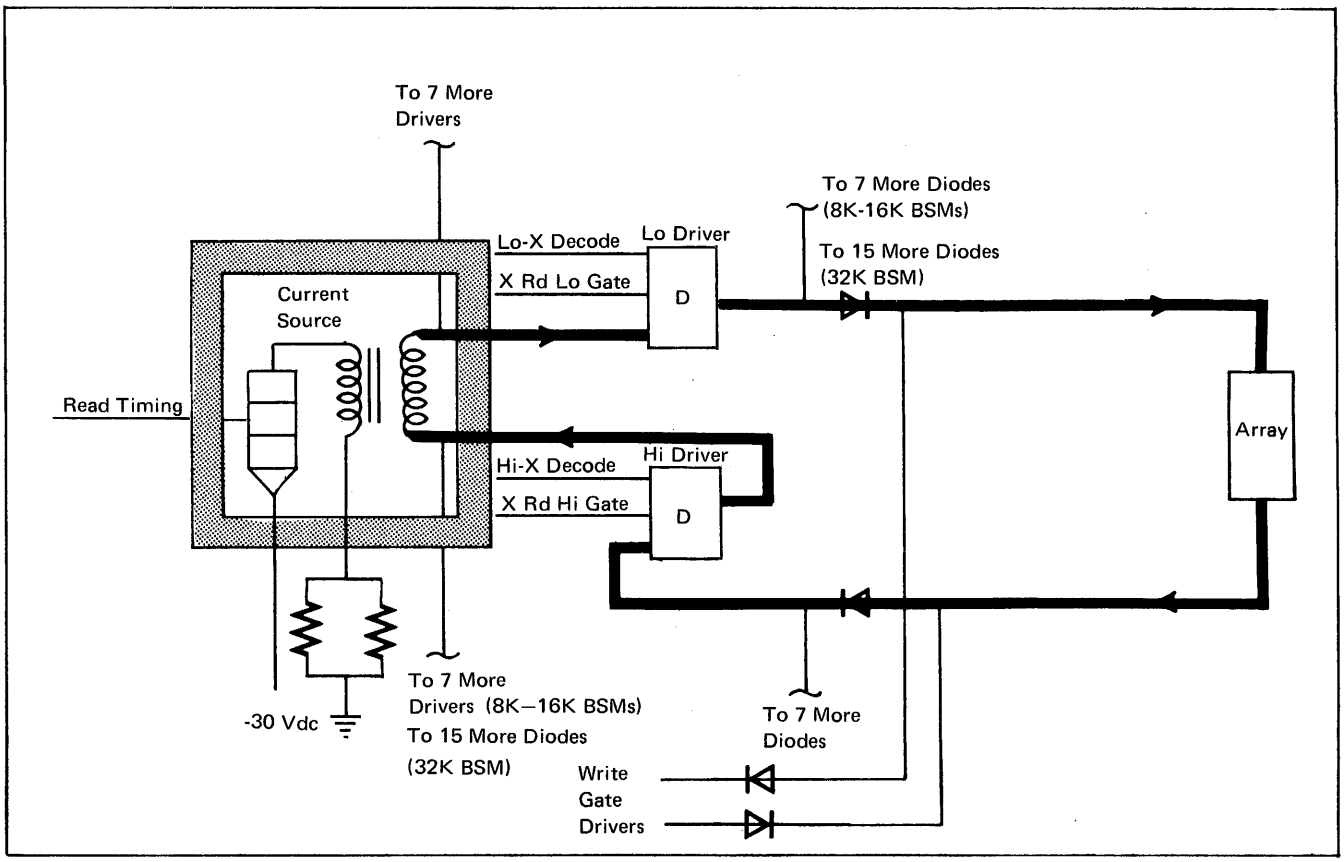
3 4 5 6	7 8 9	10 11 12	13 14 15
Y Decode Hi	Y Decode Lo	X Decode Hi A	X Decode Lo
		32K BSM → X Decode Hi B	

Figure 2-6. Addressing System

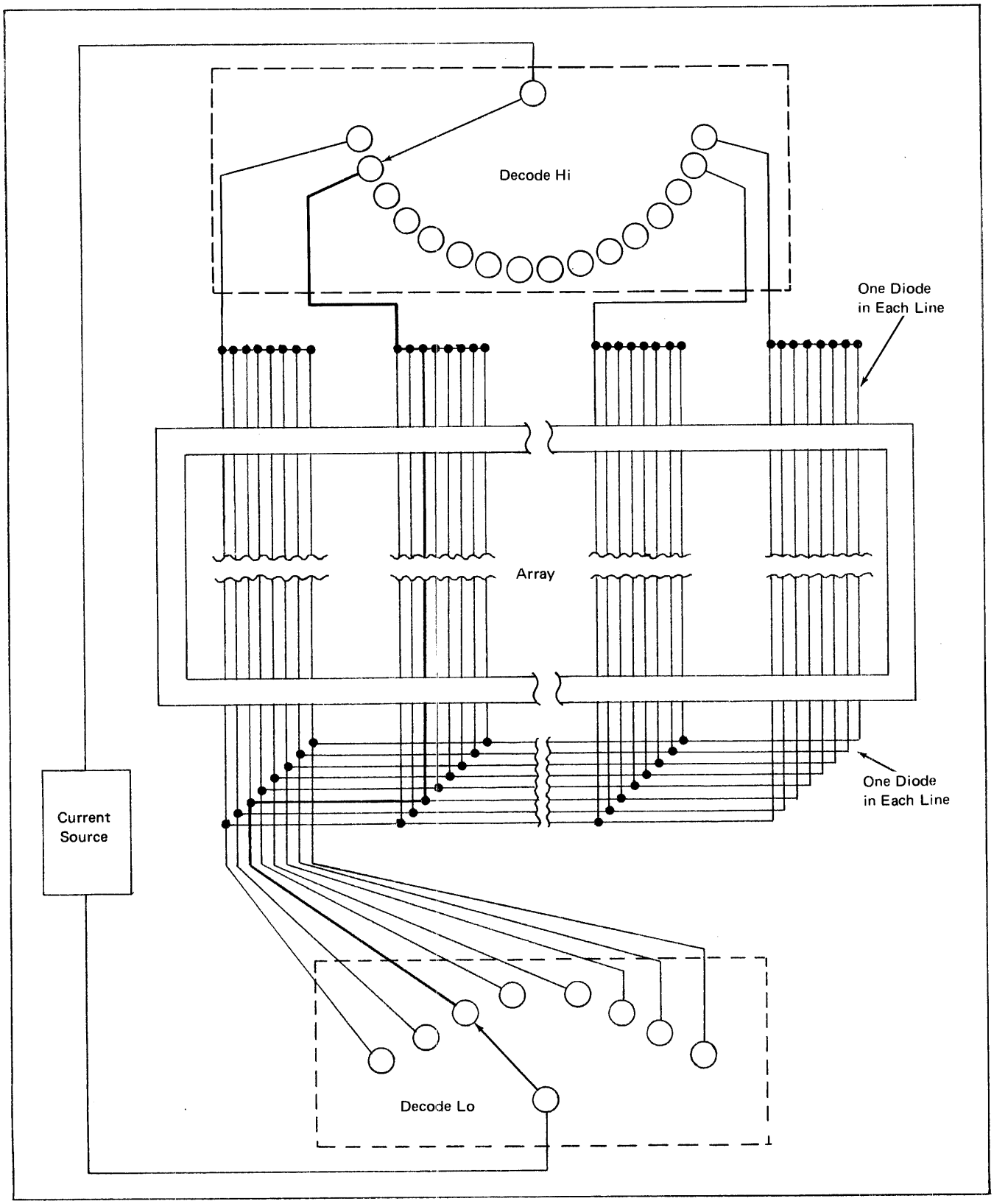
Current Sources: Supply drive current to the X- and Y-drive lines. There are four current sources in the BSM packaged on one card: X-read, X-write, Y-read, and Y-write. Each current source has a transformer primary and secondary. The primary is driven by a transistor controlled by cycle timing (Figure 2-7). The secondary is the current source for the X- and Y-drive line drivers.

Gate and Selection System: Directs drive current to one X-drive line and one Y-drive line. The gate and selection system acts like a switch at each end of the drive lines to direct the current source drive current to one drive line (Figures 2-8 and 2-8A).

During the read cycle, address decoders select a read Hi driver and a read Lo driver for one X-drive line (Figure 2-7), and one read Hi driver and one read Lo driver for one Y-drive line. During the write cycle, the same address decoders select one write Hi driver and one write Lo driver for the same X-drive line, and one write Hi driver and one write Lo driver for the same Y-drive line. These drivers, along with diodes in the X- and Y-drive lines to the array (Figure 2-9), cause current to flow in the X- and Y-drive lines in one direction during read and the opposite direction during write.

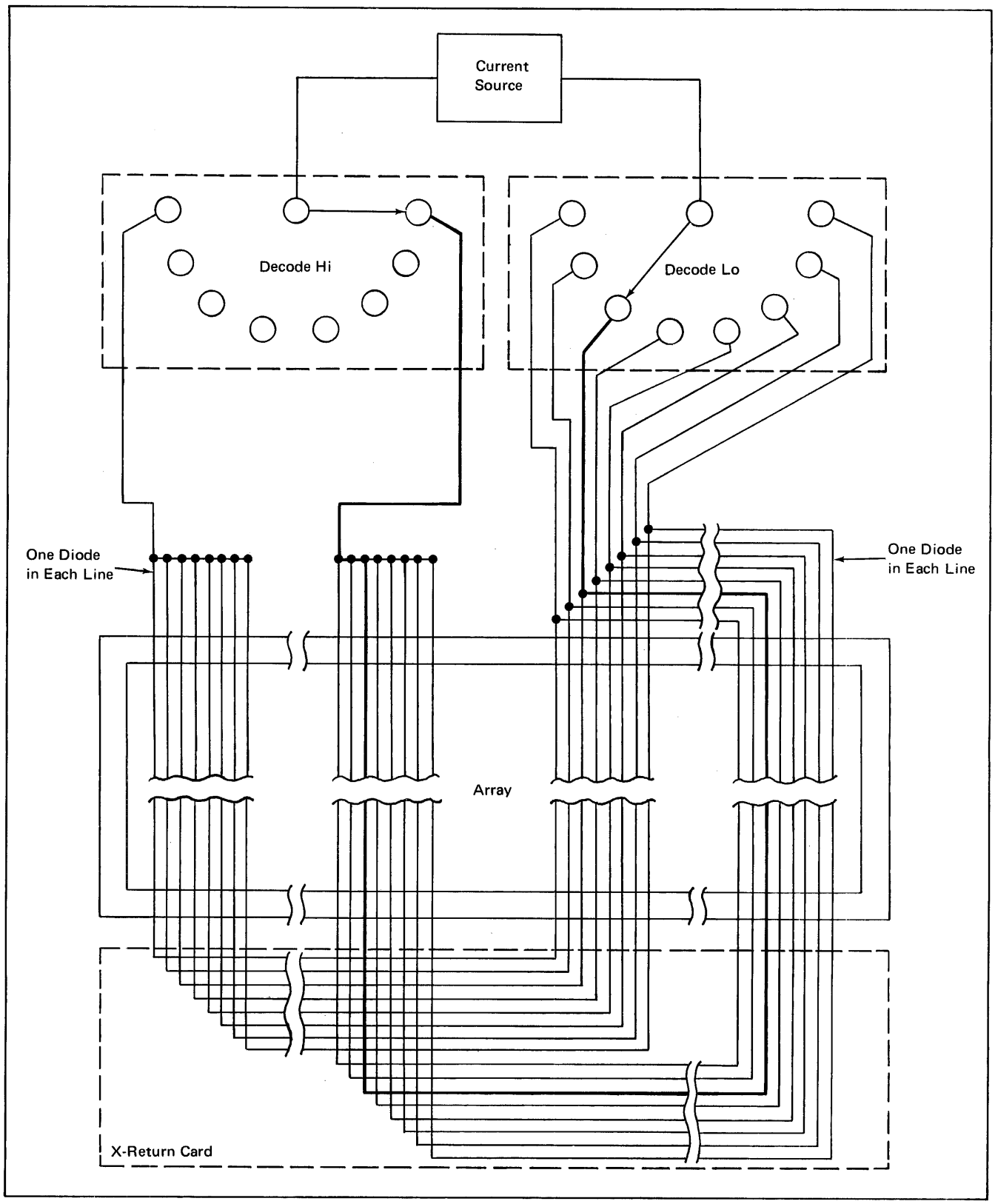


- Figure 2-7. X-Drive System



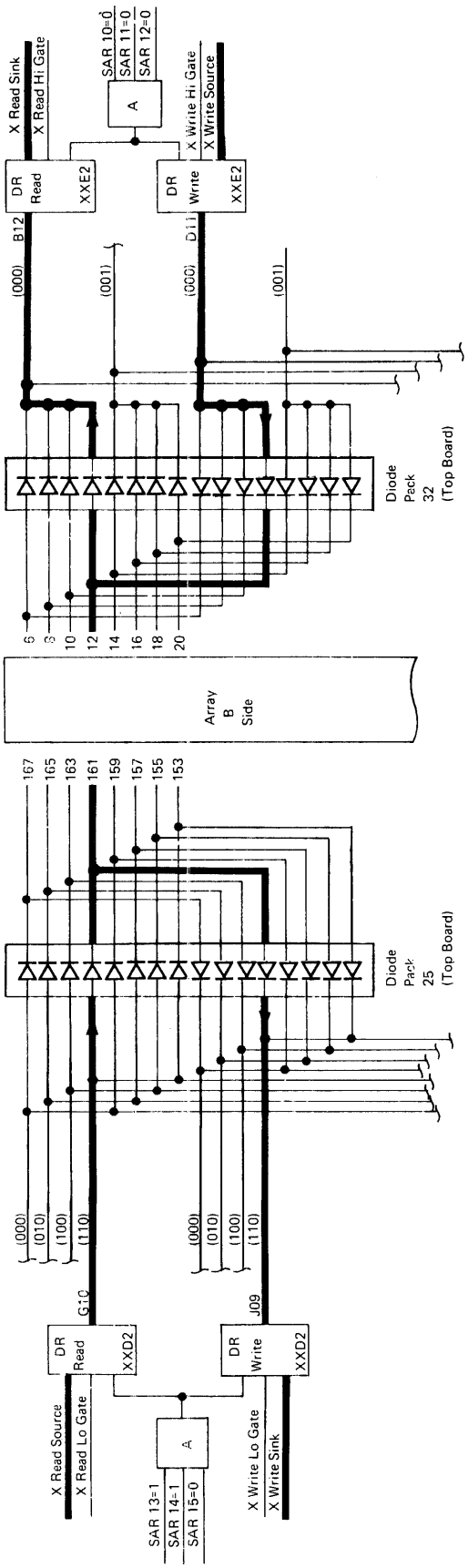
Note: Y Drive for 8K–16K BSM
X or Y Drive for 32K BSM

- Figure 2-8. Gate and Selection System X/Y (Simplified)



Note: X Drive for 8K–16K BSM

- Figure 2-8A. Gate and Selection System–X Only (Simplified)



• Figure 2-9. Array Diodes X-Read Drive (One Line–8K and 16K)

Readout

Readout from one addressable location in the BSM is accomplished by:

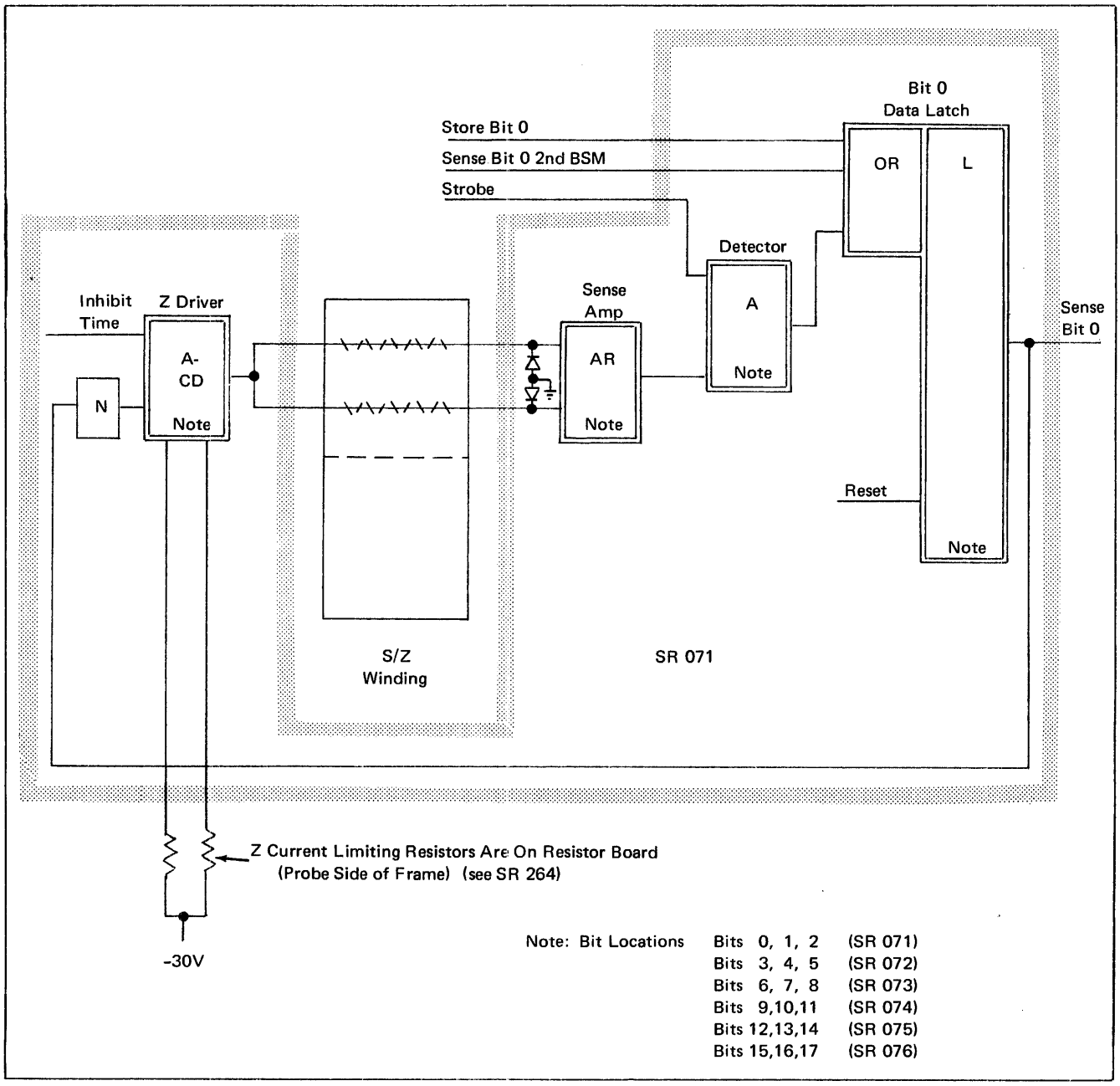
- Activating the X-read and Y-read current sources.
- Selecting, by address decode , two X-read drivers and two Y-read drivers.
- Sensing which of the selected cores have flipped from the logical one state to the zero state.
- Sending this data out via the interface, also saving it for the subsequent write cycle.

Sense/Inhibit (S/Z) Windings

- A combination S/Z winding is used (one for each data bit).
- Each winding goes through 8K cores (one-half of a plane) parallel to the X-drive lines.
- Pulses are generated on the S/Z winding when the cores change state during a read cycle. The sense amplifier circuits sense these pulses.
- During a write cycle, the S/Z winding can prevent cores from changing to ones.

During the read cycle, a core that switches induces a pulse into the S/Z winding. The sense amplifier senses a difference in voltage on both ends of the sense line and amplifies only the difference (Figure 2-10). Outputs from sense amplifiers are sent to detector circuits which, along with storage timer strobe pulses, distinguish between noise and one-bit signals and send the one-bit pulses to the SDR. During the read cycle, a core that does not switch (was a logical zero) induces no pulse in the sense winding and sets no SDR latch.

During the write cycle, if a logical one is to be stored in a core, the core is flipped by coincident X- and Y-drive currents. In this case, inhibit current does not flow in the inhibit segment (Figure 2-10). If a logical zero is to be stored, inhibit current must flow to oppose the magnetic effect of the X-drive current. With the absence of one bit from the SDR to Z driver input, the Z driver conducts and inhibit current flows. The effect of the inhibit current is to cancel the X-drive line current and the core remains in a logical zero state.



- Figure 2-10. Sense/Inhibit Logic (Bit 0)

16K or 32K BSM Byte Control

Every read operation reads out 18 bits (two bytes). Byte control circuits, controlled by SAR bit 2, determine which SDR latches (0-8 or 9-17) to send to the B register during a read operation (Figures 2-11 and 2-11A).

Byte control is not used for 8K BSMs.

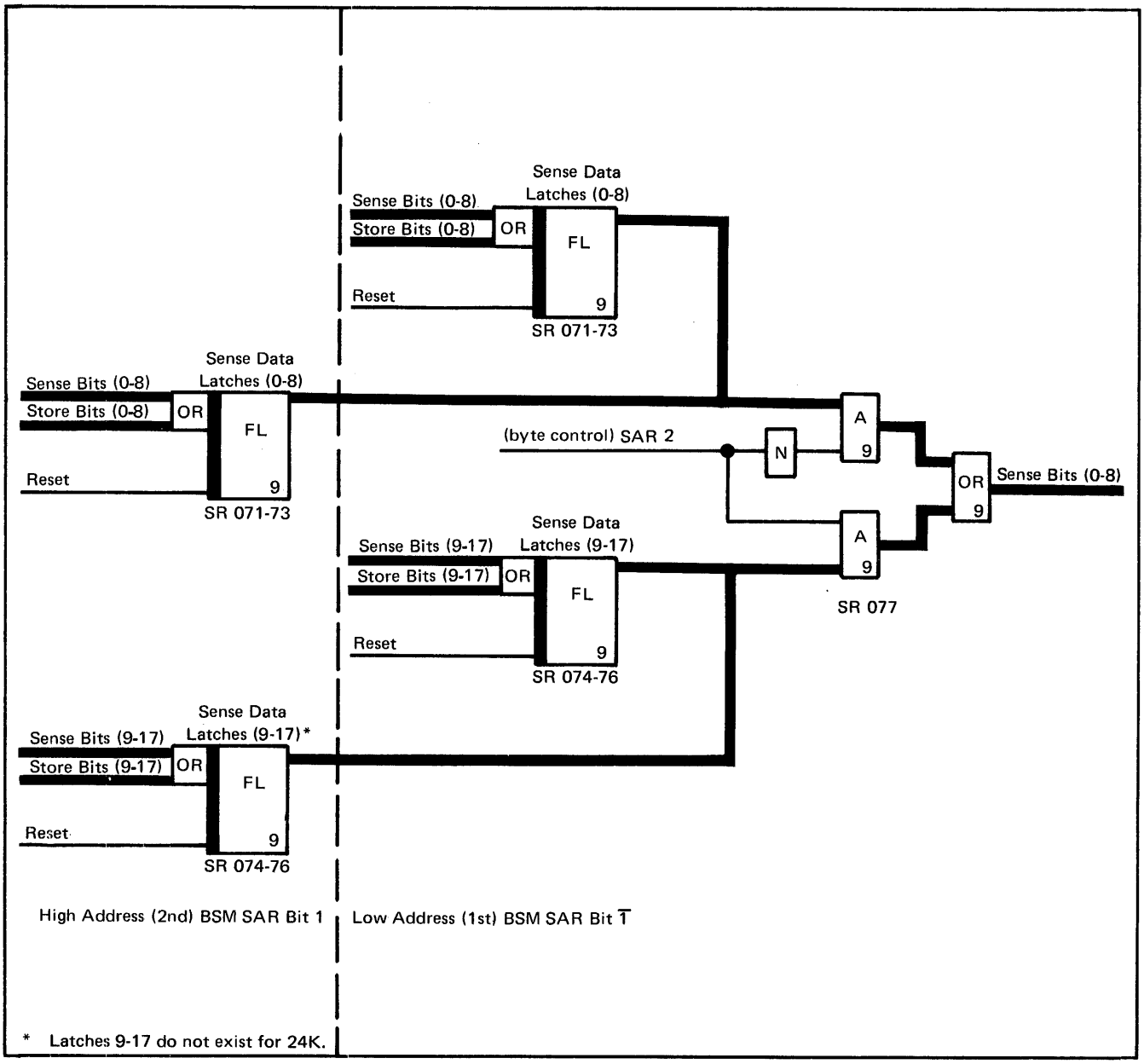
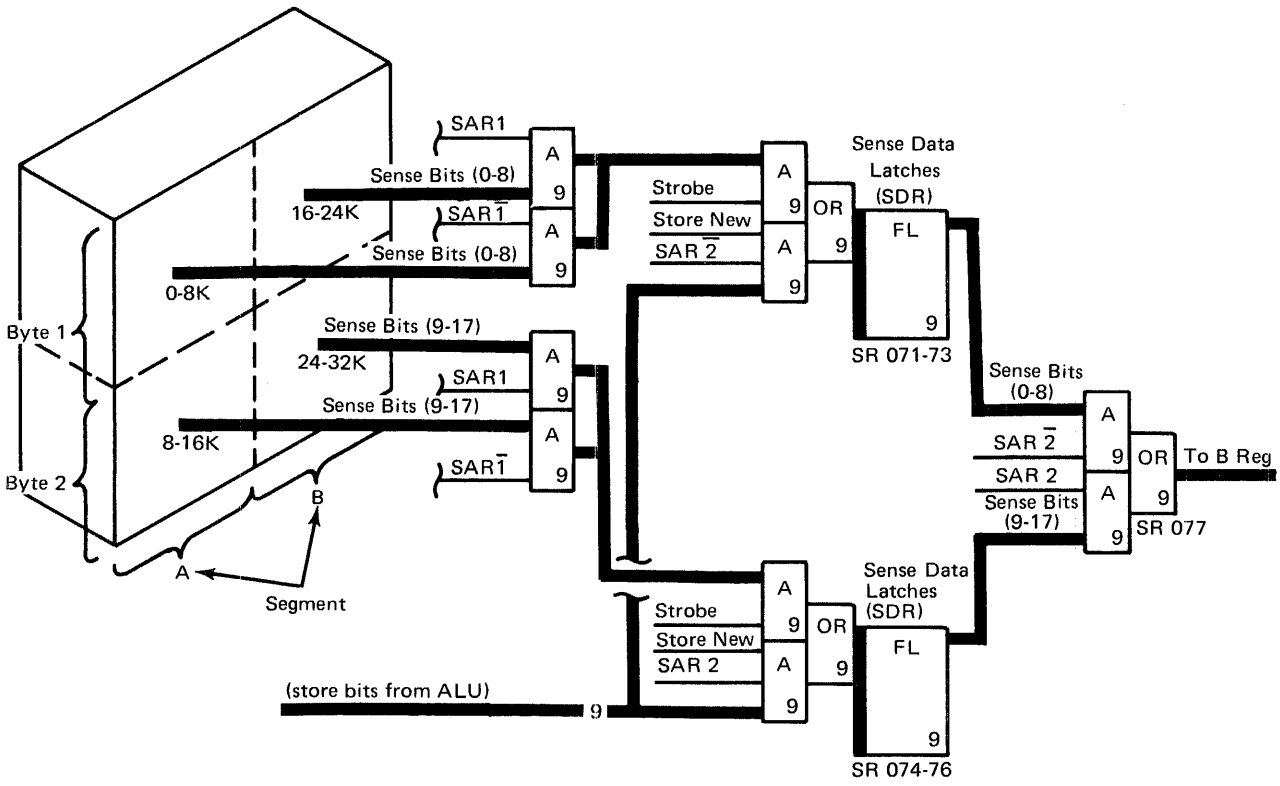


Figure 2-11. Byte Control (Dual BSMs)



- Figure 2-11A. Byte Control (32K BSM)

Write (Store)

A write cycle must follow every read cycle in order to restore the cores to their original status (before the read cycle). The read data bits are in the sense data latches (SDR) and serve as inputs to control the Z drivers. However, if this is a store new operation, 'store bit' lines from the interface set the SDR latches along with the control line 'store new'. The 'store new' line causes the SDR latches to be reset prior to their setting with new data. If SAR bit 2 was used for byte control on the readout, it is now used to generate store byte 1 or store byte 2. These two signals control the setting of the SDR latches 0-8 and 9-17 respectively. SAR bit 2 selects only one of these two bytes and the data contained in the other byte is gated back into storage.

Storage Cycle Timing

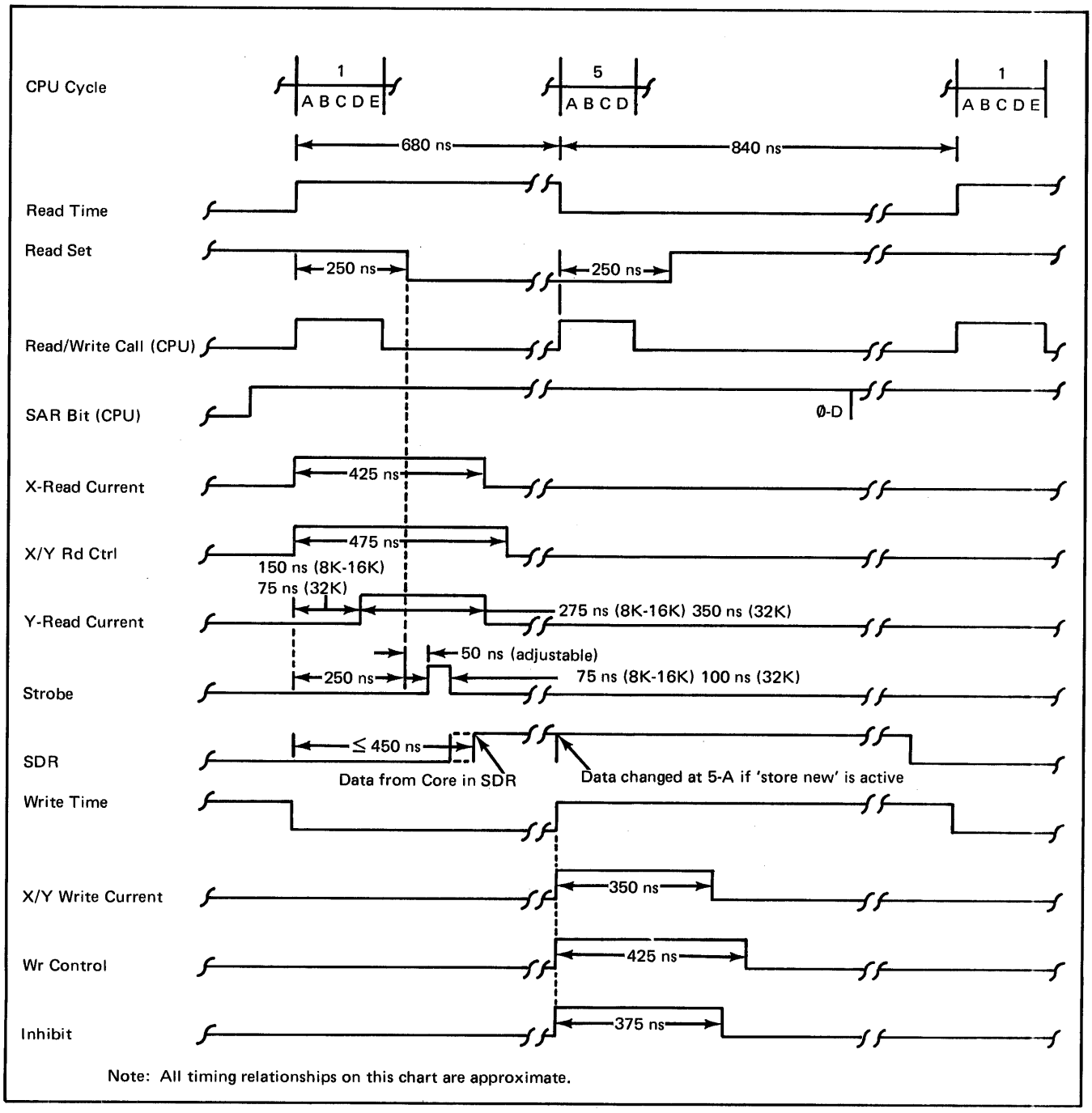
The BSM is an asynchronous unit; that is, once started, it runs independently. The CPU starts the storage cycle by issuing the read call/write call signal. This turns on the timing control latch and the read/write latch. The read/write latch on defines this cycle as readout cycle. Late in the readout cycle the set reset latch is turned on; though performing no function at this time, it ensures that the next read call/write call resets the read/write latch.

Therefore, the following cycle is a write cycle. During the write cycle, the set reset latch will be turned off, forcing the following cycle to be a readout cycle.

The system must allow about 600 ns for each cycle to be completed and also must hold the SAR (address lines) active for both cycles. (Figure 2-12 shows the cycle timings.)

Storage Timer

Each read call/write call signal turns on the timing control latch. This starts the storage timer which is a delay line tapped at 25 ns intervals. This timing control latch is turned off after 225 ns, so the duration of each delayed pulse is about 225 ns. These pulses are wired to AND circuits to provide timing signals.



- Figure 2-12. Storage Read/Write Cycle Timings

Interface

The BSM communicates with the system over a series of signal lines called the interface. All control signals, addresses, and data are transmitted over this interface (Figure 2-13).

Read Call/Write Call

Upon receipt of the first read call/write call signal, the storage timer of the BSM starts. The BSM goes through one read cycle and stops. Upon receipt of a second read call/write call signal, the BSM goes through a write cycle and stops. The first cycle following a system reset is always a read cycle. Thereafter, read and write cycles alternate.

Reset

The 'reset' line resets the storage timer latches which control the read/write cycles in the BSM; it also resets the SDR latches. This signal is present during a power-on sequence and every CPU clock 0 CD.

Store New

The 'store new' line resets the selected BSM SDR latches prior to it being set with new information at the beginning of a store cycle.

2nd BSM Select or SAR Bit 1

For dual BSM 24K or 32K systems, this line is called '2nd BSM select'. When the line is active, the high address BSM (01A-A4) is used; when it is inactive, the low address BSM (01A-B4) is used.

For 8K or 16K BSMs, this line is not wired on the CPU board (01A-B3) and thus is always inactive.

For 32K BSMs, this line is called 'SAR bit 1' and is an address bit.

SAR Bits

If a 32K single BSM is installed, 'SAR bit 1' inactive addresses segment A of the BSM or 'SAR bit 1' active addresses segment B of the BSM.

SAR bit 2 is used for byte control.

SAR bits 3-15 supply the address to the BSM addressing circuitry.

The SAR bit lines are active through both read and write cycles.

Store Bits

Store bit lines provide data input to the SDR during a 'store new' cycle.

Sense Bits

Sense bit lines carry the storage data out to the system. The lines become active within 450 ns after 'rd call/wr call' starts a read operation and remain active until changed either by 'store new' during the write operation or by the 'reset' line which is active during CPU clock 0 CD.

From System	From BSM
– READ CALL / WRITE CALL	
+ SYSTEM RESET	
+ STORE NEW	
+ 2nd BSM SELECT (SAR BIT 1)	
+ SAR BIT 2	
+ SAR BIT 3	
+ SAR BIT 4	
+ SAR BIT 5	
+ SAR BIT 6	
+ SAR BIT 7	
+ SAR BIT 8	
+ SAR BIT 9	
+ SAR BIT 10	
+ SAR BIT 11	
+ SAR BIT 12	
+ SAR BIT 13	
+ SAR BIT 14	
+ SAR BIT 15	
– STORE BIT 0	– SENSE BIT 0
– STORE BIT 1	– SENSE BIT 1
– STORE BIT 2	– SENSE BIT 2
– STORE BIT 3	– SENSE BIT 3
– STORE BIT 4	– SENSE BIT 4
– STORE BIT 5	– SENSE BIT 5
– STORE BIT 6	– SENSE BIT 6
– STORE BIT 7	– SENSE BIT 7
– STORE BIT 8	– SENSE BIT 8

• Figure 2-13. Storage Interface

Power Supply and Temperature Compensation

- DC voltages the BSM requires: +6V, -4V, -30V.
- DC voltages generated within the BSM: +3V, -14V.
- System airflow cools the BSM.

S/Z circuits use +3V (derived from the +6V input). -14V (derived from the -30V input) generates sense amplifier current.

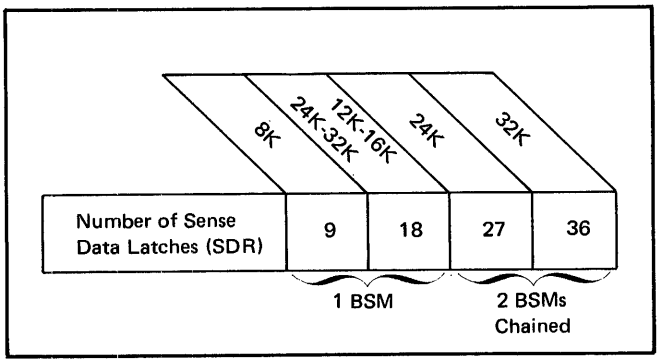
A special temperature compensated supply furnishes the -30V for the X, Y, and Z drivers. A thermistor near the core array senses the array temperature and sends any temperature variations to the -30V power supply. This input to the -30V supply regulates the voltage -75 mV for every 1° F temperature rise (-135 mV for every 1° C temperature rise).

STORAGE DATA REGISTER (SDR)

All data read from or written into CPU storage (BSM) is stored in the sense data latches which are in the BSM. These sense data latches are also called the Storage Data Register (SDR).

During a read operation, the storage strobe gates data read from storage into the SDR latches. Characteristics of ferrite memory cores and components vary in BSMs; therefore, the time that data is available to the SDR latches can vary. The strobe pulse is adjustable to compensate for these variations.

Depending on the storage capacity installed, there is a varying amount of SDR latches (Figure 2-14). However, no matter what the amount of SDR latches installed is, during a read operation, only one byte (nine SDR latches) is gated to the B register.



- Figure 2-14. Sense Data Latch Requirements

When an 8K BSM is installed, there are nine SDR latches (one byte) available to be gated to the B register during a read operation. If a 16K or 32K BSM is installed, there are 18 SDR latches available, but only one byte is gated to the B register (gated by SAR bit 2). If two chained BSMs are used to make up 24K or 32K of storage, there are SDR latches available in each BSM. During a read operation, 'SAR bit 1' selects either one of the two BSMs and its SDR latches. (Refer to FEMD 4-080 for storage data flow.) SAR bit 2 gates nine of these SDR latches (one byte) to the B register.

During a read operation, data flows from storage to the storage data register and then to the B register. During a write (store) operation when 'store new' is active, data is gated into the SDR latches from the ALU at clock 5. During a write operation if 'store new' is not active, all data from the SDR latches is returned to storage.

STORAGE ADDRESS REGISTER (SAR)

A two byte address contained in the storage address register (SAR) addresses main storage. A selected local storage register (LSR) provides two bytes to the SAR each machine cycle to serve as the address. A parity bit accompanies each byte to provide an 18-bit SAR. All bits are not always used. (See the description of SAR bits in the section "Interface.")

B REGISTER

The B register is a one-byte buffer to gate into the ALU all data which the contents of the A register can modify. The data in the SDR is normally gated into the B register every machine cycle, but can be inhibited if the operation requires. During an I/O cycle, the attachment has control of this gating.

A REGISTER

All data which modifies the contents of the B register in the ALU is buffered in the one-byte A register. The modifier comes from the local storage register (LSR), the condition register (CR), or from an I/O device on data bus in (DBI). For normal address modification, output of the LSRs are fed to the B register, and through the ALU while the A register provided the forced modifier.

All incoming I/O data is fed through the A register and into the ALU. There is no other path for the I/O data to enter the CPU or main storage.

ALU

The ALU is a multiple function unit which receives information from the A and B registers and performs the following functions with the A and B register data:

- Logical OR
- Logical AND
- Test for presence or absence of bits
- Pass B register through
- Pass A register through
- Binary subtract
- Binary add
- Decimal subtract
- Decimal add

The ALU operates on a full byte of information. Each register can supply either a full byte or one half of the byte depending upon the operation. The ALU is used four times during each machine cycle and is loaded each even clock CD time except 0 (Figure 2-15). The results of each computation are available from the latches while the components within the ALU are starting with the next computation.

The A and B registers, except the P bits, enter the ALU in parallel form and result in a single byte output. Each ALU position (bit 7 through bit 0) consists of a group of AND, OR, and exclusive OR blocks which give the correct output for the desired function. Because the parity of the result does not stay constant with the inputs, correct parity is generated for the results.

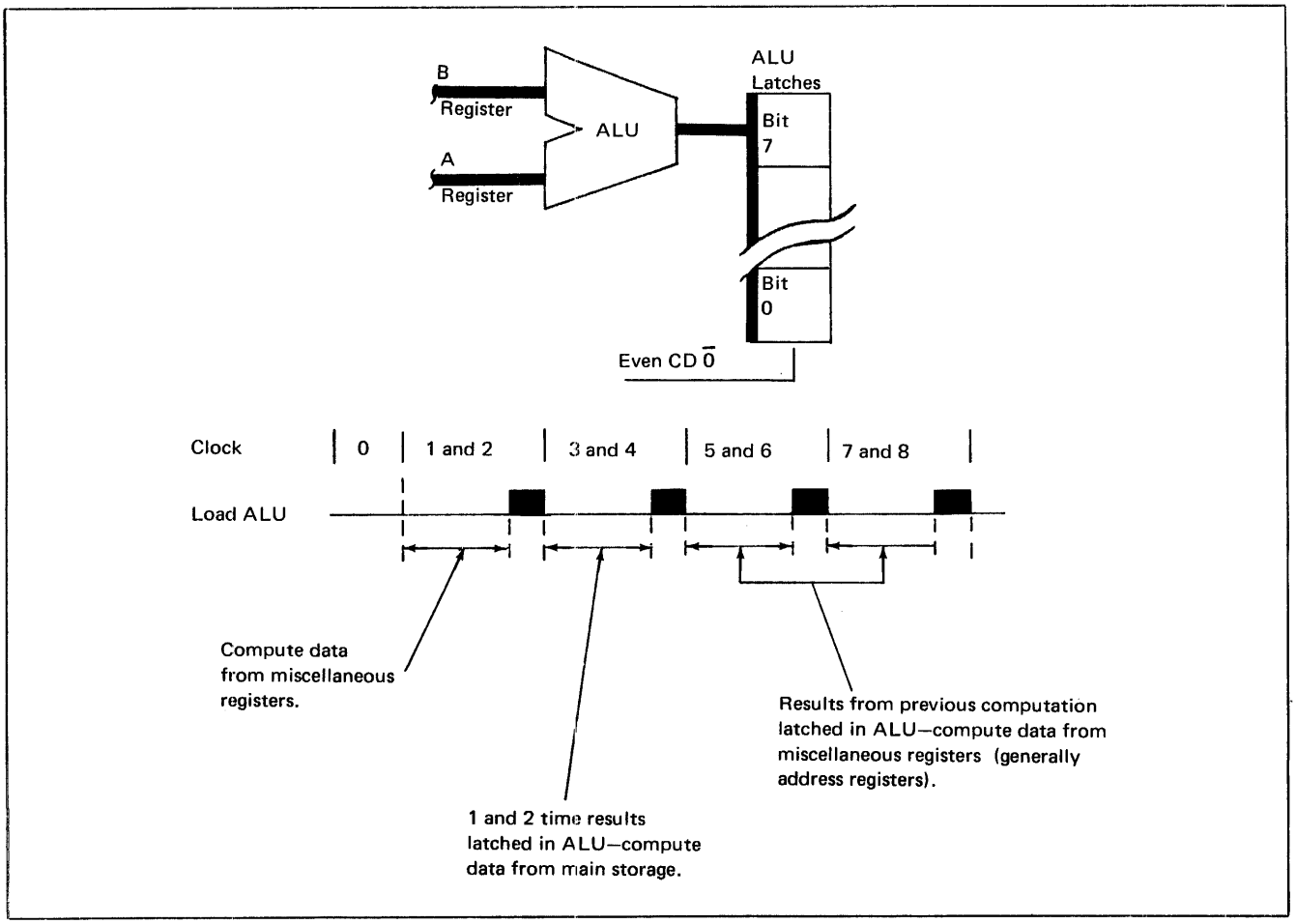


Figure 2-15. ALU Timing

The output from the ALU is sent to main storage, local storage registers, operation register, Q register, condition register, and data bus out. The use of this output is covered under the individual operations.

AND/OR and Test False

Figure 2-16 illustrates the AND/OR and test false functions for a single bit position. Outputs from the test false lines are used to set the CR test false latch. These outputs depend upon the presence or absence of bits in the A and B registers and the active control line (AND or OR). Test false outputs are covered under the operations that use them.

The AND function is a bit-by-bit comparison of the two registers and requires the same bit in each register to have that bit out. The OR function gives an output for any bit which is present in either register. Figure 2-17 contains the outputs available through the use of the AND and OR control lines.

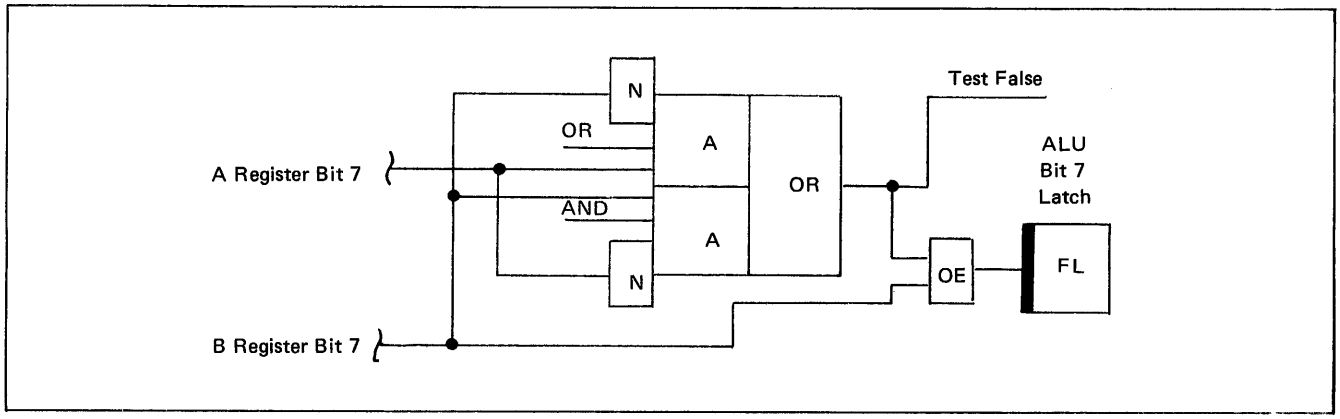


Figure 2-16. Single Bit AND/OR

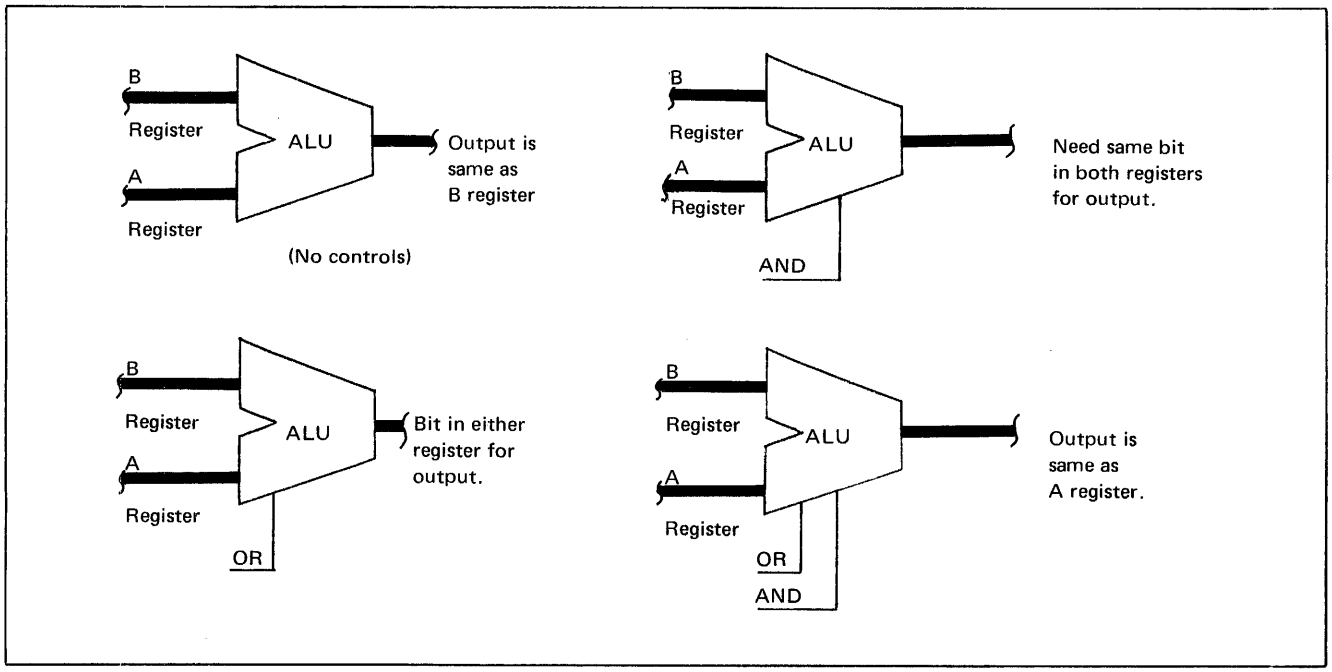


Figure 2-17. ALU-AND/OR Functions

Binary Subtraction

Figure 2-18 shows a decimal comparison between the conventional method of subtraction and the type of subtraction used by the System/3. Under the conventional method, whenever it is necessary to borrow from the next position, the minuend is effectively increased by 10 in the position where the subtraction is taking place and decreased by 1 in the position that is borrowed from. For instance, when subtracting the 6 from the 4 in the units positions, after borrowing from the tens position, the units position of the minuend effectively becomes 14. Because of the borrow, the tens position is reduced to 1. In this example, another borrow is necessary, so after the borrow the tens position of the minuend is effectively 11 and this method continues to the end of the problem.

The same result is reached if, instead of reducing the minuend by 1 after borrowing, the subtrahend is increased by 1 with a carry (Figure 2-18). Thus, in the tens position of the example shown, the carry method effectively subtracts 5 from 12 instead of 4 from 11 as in the conventional method.

Binary subtracting is done in the same way except for the value of the borrows. Because decimal numbers have ascending powers of 10, a borrow has an effective value of 10. Similarly, since binary numbers have ascending powers of 2, a borrow has an effective value of 2. Figure 2-19 illustrates this by subtracting the hexadecimal value B/F from E/B. After subtracting the first two positions, it becomes necessary to borrow in order to subtract the third position. This borrow has an effective value of 2 and the result of the subtraction is 1. Using the carry method of subtraction, a carry to the fourth position gives the subtrahend an effective value of 2. This forces a borrow from the next position which, when added to the minuend, gives it an effective value of 3. This method continues to the end of the problem.

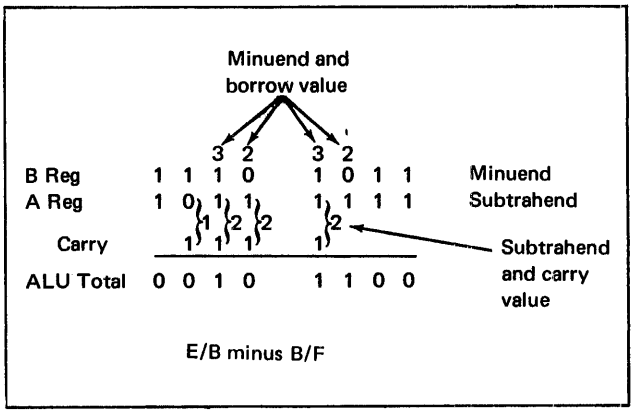


Figure 2-19. Binary Subtract

The minuend enters the ALU from the B register and the subtrahend enters from the A register (Figure 2-20). The character is subtracted bit-by-bit, starting with bit 7 and continuing through bit 0. Carries from bit to bit are internal but if there is a carry from bit 0 it is held in the carry triggers until it is needed in bit 7 (Figure 2-20). Figure 2-21 illustrates the subtract function for a single bit position.

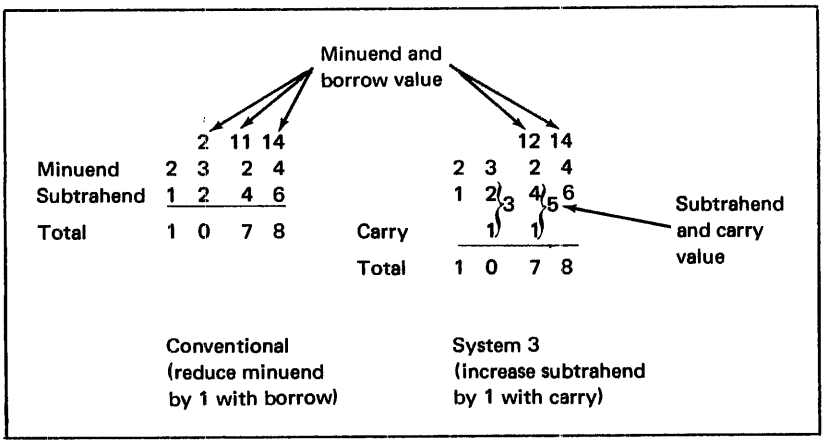


Figure 2-18. Subtraction—Borrow Compared to Carry

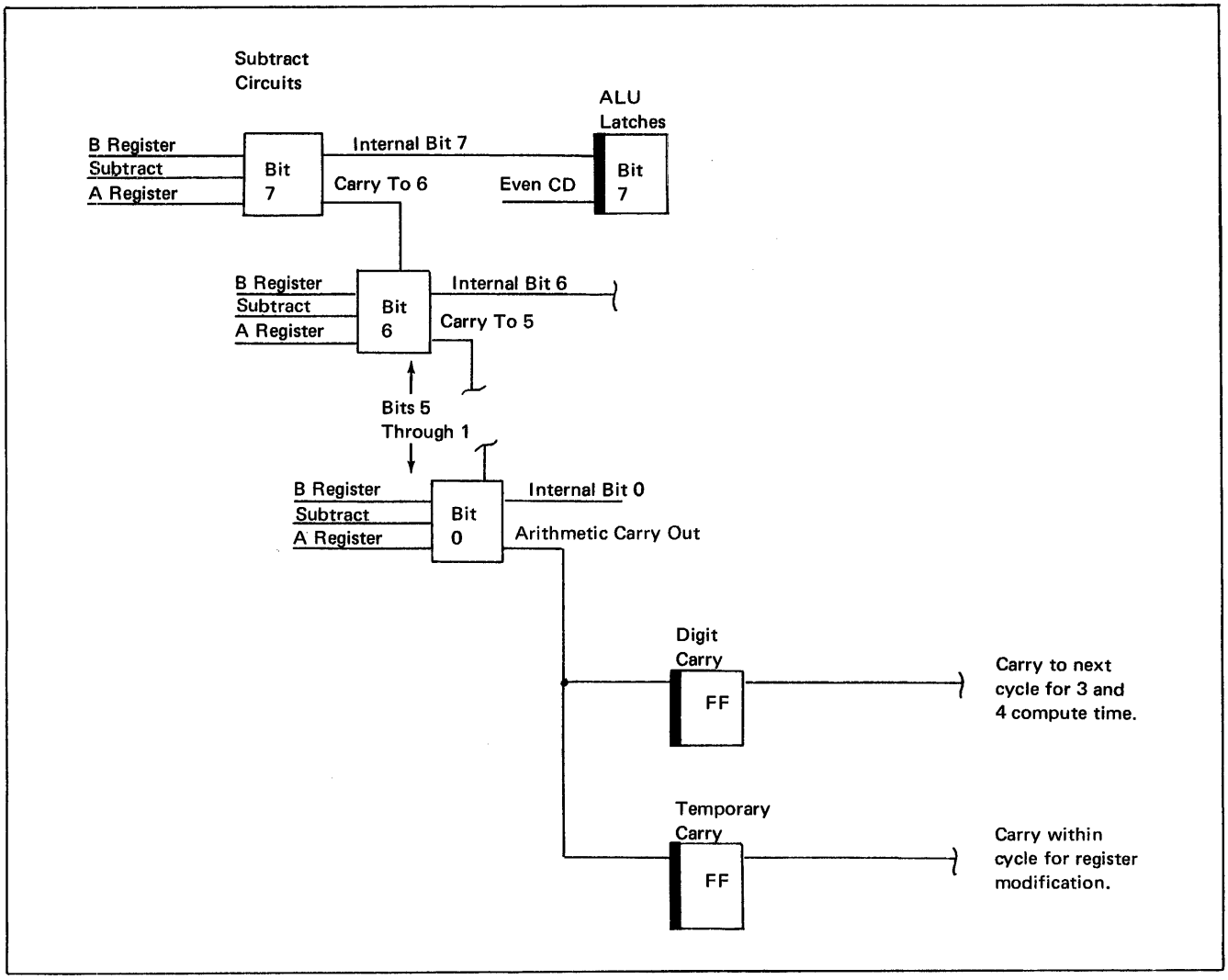


Figure 2-20. ALU Data Flow–Binary Subtract

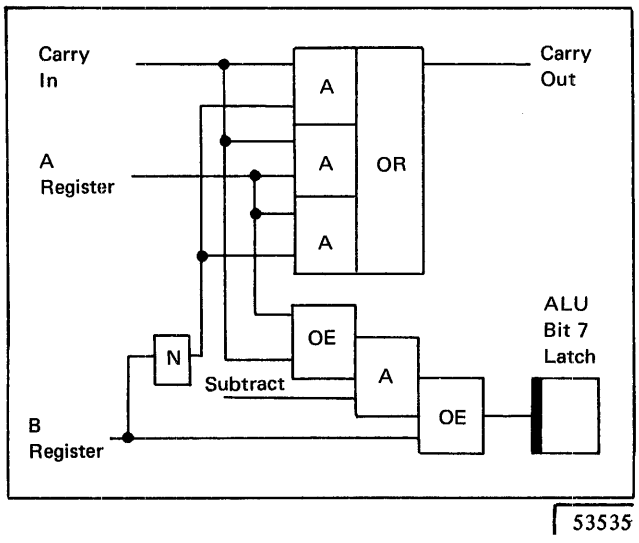


Figure 2-21. Single Bit Subtract

Binary Addition

Since the ALU is designed for binary subtraction, it is necessary to change the figures to be added in a way that will produce the correct results when they are subtracted. This is accomplished by complementing the A register and subtracting it from the B register.

The A register complement figure is an exact binary complement. That is, a bit is replaced by no-bit and no-bit is replaced by a bit. In order to get a true complement figure, it is necessary to force a carry into the low order bit of the first character. Figure 2-22 shows the method used to add the hexadecimal values 8/F 8/3 and 3/F 9/3.

The ALU controls and circuits are the same as binary subtract except for complementing the A register and forcing a carry into bit 7 in the first cycle (Figure 2-23).

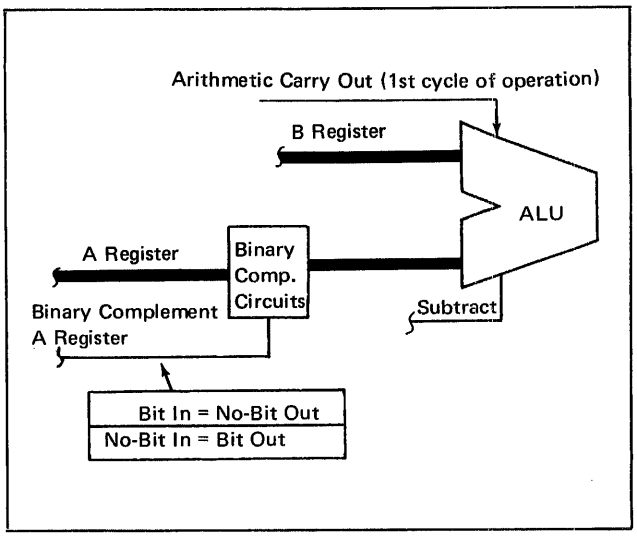


Figure 2-23. Binary Addition Data Flow

Decimal Subtraction

Since the ALU is a binary subtractor, it is capable of handling the binary representation of decimal numbers. However, because decimal numbers only use one-half of each byte, the ALU must be split in half to subtract them. Thus, a carry from bit 4 to bit 3 is used to set the digit carry trigger (Figure 2-24).

Borrowed Amount	32	222 22	
Minuend (True)	1000 1111	1000 0011	
Subtrahend (Complement)	1100 0000	0110 1100	
Carry	1	1111 1 1	Forced Carry
ALU Total	1100 1111	0001 0110	

8/F 8/3 plus 3/F 9/3

Figure 2-22. Binary Add (Complement and Subtract)

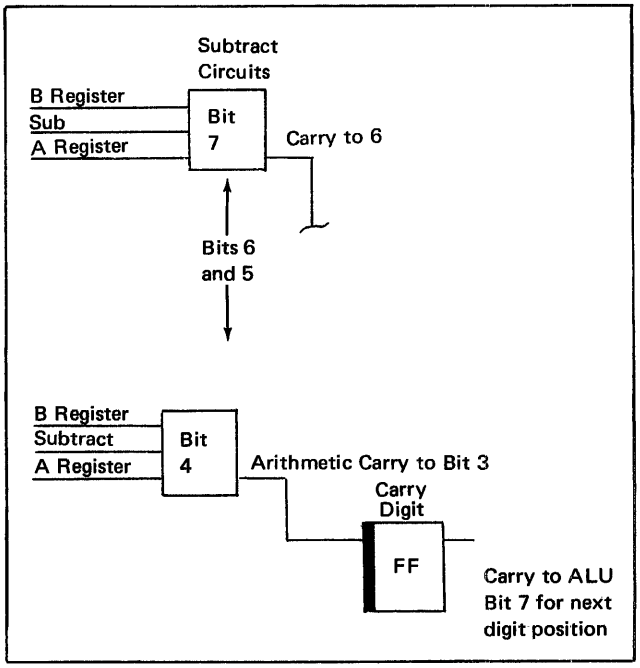


Figure 2-24. Carry Control—Decimal Subtract

Subtraction is done in the same way as binary subtraction as long as the minuend (B register) is larger than the subtrahend (A register). But, because the ALU is capable of handling digits up to 15 (F in hexadecimal) and in this case the digits have a maximum value of 9, the ALU reaches an incorrect decimal result when it becomes necessary to borrow from the next digit (A register larger than B register). This difference of 6 must be subtracted from the result in order to reach a correct result (Figure 2-25). The decimal correct circuits are activated by the carry from the bit 4 position. Figure 2-26 shows the data flow for decimal subtraction and contains a table of the bit correction for the legitimate decimal characters.

Bits 0 to 3 of the low order decimal character contains the sign of the field. The ALU output for bits 0 to 3 is determined by the sign control circuits and is covered under the individual operations.

B Larger Than A 8 - 3		A Larger than B 3 - 8	
	222		2
B register	1000	B Register	0011
A register	0011	A Register	1000
Carry	111	Carry	
Total	0101	Total	1011

Note:
Subtract and complement functions affect only the digit portions. The zone portion is not affected.

	2
Total	1011
Decimal Correct	0110
Carry*	1
Corrected Total	0101

*Mathematical carry; not done by carry circuit

Figure 2-25. Decimal Correction

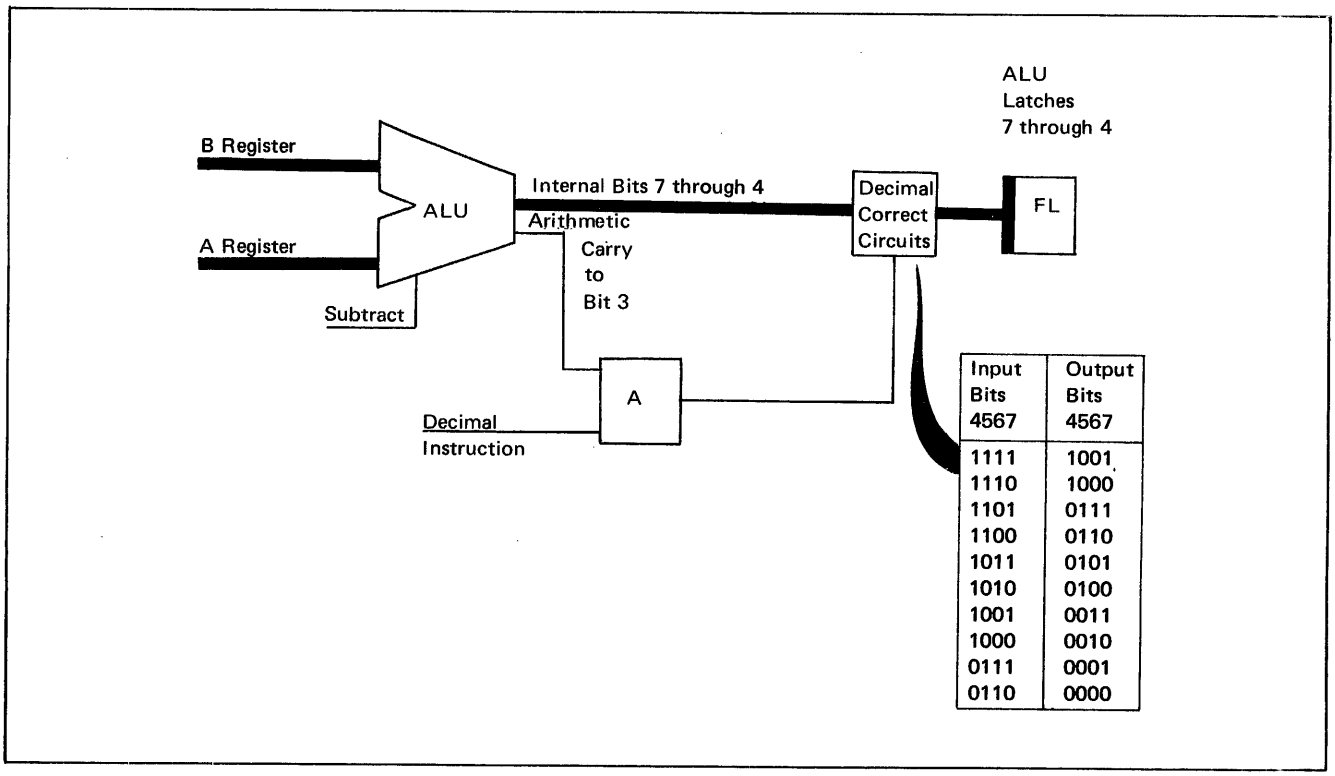


Figure 2-26. Decimal Subtract Data Flow

Decimal Addition

Decimal addition is similar to decimal subtraction in the same way binary addition and subtraction are similar. That is, to add decimal digits it is necessary to complement the A register and subtract it from the B register.

However, since the characters being complemented are decimal, the binary equivalent of the 9's complement is used. Figure 2-27 gives an example of decimal addition. As in subtraction, if the complemented A register digit is larger than the B register, the result must be corrected.

Figure 2-28 shows the decimal add data flow with a table for the 9's complement of the legitimate decimal digits.

	2233
B Register	0011
A Register Complemented	0111
Carry*	1111←Forced
Total	1011
	2
Total	1011
Decimal Correct	0110
Carry*	1
Corrected Total	0101

B Register (3) plus A Register (2)

*Mathematical carry; not done by carry circuits

Figure 2-27. Decimal Addition (Complement and Subtract)

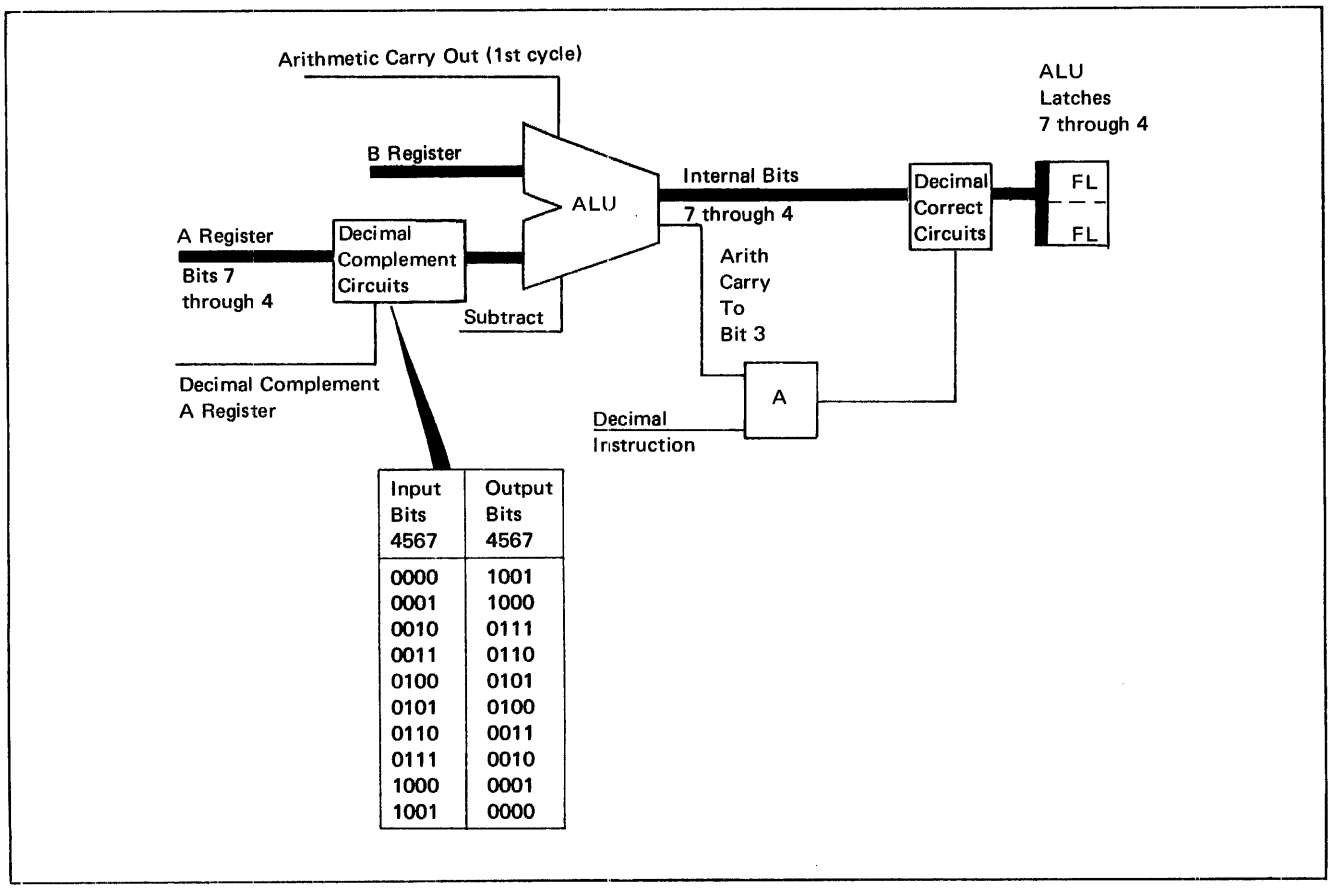


Figure 2-28. Decimal Addition Data Flow

Recomplement

Decimal adding or subtracting under certain conditions produces a result which is a complement form of the correct result. This requires a recomplement operation whereby the result is fed through the ALU a second time to change it to its true form. A recomplement is necessary if the operation is:

- An add operation with unlike signs for the two fields and the A field is larger than the B field.
- A subtract operation with like signs for the two fields and the A field is larger than the B field.
- A result is minus zero.

The need for a recomplement cycle is signaled by a carry from the high order position of the field. In the case of a minus zero result, the recomplement is signaled by the condition register and is covered under the decimal operations. Figure 2-29 contains the data flow for recomplementing.

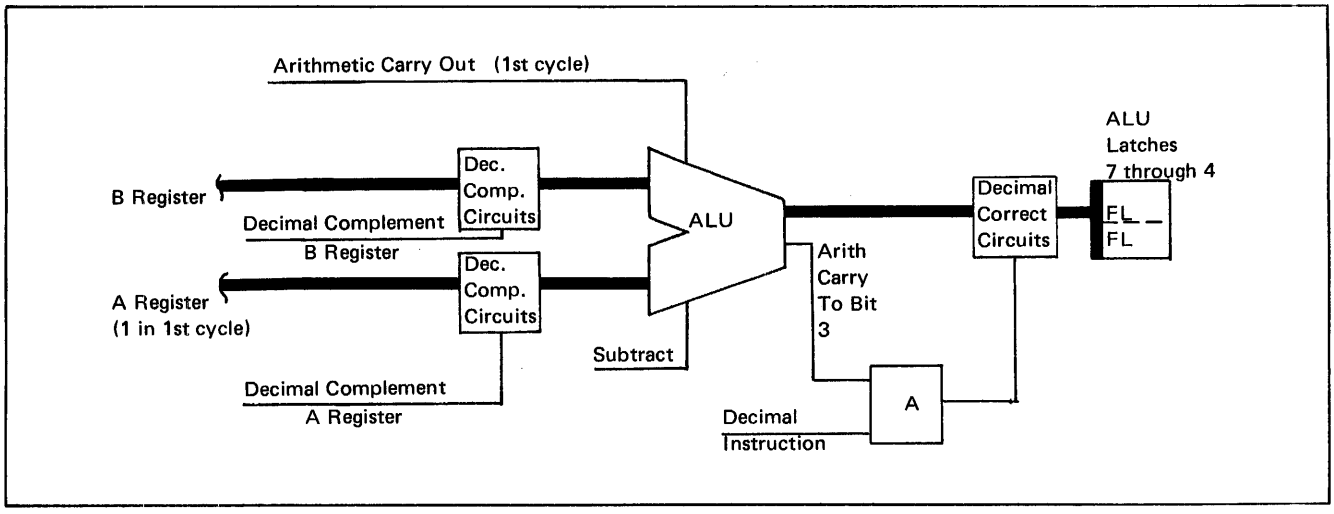


Figure 2-29. Recomplement Data Flow

Figure 2-30 illustrates the method used to recomplement. After subtracting 52 from 27, the ALU result is 75. To recomplement, the result is fed back through the B register and is decimal complemented before entering the ALU. The A register is set to 1 on the first cycle and is left blank each remaining cycle. The decimal complement of the A register is subtracted from the B register complement. A carry is forced into the first digit just as in regular complementing. The final result is the ten's complement of the original result.

CHECK ALU

Parity Generation and Parity Check

Because the parity of the results does not stay constant with the inputs, correct parity is generated for the ALU output. After the data leaves the A and B registers, it can be altered by the decimal and binary complement circuits, the ALU, the decimal correct circuits, and the sign control circuits. The parity changes caused by all these must be considered when determining the parity of the ALU output.

A second ALU, or check ALU, is provided to determine the parity changes which take place within the ALU (Figure 2-31). The check ALU does not have a latched output, decimal correction, or sign control, but otherwise performs similarly to the ALU. The output of the check ALU is a group of exclusive ORs which count the number of changes made to the B register complement after it enters the ALU.

The check ALU output is then added to the changes caused by the B register complement, sign control, and the decimal correct circuits to determine if a P bit is required for the ALU. The output of the ALU is then checked to ensure that it has odd parity.

The A register complement circuits are checked separately. Incorrect parity from either the ALU or the A register complement circuits cause an ALU parity error (Figure 2-31).

Carry Check

An additional set of carry triggers is used to control carries from the high order bit of the check ALU. The triggers function identically to the digit carry trigger and the temporary carry trigger used for the ALU. The outputs of the two carry control groups are then compared to check for the correct number of carries for the ALU (Figure 2-32).

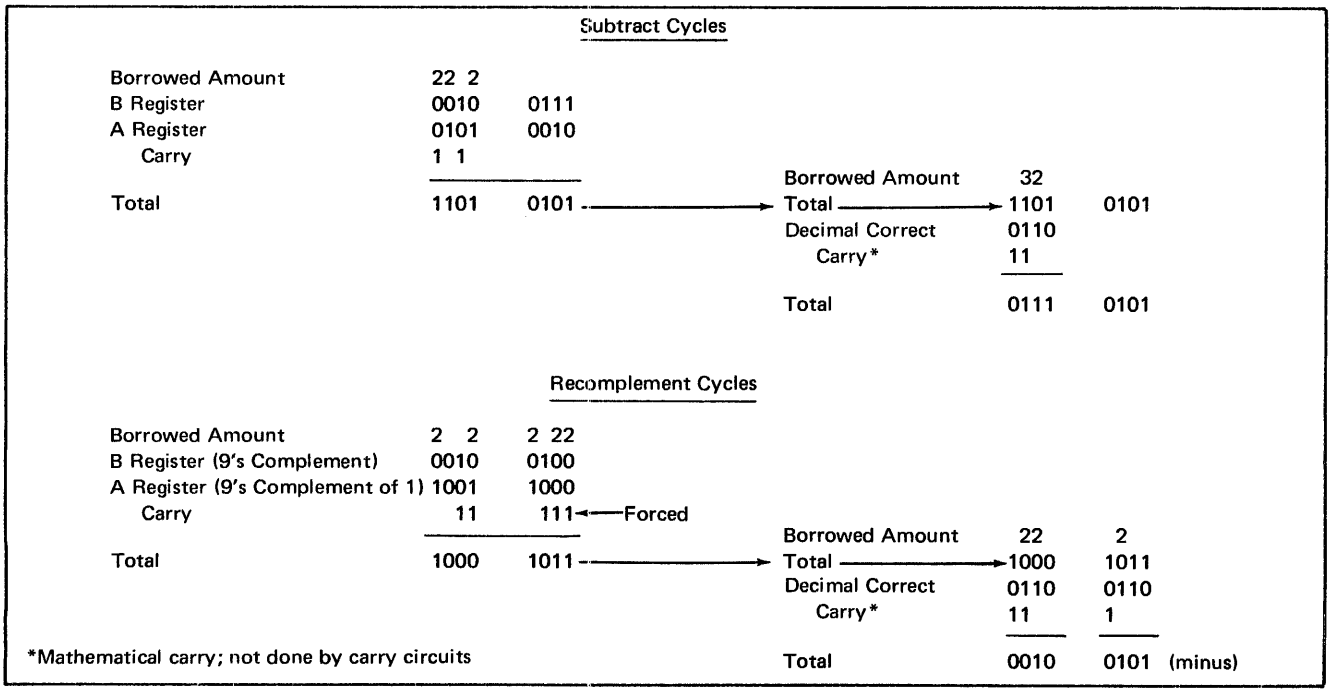


Figure 2-30. Recomplement

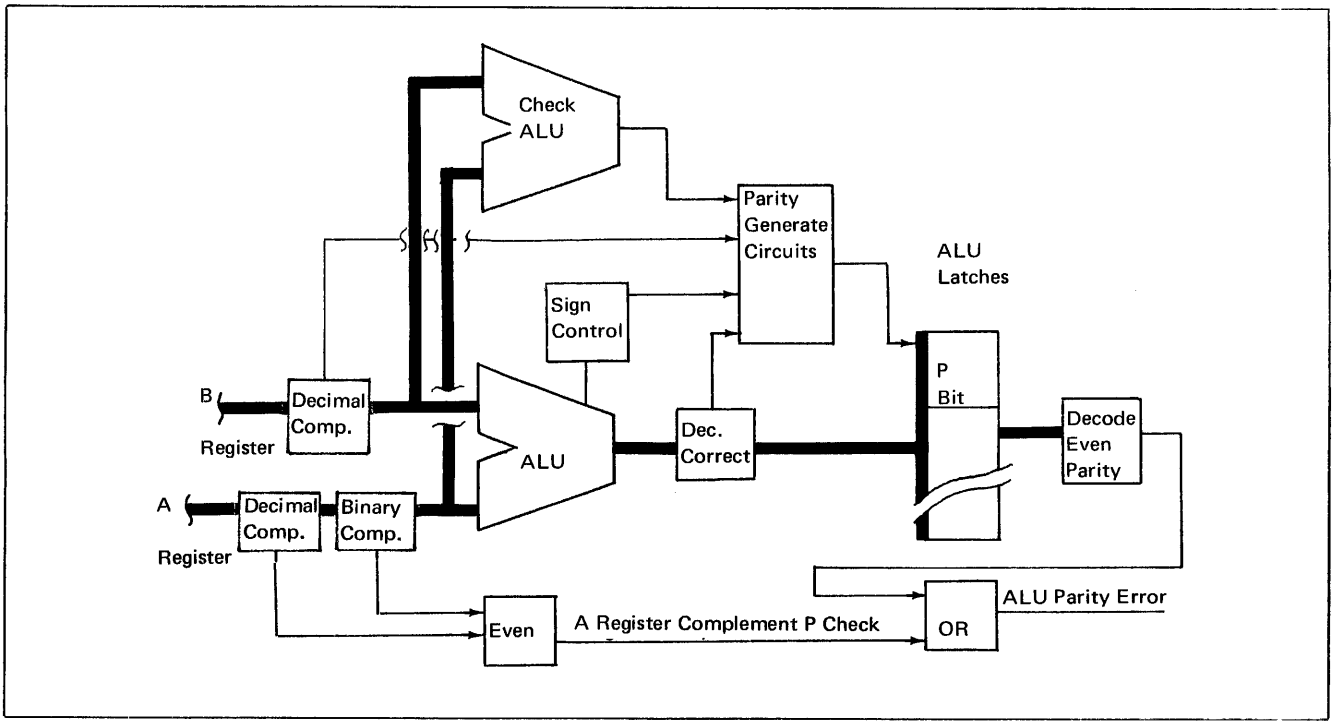


Figure 2-31. ALU Parity Check

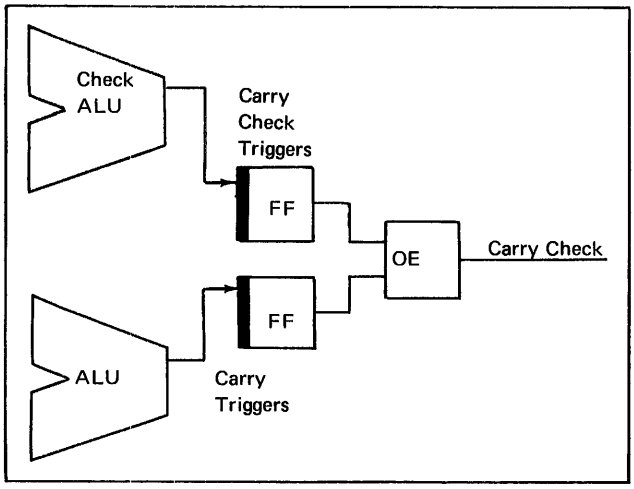


Figure 2-32. ALU Carry Check

LOCAL STORAGE REGISTERS (LSR)

The LSRs are a group of 9 bit registers (one byte plus parity) that serve as halfword address registers. They have the following primary functions:

- Maintaining sequential instruction addresses
- Maintaining current operand addresses during instruction execution
- Maintaining I/O data area addresses

In addition, LSRs have the following secondary functions:

- Index registers for modification of operand addresses
- Interim storage for data, length count, and program condition status referred to as 'scratch pad' type of storage

Figure 2-33 lists the LSRs for the base system and available features. To read out data only 'select' is needed. To write, 'data', 'write hi or write lo' and 'LSR select' are needed. The following is a list of the functions of the base system LSRs:

Instruction Address Register–Used to keep track of the storage of the next sequential instruction byte to be read out of storage. At the beginning of each I-cycle, the address in the IAR is gated into the SAR to be decoded. During each I-cycle, the contents of the IAR is incremented by 1 in preparation for the next I-cycle.

A Address Register–The AAR keeps track of the storage address of the next byte to be addressed in the A field. During I-cycles, the A field address is taken from the instruction and loaded into the AAR. At the beginning of each A-cycle, the address in the AAR is gated into the SAR. During each A-cycle, the contents of the AAR is decremented by 1 in preparation for the next A-cycle.

B Address Register–The BAR keeps track of the storage address of the next byte to be addressed in the B field. During I-cycles, the B field address is taken from the instruction and loaded into the BAR. At the beginning of each B-cycle, the address in the BAR is gated into the SAR. During each B-cycle, the contents of the BAR is normally decremented by 1 in preparation for the next B-cycle.

Index Register 1 and Index Register 2–These registers can each store a two byte address to be used in indexing operations. During an indexing operation, the CPU automatically adds the single byte displacement from the instruction to the contents of XR1 or XR2 to obtain the actual B or A

LOCAL STORE REGISTERS (BASE SYSTEM)

LOCAL STORE REGISTERS (BASE SYSTEM)

01	Prog Level 1 Instr Address Reg	P1 - IAR
02	Prog Level 1 Address Recall Reg	P1 - ARR
03	A Address Reg	AAR
04	Spare	SPARE 1
05	Prog Level 1 Index Reg 1	P1 - XR1
06	Prog Level 1 Status Reg*	P1 - PSR
07	B Address Reg	BAR
08	MFCU Print Data Address Reg	MPTAR
09	Prog Level 1 Index Reg 2	P1 - XR2
10	Line Printer Data Address Reg	LPDAR
11	Line Printer Image Address Reg	LPIAR
12	MFCU Punch Data Address Reg	MPCAR
13	MFCU Read Address Reg	MRDAR
14	Length Count Reg LCR Data Recall Reg	DRR
15	Interrupt Level 1 Instr Address Reg	IAR - 1
16	Interrupt Level 1 Address Recall Reg	ARR - 1

* PSR Lo is used as the Condition Recall Register, CRR;
PSR Hi is used as the Length Count Recall Register, LCRR.

LOCAL STORE REGISTERS (FEATURE 1)

01	Prog Level 2 Instr Address Reg	P2 - IAR
02	Prog Level 2 Address Recall Reg	P2 - ARR
03	Bi-Sync Comm Adapter Address Reg	ASCAR
04	Serial I/O Channel Address Reg	SIAR
05	Prog Level 2 Status Reg*	P2 - PSR
06	Interrupt Level 4 Instr Address Reg	IAR - 4
07	Interrupt Level 4 Address Recall Reg	ARR - 4
08	Disk File Control Address Reg	DFCR
09	Prog Level 2 Index Reg 2	P2 - XR2
10	Spare	SPARE 4
11	Interrupt Level 2 Instr Address Reg	IAR - 2
12	Interrupt Level 2 Address Recall Reg	ARR - 2
13	Disk File Data Address Reg	DFDR
14	Prog Level 2 Index Reg 1	P2 - XR1
15	Interrupt Level 0 Instr Address Reg	IAR - 0
16	Interrupt Level 0 Address Recall Reg	ARR - 0

LOCAL STORE REGISTERS (FEATURE 2)

01-14	Spare	Spare
15	Interrupt Level 3 Instr Address Reg	IAR-3
16	Interrupt Level 3 Address Recall Reg	ARR-3

Figure 2-33. Local Storage Registers

field address. The contents of the index registers is not changed as a result of the addition. The resulting address is placed in the BAR or the AAR.

Address Recall Register–On a branch instruction, the ARR contains the 'branch to' address. On a decimal instruction, the ARR retains the starting address of the B field in the event recomplementing is required. On an insert and test characters instruction, the ARR contains the address of the first significant digit encountered.

Length Count Register–The LCR is a one byte register that contains the length count of the B and A fields. It gets decremented by 1 on each B-cycle except the first one.

Data Recall Register–The DRR is a one byte register that provides temporary storage for the data character read out of storage during each A-cycle. It is also used to store the Q code of single address instructions.

Program Status Register–The high byte of the PSR is used as the Length Count Recall Register (LCRR). The LCRR is used only during a recomplement operation. It stores the length of the data fields and is decremented on each recomplement cycle except the first. The low byte of the PSR is used as the Condition Recall Register (CRR). The CRR is used to store the contents of the condition register when changing program levels (used only with dual programming).

MFCU Read Data Address Register–The MRDAR keeps track of which storage position is to be addressed next while reading data from a card into the card read area in core storage.

MFCU Punch Data Address Register–The MPCAR keeps track of which storage position on the MFCU print data area is to be addressed next during a punch operation.

MFCU Print Data Address Register–The MPTAR keeps track of which storage position in the MFCU print data area is to be addressed next during an MFCU print operation.

Line Printer Data Address Register–The LPDAR keeps track of which storage position in the line printer data area is to be addressed next during a print operation.

Line Printer Image Address Register–The LPIAR keeps track of which storage position in the chain image area is to be addressed next during a print operation of the line printer.

Interrupt Level 1–The IAR-1 contains the address of the next sequential instruction byte to be read out of storage during an interrupt level 1 operation.

Interrupt Level 1 Address Recall Register–The ARR-1 has the same function as the ARR, but is active only during an interrupt level 1 operation.

The registers are paired (Figure 2-34) to give an LSR Hi (the high order byte) and an LSR Lo (the low order byte). Only one byte can be written into an LSR at a time. To write into an LSR, it is necessary to activate the select line for a pair of registers and the 'LSR write Hi' or 'LSR write Lo'. All 18 bits for the LSR selected are available at the output of the LSR array. These bits can be gated to the SAR (18 bits), the B register (9 bits), and the A register (9 bits) for modification of the addresses in these registers.

FEMD 4-070 shows the circuitry for the LSRs.

LSR select and write lines are normally controlled by the CPU. However, during an I/O cycle, the I/O attachment can control the select lines for the LSR assigned to it.

FEMD 4-072 shows the LSR select circuitry and FEMD 4-076 shows the LSR select I/O circuitry.

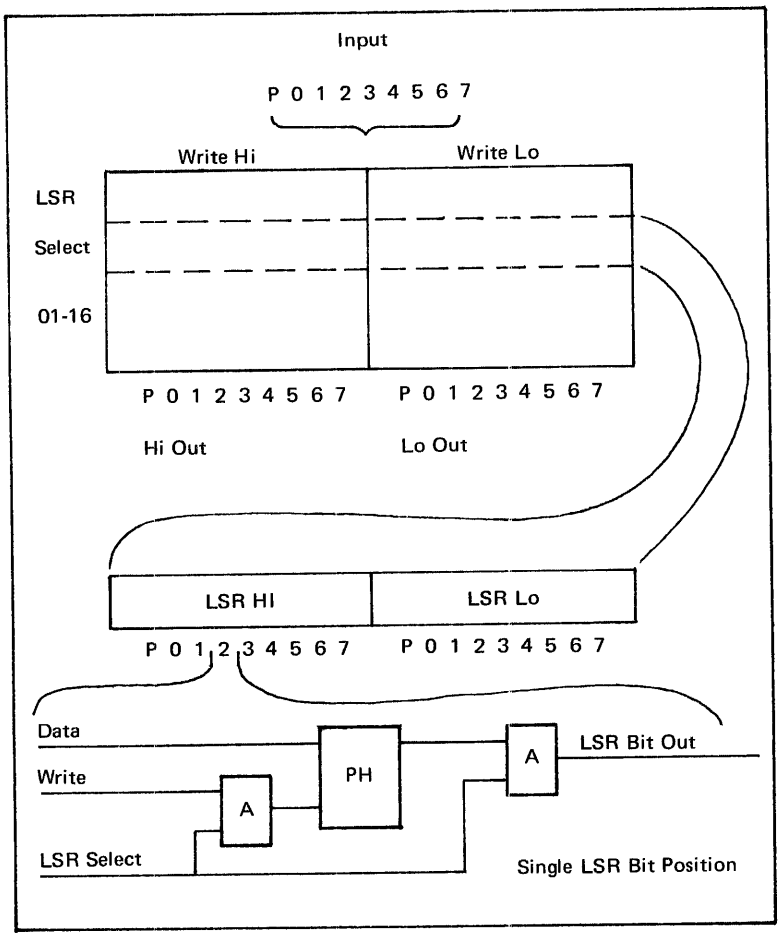


Figure 2-34. Local Storage Register Array

OP REGISTER

Op codes of the CPU are eight bits wide and are stored in the op register. The op codes are sent to the op register from main storage through the SDR, the B register, and the ALU. Decode of the op register is performed to determine which instruction the machine should execute. When an I/O instruction is decoded, the I/O attachments receive this information through signal lines which are a direct output of the decode logic.

Q REGISTER

Each instruction includes a Q byte which generally serves to extend or modify the op code. During the execution of an instruction the Q byte is stored in the Q register. This register is loaded through the same data path as the op register and its output is decoded to determine the necessary information for the CPU to execute the instructions. During I/O instructions the Q register stores the Q byte, but each I/O device also receives the Q byte from the ALU output on DBO at the same time. The CPU does not use the Q byte of the I/O instruction even though it is stored in the Q register.

CONDITION REGISTER (CR)

The condition register is a six bit register which contains the six conditions on which the system may test as a result of instruction execution. These six bits are tested as follows:

2	3	4	5	6	7	
x	x	x	x	x	1	Equal condition
x	x	x	x	1	x	Low condition
x	x	x	1	x	x	High condition
x	x	1	x	x	x	Decimal overflow condition
x	1	x	x	x	x	Test false condition
1	x	x	x	x	x	Binary overflow condition

The equal, low, high, and binary overflow conditions reflect the result of executing the last instruction which affected them (one of the following):

- Add zoned decimal
- Zero and add zoned
- Subtract zoned decimal
- Edit
- Compare logical characters
- Add logical characters
- Subtract logical characters
- Add to register
- Compare logical immediate

The decimal overflow or the test false condition is set by the first instruction which results in that condition and can be reset by one of the following:

- Branch on condition
- Jump on condition
- Load register (PSR) instruction (loads the respective bit position to zero)
- System reset

System reset initializes the condition register to:

- Equal condition
- Not overflow condition
- Not test false condition

Loading of the condition register may be from the ALU output, but normally the bits are set individually in the latches by the CPU logic as a result of instruction execution. When the CR contents is needed for program testing, its output is fed to the A register and into the ALU.

The lower six bits of the program status register (PSR) contain the image of the condition register for the specific program level. PSR is used to save and to initialize condition register settings of the individual program levels.

Figure 2-35 shows the condition register settings.

The conditions of the I/O attachment logic are stored in registers in the attachment and do not affect the state of the condition register.

Operation / Condition	Bit	Decimal Arithmetic	Logical Compare Sub Logical	Add Logical Add to Register	Edit	Test Bits	Branch or Jump on Condition
Equal	7	Result is Zero	First Operand is Equal to Second	Result is Zero	Source is Zero		
Low	6	Negative	First Operand is Lower than Second	No Carry and Non-Zero Result	Negative		
High	5	Positive	First Operand is Higher than Second	Carry and Non-Zero Result	Positive		
Decimal Overflow	4	Overflow					Overflow Reset if Tested
Test False	3					Test False	Test False Reset if Tested
Binary Overflow	2			Overflow			

Figure 2-35. Condition Register Settings

I/O INTERFACE

The CPU acts as a controller for all I/O devices operating over a single I/O attachment interface. The I/O devices operate in a cycle steal mode over an interface called input/output channel (I/O Channel).

The I/O Channel consists of:

1. A set of powered signal lines which carry information to and from the CPU, and
2. Logic to establish cycle steal priorities.

The following priority is assigned to the devices using cycle steal:

1. Disk File Read/Write
2. Spare
3. Printer
4. SIOC
5. Spare
6. BSCA
7. MFCU Read and Punch
8. Spare
9. Spare
10. Spare
11. 1442
12. MFCU Print
13. Spare
14. Spare
15. Spare
16. Spare
17. Spare
18. Spare
19. Spare
20. Disk File Control

This priority assignment makes it possible for the interface to operate on a time sharing basis without the need for I/O buffers. Once a cycle steal request has been granted, the attachment has complete control of the CPU for that cycle.

FEMD 4-100 shows the I/O interface lines.

DBI Translator

The DBI translator is used during clock 2 and 3 time to translate the 96 column card code into EBCDIC. The translator is not used in every I/O cycle and if the 'translate in' line is inactive, the I/O data is transferred to the A register unchanged. Figure 2-36 shows a 'translate in' conversion table.

FEMD 4-060 shows the circuitry for the DBI translator.

Used only for IPL

B	A	8	4	2	1	6 Bit Card Code Positions
0	1	2	3	4	5	6 7 DBI Bits

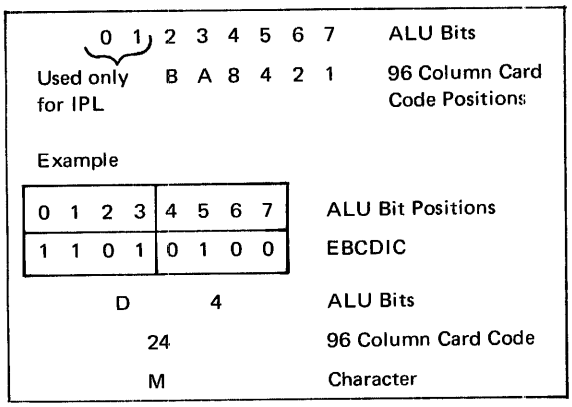
Example

0	1	2	3	4	5	6	7	DBI Bits
0	0	1	0	0	1	0	0	Card Code
2				4				Code in EBCDIC
D				4				

DBI BITS 0,1,2,3 → / 4,5,6,7 ↓	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	40 SPACE	F0 ϕ	60 –	D0	00	B0	20	90	C0	70	E0	6A	80	30	A0	2A
1	F1 1	61 /	D1 J	C1 A	B1	21	91	81	71	E1	51	41	31	A1	11	01
2	F2 2	E2 S	D2 K	C2 B	B2	A2	92	82	72	62	52	42	32	22	12	02
3	F3 3	E3 T	D3 L	C3 C	B3	A3	93	83	73	63	53	43	33	23	13	03
4	F4 4	E4 U	D4 M	C4 D	B4	A4	94	84	74	64	54	44	34	24	14	04
5	F5 5	E5 V	D5 N	C5 E	B5	A5	95	85	75	65	55	45	35	25	15	05
6	F6 6	E6 W	D6 O	C6 F	B6	A6	96	86	76	66	56	46	36	26	16	06
7	F7 7	E7 X	D7 P	C7 G	B7	A7	97	87	77	67	57	47	37	27	17	07
8	F8 8	E8 Y	D8 Q	C8 H	B8	A8	98	88	78	68	58	48	38	28	18	08
9	F9 9	E9 Z	D9 R	C9 I	B9	A9	99	89	79	69	59	49	39	29	19	09
A	7A :	50 &	5A !	4A c	3A	10	1A	0A	FA	EA	DA	CA	BA	AA	9A	8A
B	7B #	6B ,	5B $	4B .	3B	2B	1B	0B	FB	EB	DB	CB	BB	AB	9B	8B
C	7C @	6C %	5C *	4C <	3C	2C	1C	0C	FC	EC	DC	CC	BC	AC	9C	8C
D	7D '	6D _	5D)	4D (	3D	2D	1D	0D	FD	ED	DD	CD	BD	AD	9D	8D
E	7E =	6E >	5E ;	4E +	3E	2E	1E	0E	FE	EE	DE	CE	BE	AE	9E	8E
F	7F "	6F ?	5F ¬	4F I	3F	2F	1F	0F	FF	EF	DF	CF	BF	AF	9F	8F

Figure 2-36. DBI Translator

53540



DBO Translator

The DBO translator is used during clock 4 and 5 time to translate ALU data (EBCDIC) to 96 column card code. The DBO translator is not used during every I/O cycle and if the 'Translate Out' line is inactive, the data is transferred to the I/O attachment unchanged. Figure 2-35 shows a 'translate out' conversion table.

FEMD 4-065 shows the circuitry for the DBO translator.

ALU BITS 0,1,2,3 → / 4,5,6,7 ↓	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	40	5A	60	D0	00 SPACE	1A &	20 –	90	C0	70	E0	50	80	30 }	A0	10 0
1	F1	E1	51	C1	B1	A1	11 /	81	71	61	D1	41	31 A	21 J	91	01 1
2	F2	E2	D2	C2	B2	A2	92	82	72	62	52	42	32 B	22 K	12 S	02 2
3	F3	E3	D3	C3	B3	A3	93	83	73	63	53	43	33 C	23 L	13 T	03 3
4	F4	E4	D4	C4	B4	A4	94	84	74	64	54	44	34 D	24 M	14 U	04 4
5	F5	E5	D5	C5	B5	A5	95	85	75	65	55	45	35 E	25 N	15 V	05 5
6	F6	E6	D6	C6	B6	A6	96	86	76	66	56	46	36 F	26 O	16 W	06 6
7	F7	E7	D7	C7	B7	A7	97	87	77	67	57	47	37 G	27 P	17 X	07 7
8	F8	E8	D8	C8	B8	A8	98	88	78	68	58	48	38 H	28 Q	18 Y	08 8
9	F9	E9	D9	C9	B9	A9	99	89	79	69	59	49	39 I	29 R	19 Z	09 9
A	7A	6A	FO	4A	3A c	2A !	BO	0A :	FA	EA	DA	CA	BA	AA	9A	8A
B	7B	6B	5B	4B	3B .	2B $	1B ,	0B #	FB	EB	DB	CB	BB	AB	9B	8B
C	7C	6C	5C	4C	3C <	2C *	1C %	0C @	FC	EC	DC	CC	BC	AC	9C	8C
D	7D	6D	5D	4D	3D (	2D)	1D –	0D '	FD	ED	DD	CD	BD	AD	9D	8D
E	7E	6E	5E	4E	3E +	2E ;	1E >	0E =	FE	EE	DE	CE	BE	AE	9E	8E
F	7F	6F	5F	4F	3F ı	2F	1F ?	0F "	FF	EF	DF	CF	BF	AF	9F	8F

Figure 2-37. DBO Translator

53542

TWO ADDRESS INSTRUCTION

I-Cycles

- Load operation code into op register.
- Load Q code into Q register, LCR, and LCRR.
- Load B field address into BAR.
- Load B field address into ARR for decimal instructions.
- Load A field address into AAR.

When performing two address instructions, the following information must be known:

1. What is the operation?
2. Where are the fields located?
3. How long are the fields or is there any special consideration that must be given them?

I-cycles are used to load the various controlling registers with this information.

First, an I-op cycle transfers the operation code from main storage to the op register. For two address instructions, the Q code generally contains the length count for the fields. The only exception to this is the move hex character operation in which the Q code controls the data flow. Differences in Q code use are covered under the individual operations. However, in all cases an I-Q cycle loads the Q code into the Q register, the LCR, and the LCRR.

Two cycles, I-H1 and I-L1, are used to load the B field address into the BAR. For decimal instructions, the B field address is also loaded into the ARR. If indexing is used, a single I-X1 cycle replaces the I-H1 and I-L1 cycles.

An I-H2 and I-L2 cycle load the A field address into the AAR. The A field address can also be indexed by replacing the I-H2 and I-L2 cycles with a single I-X2 cycle.

I-Op Cycle

The first step in an I-op cycle, as in all cycles, is to address the core storage location to be used during that cycle. At clock 0 time (Figure 3-1) the contents of the IAR are transferred to the SAR. The operation register byte is then read from core storage and enters the SDR.

During clock 3 and 4 times the byte is transferred through the B register and is latched into the ALU output (Figure 3-2). At clock 5 time the latched ALU output is then stored in the op register. Read call/write call stores the SDR contents back into the same core storage location to regenerate the operation character.

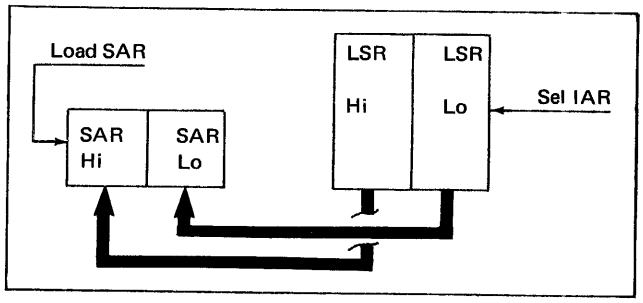


Figure 3-1. Instruction Cycle–Storage Addressing

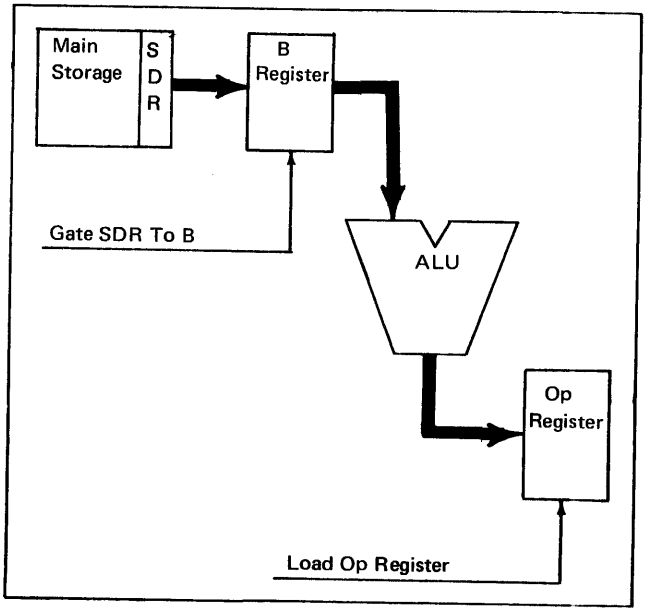


Figure 3-2. Storage to Op Register Data Flow

The rest of the cycle is then used to increment the IAR so that the next instruction position can be addressed. Figure 3-3 shows that a 1 is added to the IAR contents. Two steps are required because of the possibility of a carry from the low order to the high order position.

FEMD 5-010 contains the circuit description.

I-Q Cycle

The I-Q cycle is the same as an I-op cycle, except that the Q code byte is stored in the LCR, the LCRR, and the Q register (Figure 3-4). The IAR is incremented in the same manner as for an I-op cycle (Figure 3-3).

FEMD 5-010 contains the circuit description.

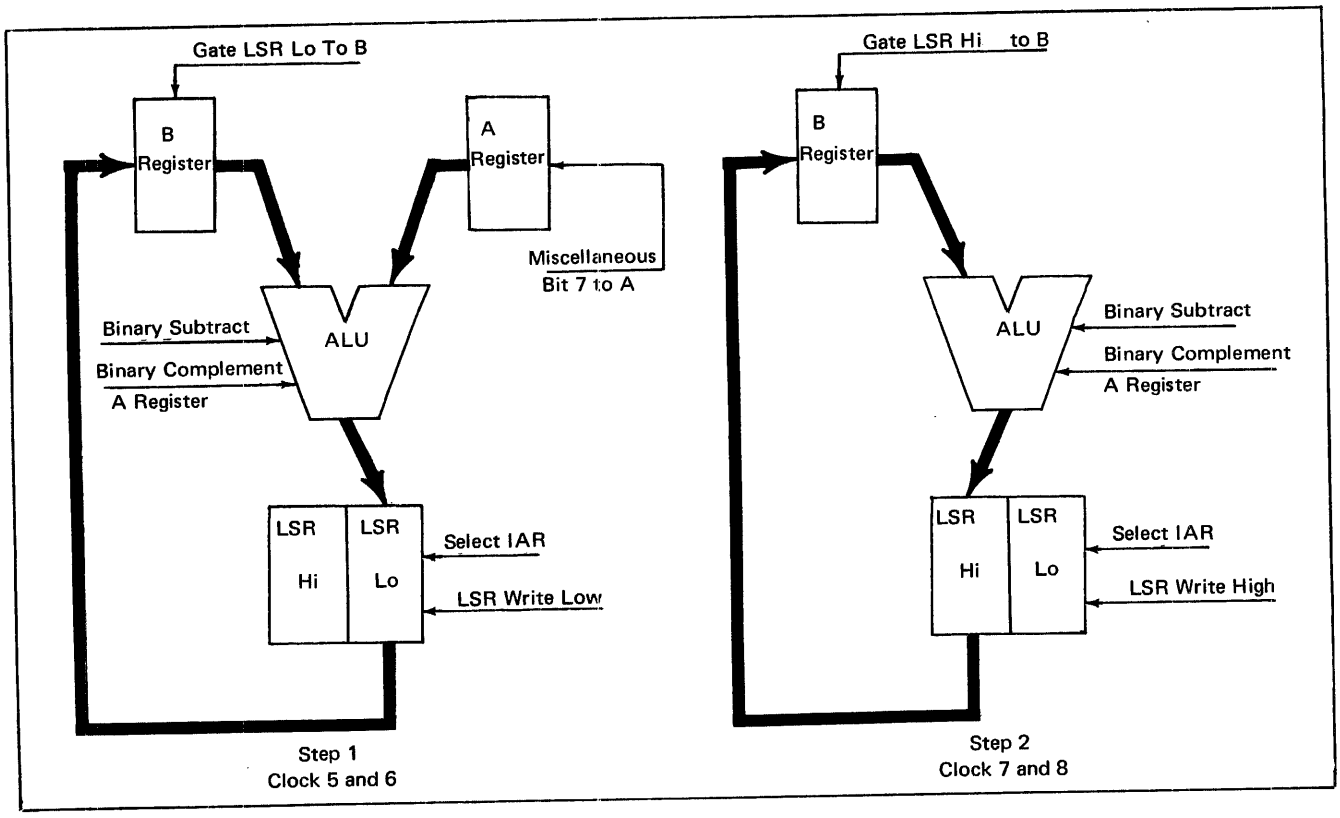


Figure 3-3. Incrementing IAR

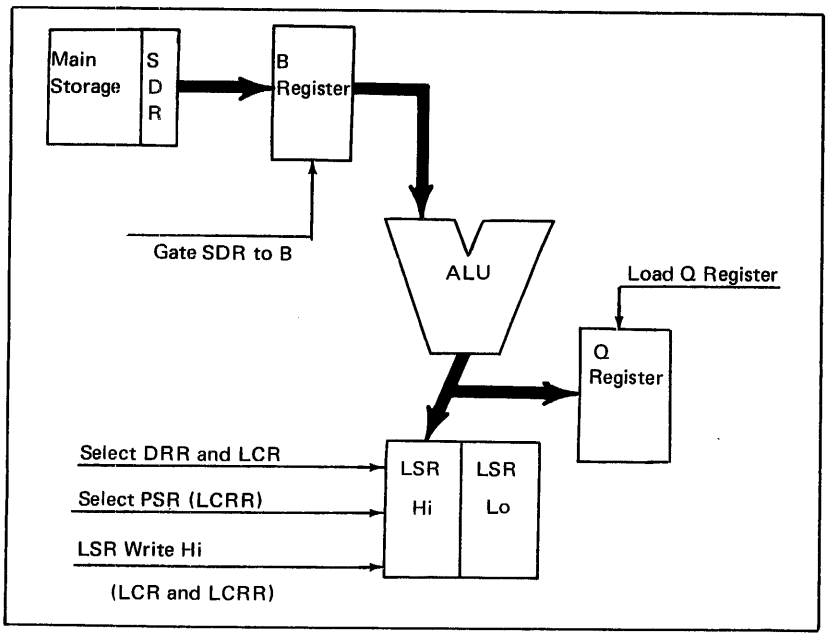


Figure 3-4. I-Q Cycle-Storage to Registers Data Flow

I-H1 and I-L1 Cycles

I-H1 and I-L1 cycles are the same as the I-op and I-Q cycles except that the data bytes, B field address, are stored in the BAR. During the I-H1 cycle, the first byte is stored in the high order position of the BAR (Figure 3-5). The following cycle, I-L1, stores the second byte in the low order position of the BAR. For decimal instructions, the bytes are also stored in the ARR.

FEMD 5-030 contains the circuit description.

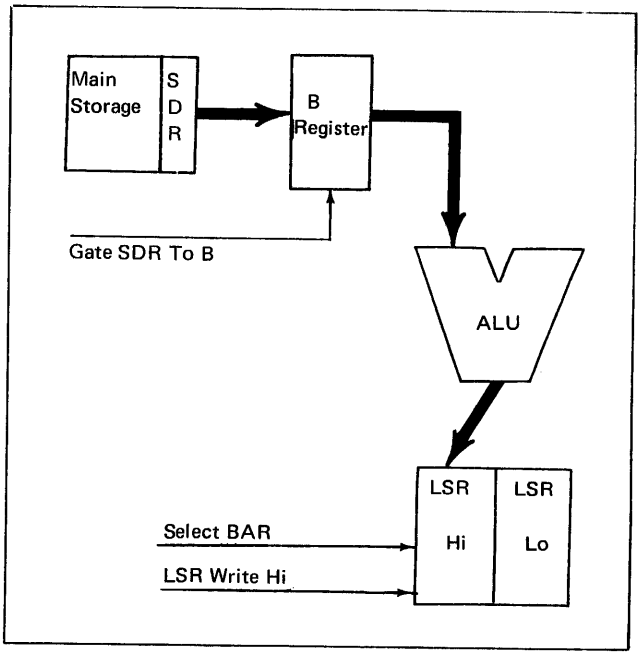


Figure 3-5. I-H1 Cycle–Storage to BAR High

I-H2 and I-L2 Cycles

I-H2 and I-L2 cycles are the same as I-H1 and I-L1 cycles except the address bytes are stored in the AAR.

FEMD 5-030 contains the circuit description.

Indexing

The need for I-X cycles is determined by the bit structure of bits 0 through 3 of the operation register. An I-X1 cycle results from the presence of either bit 0 or bit 1, but not both; an I-X2 cycle from either bit 2 or 3, but not both. The bit which is present also determines the index register used (Figure 3-6).

Cycle	Operation Register Bit	Index Register Selected
I-X1	1	XR1
	0	XR2
I-X2	3	XR1
	2	XR2

Figure 3-6. Index Register Selection

During an I-X1 cycle the IAR is selected and loaded into the SAR in the same manner as for other instruction cycles. At clock 3 time the low order position of the selected index register is entered into the A register (Figure 3-7). The address byte is read from main storage and is gated from the SDR to the B register. The two bytes are then added in the ALU. At clock 4 time, the index register is dropped and the BAR is selected. The ALU contents are then written into the low order position of the BAR. If a carry results from the computation, it is added to the high order position of the BAR during clock 7 and 8 time (Figure 3-8). The high order position of the selected index register is gated into the B register and added in the ALU. At clock 8 time the results are written into the high order position of the BAR. If the operation is a decimal or branch operation, the results are also stored in the ARR. An I-X2 cycle operation is the same as an I-X1 except that the results are written in the AAR.

Since clock 7 and 8 times are normally used to increment the high order position of the IAR, the incrementing sequence is changed for an I-X cycle. Because the high order position of the IAR can be affected only by a carry from the low order position, the CPU looks ahead to determine if a carry will be needed. During clock 1 and 2 times the

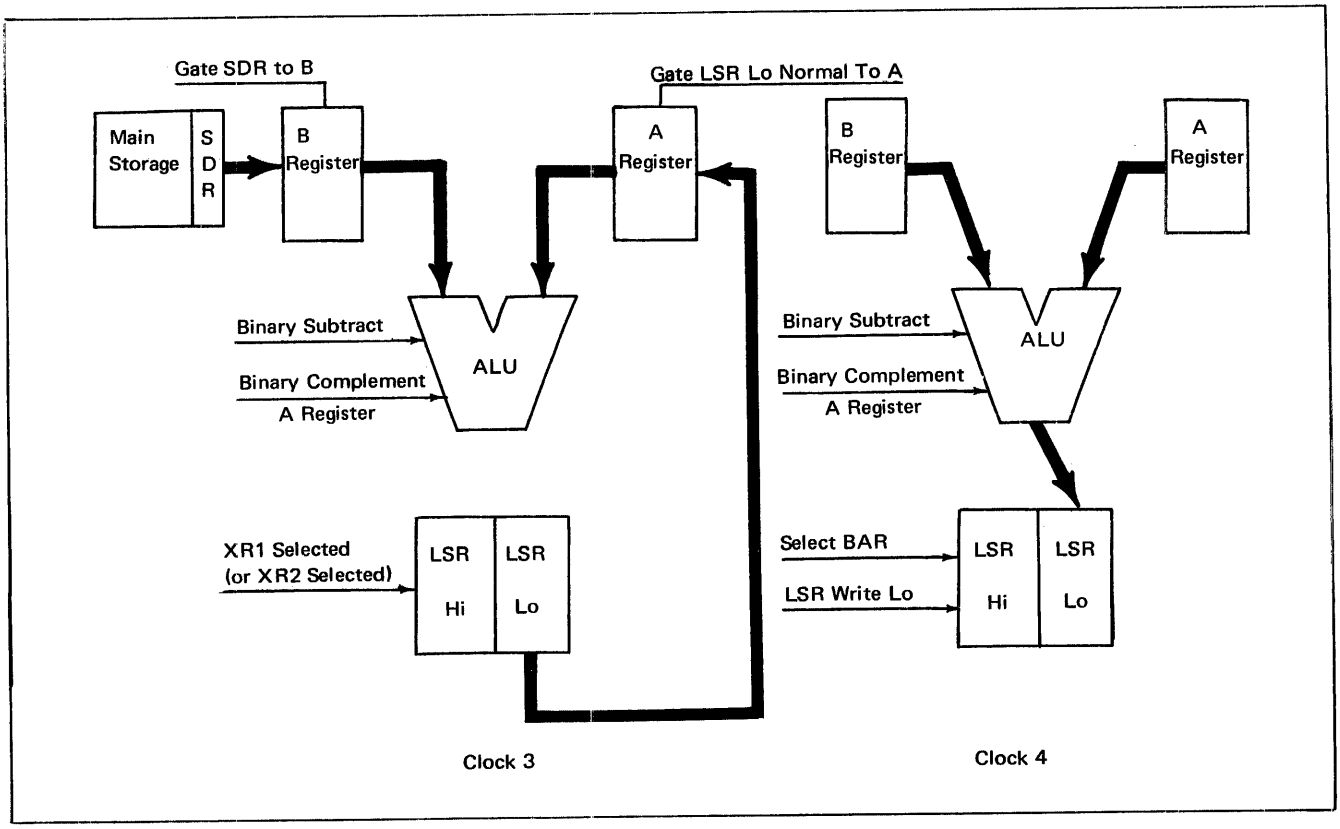


Figure 3-7. I-X1 Cycle–Indexing BAR Low

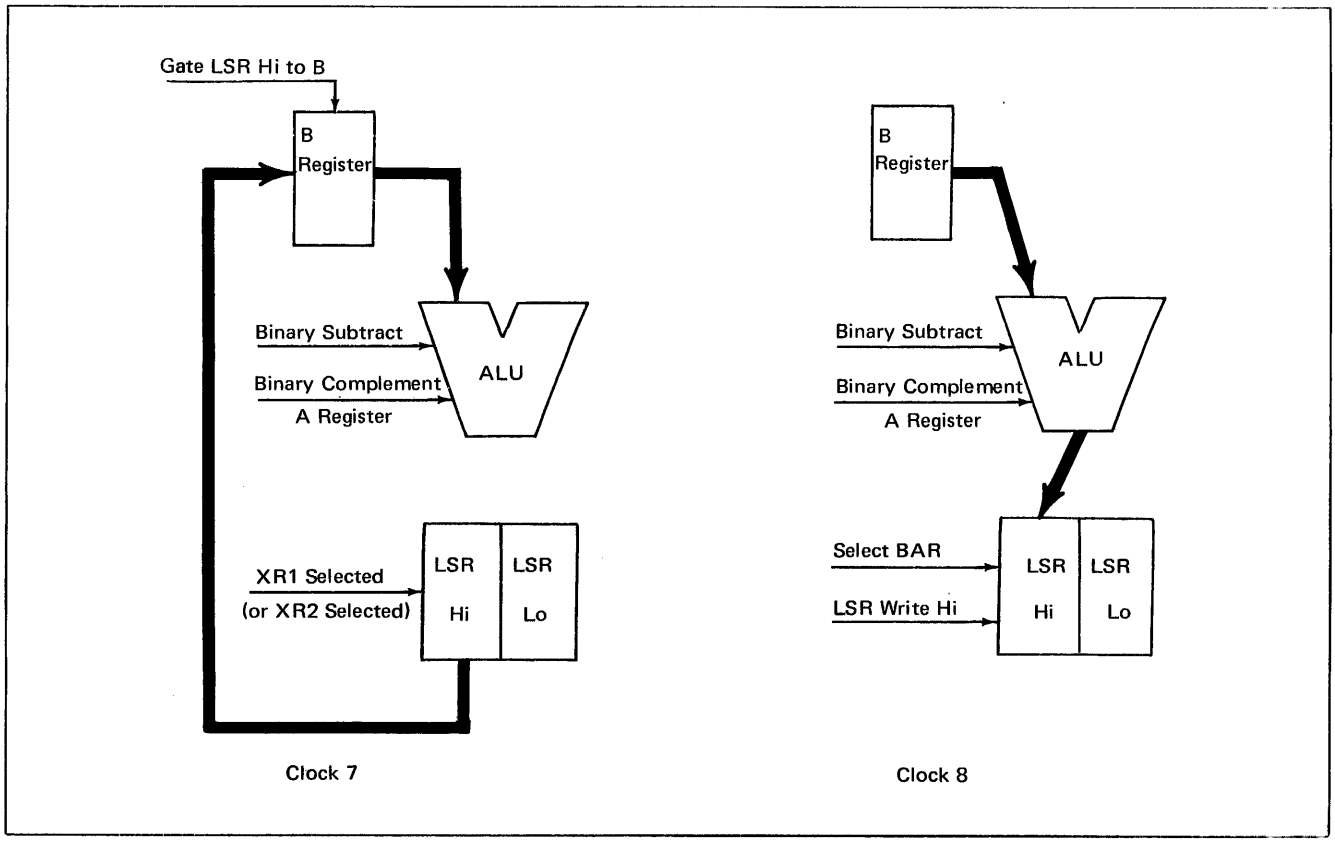


Figure 3-8. I-X1 Cycle–Indexing BAR High

low order of the IAR is decoded to determine if it contains F/F (Figure 3-9). If it does, a one is loaded in the A register to substitute for the carry. The high order position of the IAR is entered into the B register and the two are added in the ALU. The ALU results are then loaded in the high order position of the IAR. At clock 5 and 6 time the low order position of the IAR is incremented in the normal manner.

FEMD 5-040 contains the circuit description.

Execution Cycles

Because only one byte at a time can be removed from or placed into main storage, two cycles per byte are required when controlling data between two different storage locations. During the A cycle, the A field byte is removed from storage and retained. The B cycle is then used to remove the B field byte from storage and, depending upon the particular operation, to determine what to do with each byte.

The B cycle operation is covered under the individual operations but since the A cycle data flow is the same, regardless of the operation, it can be covered as a separate topic. Some operations require the condition register to be reset to equal during the first A cycle and some require the use of sign control for the A field character, but the basic data flow remains the same.

A Cycle

- Store A field byte in DRR.

The first step in an A cycle, as in all cycles, is to address the core storage location to be used during that cycle. At clock 0 time the contents of the AAR are transferred to the SAR in the same manner that the IAR was transferred.

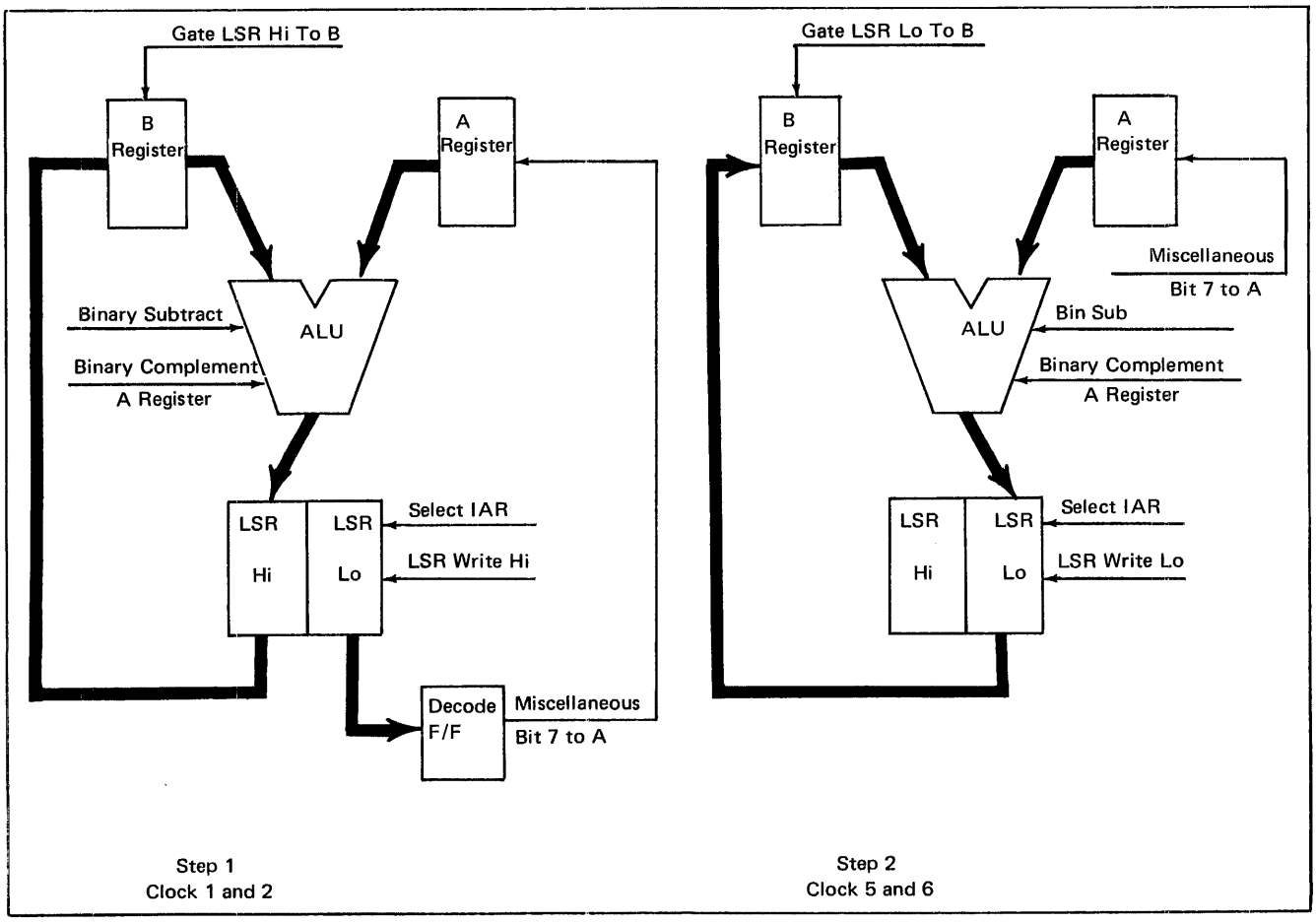


Figure 3-9. I-X Cycle–Incrementing IAR

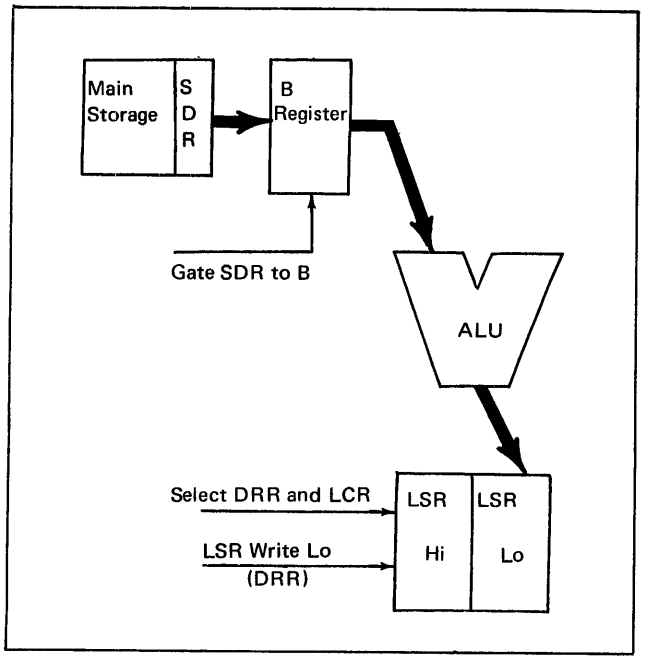


Figure 3-10. A-Cycle Storage to DRR Transfer

No data is transferred during clock 1 and 2 times as the CPU waits for the data to read from core storage and enter the SDR. During clock 3 and 4 times the byte is transferred through the B register and ALU and stored in the DRR (Figure 3-10). At clock 5, 'read write pulse' stores the SDR contents back into the same core storage location to regenerate the A field character.

The rest of the cycle is then used to decrement the AAR so that the next position of the A field can be addressed if necessary. Figure 3-11 shows that a 1 is subtracted from the AAR. Two steps are required because of the possibility of a carry from the low order to the high order position.

Refer to the operation flowcharts in the *IBM 5410 Central Processing Unit FEMD,* SY31-0202, for the circuit description.

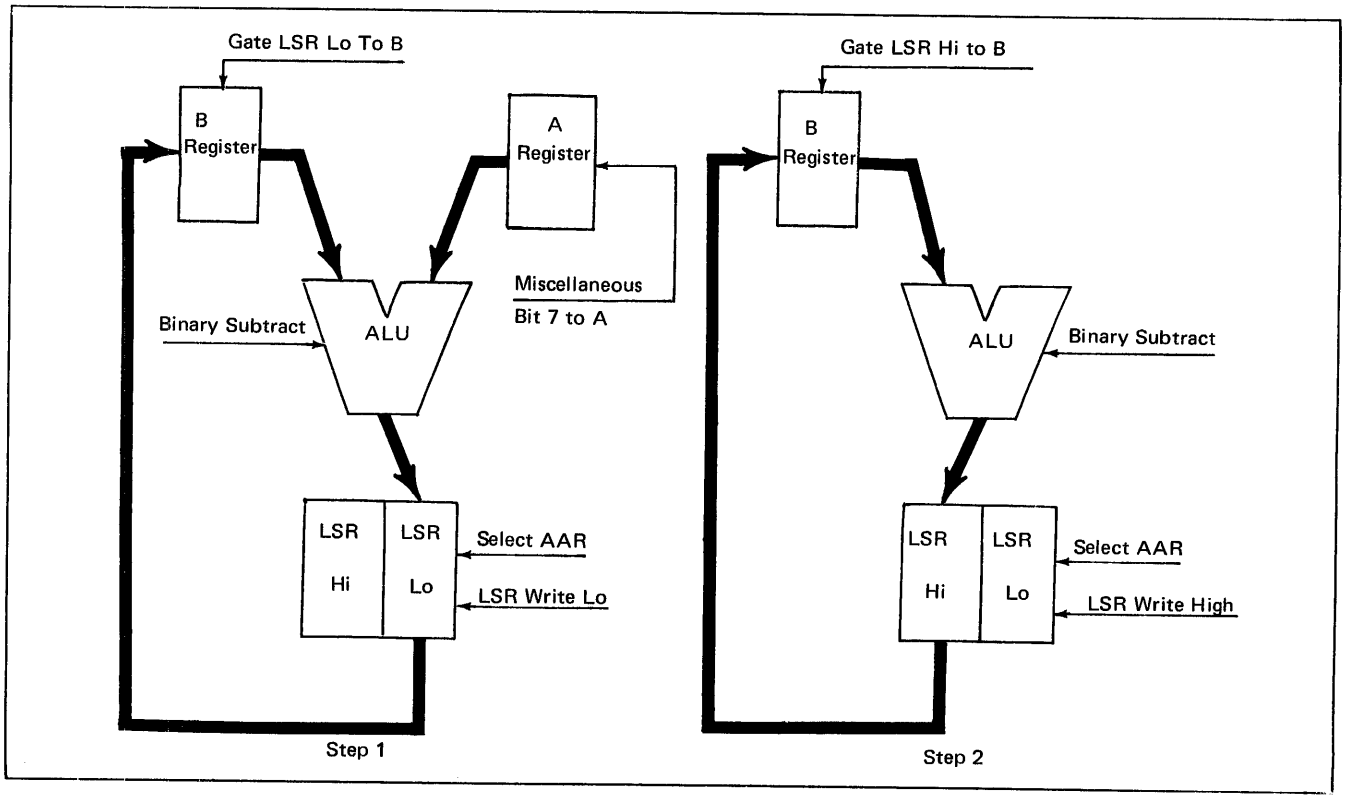


Figure 3-11. Decrementing AAR

Add Logical Characters—ALC

- Binary add A field to B field data.
- A and B fields are same length (Q code plus 1).

The add logical characters operation adds the A field data, one byte at a time, to the B field data. The entire A field byte (bits 7 through 0) is binary added to the B field byte. The operation begins with the low order position of each field and continues until the high order position is reached. Both fields are the same length, which is 1 more than the Q code.

The CPU performs the add logical characters operation with a series of A and B cycles. First an A cycle removes the first A field byte from storage and retains it in the DRR. Then a B cycle removes the first B field byte from storage, adds it to the A field byte, and stores the result in the B field units location. The next A field byte is then added to the second B field byte through the same process, and so on to the end of the field.

After the first A field byte has been stored in the DRR and the AAR has been decremented (refer to A cycle), the CPU enters into a B cycle. The BAR is selected and loaded into the SAR in the same manner that the IAR was transferred (Figure 3-1).

The B field units byte is read from storage and is loaded into the B register. The A field byte is transferred from the DRR to the A register and the two bytes are binary added in the ALU (Figure 3-12). Store new enters the result into the SDR and read call/write call then writes the new data into the B field units storage location.

The BAR is then decremented in the same manner that the AAR was decremented. The Q register is tested to see if the end of the field has been reached (blank Q register). If the Q register is not blank, the CPU takes another A cycle and another B cycle to add the next characters; if the Q register is blank, the 'op-end' trigger is turned on and the operation ends.

During clock 1 and 2 of each B cycle, except for the first, the LCR is decremented (Figure 3-13). The LCR contains the field length which was stored there during the I-Q cycle. The result, which is latched into the ALU at clock 2CD time, is loaded into the Q register at clock 3 time. By not decrementing the LCR on the first B cycle, the field length becomes 1 more than the Q code.

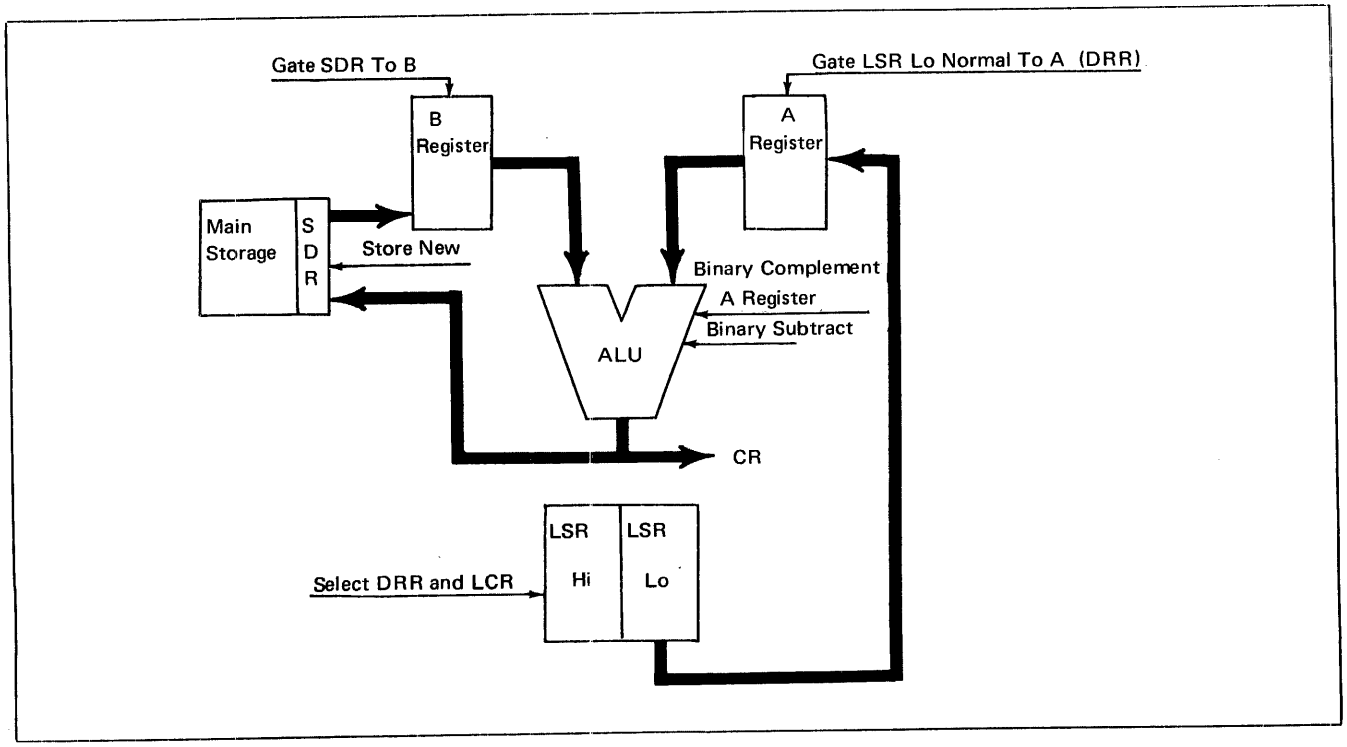


Figure 3-12. Add Logical Operation—Adding Characters

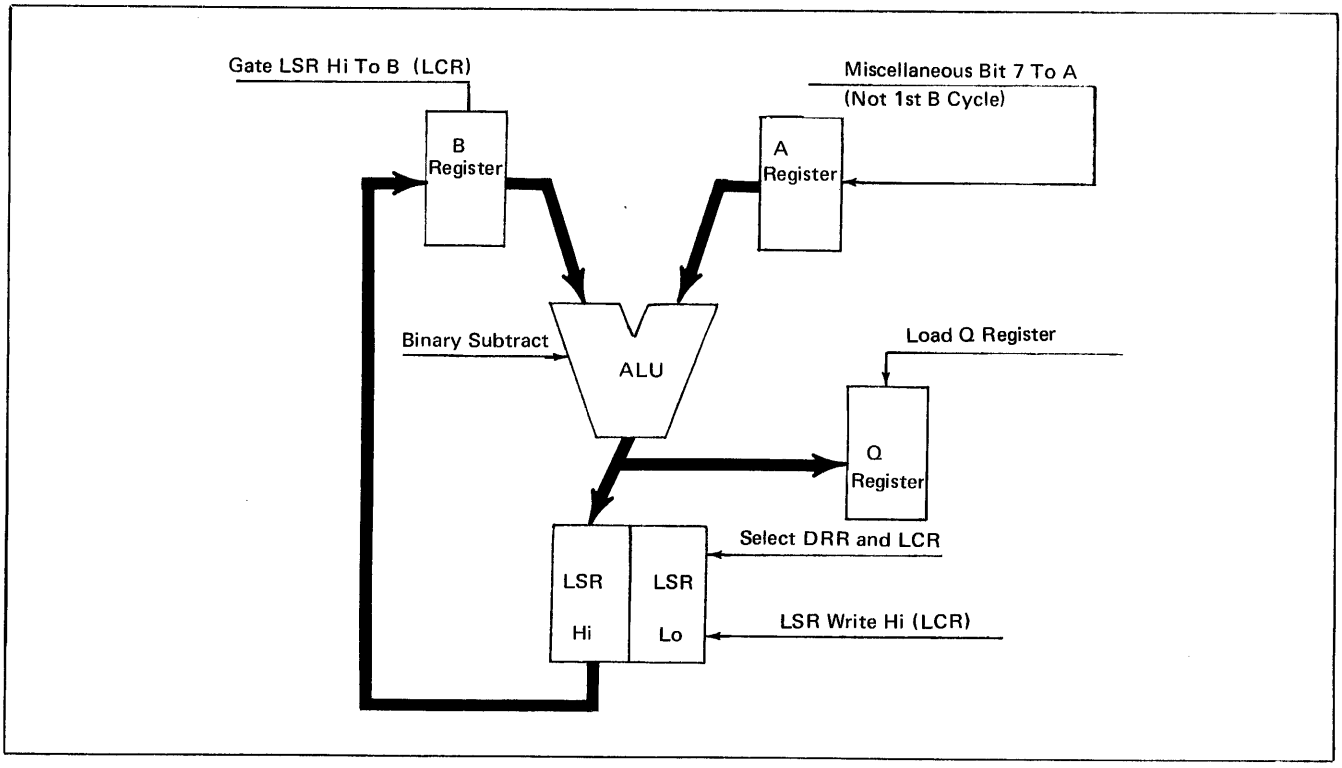


Figure 3-13. Decrementing Length Count–Equal Length Fields

An additional function of the add logical characters operation is to set the condition register. Figure 3-14 shows the condition register settings and the significance of each result.

During clock 1 and 2 of the first B cycle of the operation, the condition register is reset to equal. During each B cycle, after computing the A and B field data at clock 3 and 4 time, the ALU output is sampled. If the ALU output is all zeros, the condition register remains set to equal. However, if an ALU output occurs during any B cycle the result can no longer be equal and the equal condition is reset.

Once the equal condition has been reset the final high or low setting of the condition register is not determined until the last B cycle of the operation. In the meantime, because of the machine circuitry, a CR high condition will be indicated. During the last B cycle (Q register all zeros) if a carry results from the computation, the CR is set to low; if no carry occurs the condition register remains set to high.

If there is no carry from the high order position, the CR binary overflow condition is also set. This is an indication that the result is too large to be contained in the B field.

FEMD 5-080 contains the circuit description.

Equal	Low	High	Binary Overflow
Result is zero	No Carry and non-zero result	Carry and non-zero result	Result too large for field (no high order carry)

Figure 3-14. Condition Register–Add Logical Characters

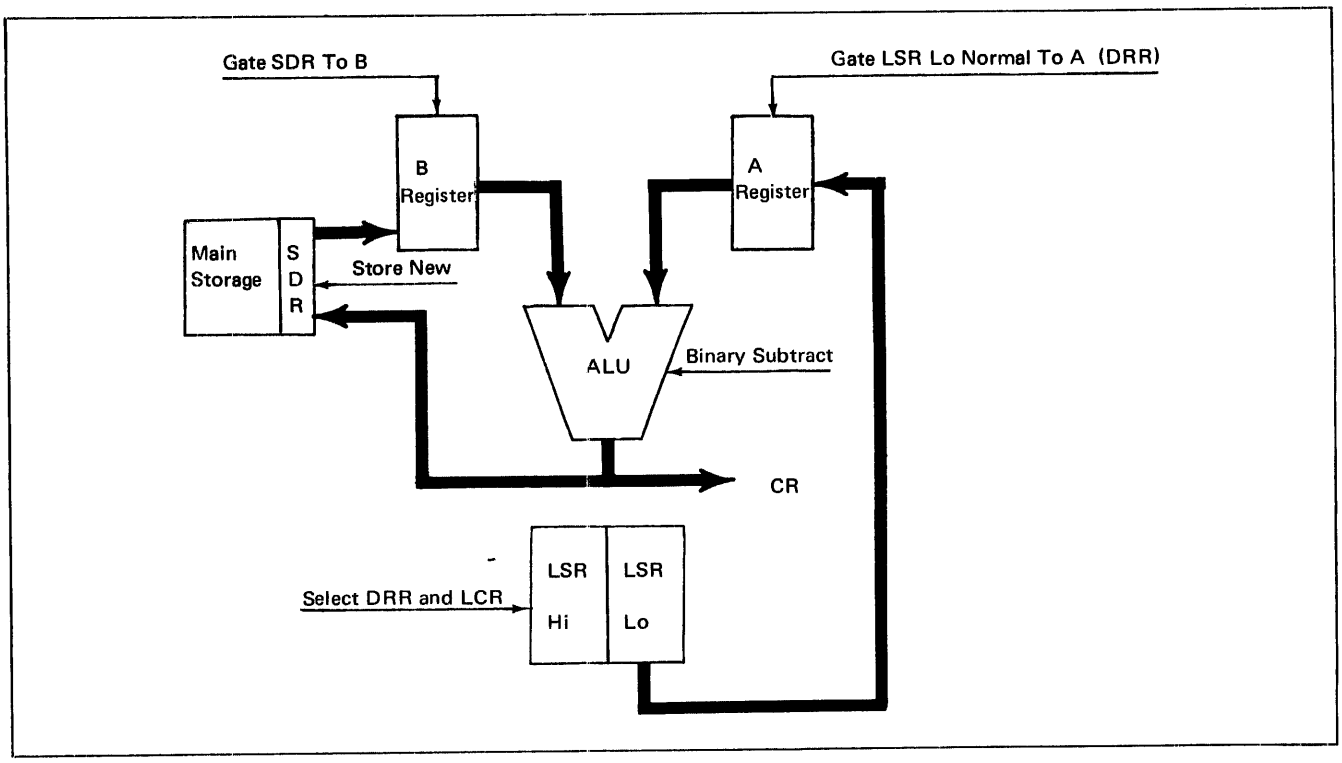


Figure 3-15. Subtract Logical Operation–Subtracting Characters

Subtract Logical Characters–SLC

- Binary subtract A field characters from B field characters.
- A and B fields are the same length (Q code plus 1).

The subtract logical characters operation is the same as the add logical characters operation except that the A field data is subtracted from the B field data (Figure 3-15). Figure 3-16 shows the significance of the CR settings. Although the settings have a different significance, the CR is set in the same manner as in the add logical characters operation, except the binary overflow is not set on.

FEMD 5-080 contains the circuit description.

Compare Logical Characters–CLC

- Compare A field data to B field data.
- A and B fields are the same length (Q code plus 1).

During the compare logical characters operation, the CPU compares the two fields by binary subtracting the A field data from the B field data. The operation is the same as a subtract logical operation except that the results are not entered into storage (Figure 3-17). Instead, the A and B fields remain unchanged by the operation and the ALU results are used merely to set the condition register (Figure 3-16).

FEMD 5-080 contains the circuit description.

Move Characters–MVC

- Move A field characters to B field location.
- A and B fields are the same length (Q code plus 1).

The move characters operation moves the A field data, one byte at a time, into the B field location. The operation begins with the low order position of each field and continues until the high order position is reached.

The operation is the same as add logical characters, except that the B field character is not loaded into the B register (Figure 3-18). With the B register blank, the operation is the same as adding the A field to zero.

Equal	Low	High
A and B fields are equal	B field is lower than A field	B field is higher than A field

Figure 3-16. Condition Register–Subtract or Compare Logical Characters

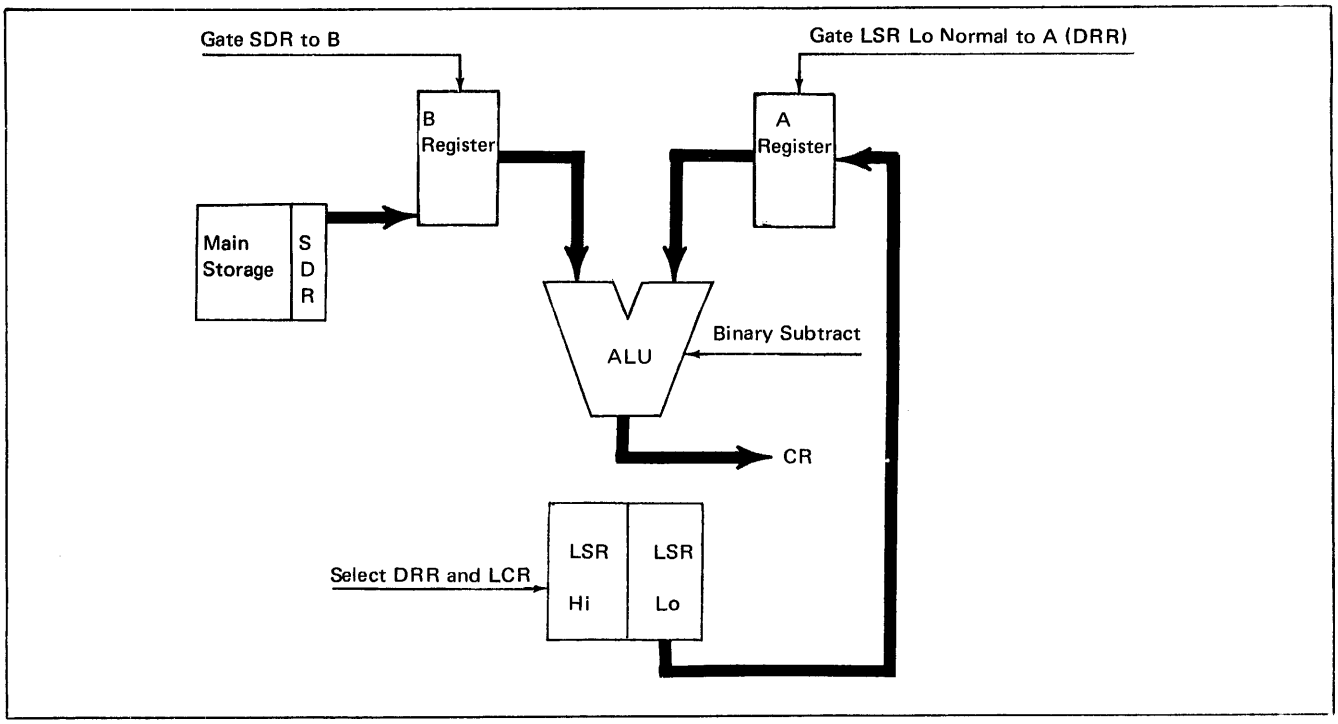


Figure 3-17. Compare Logical Operation–Comparing Characters

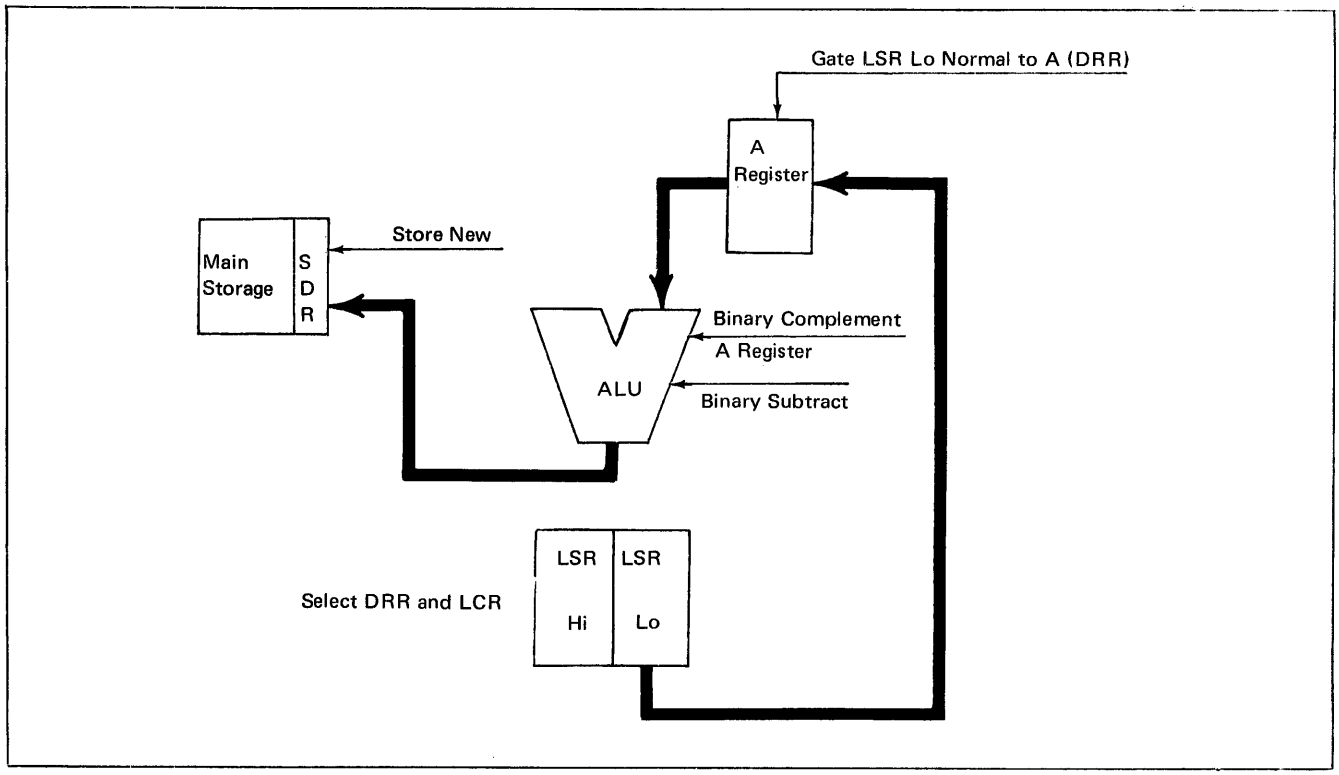


Figure 3-18. Move Character–Storing Data

The AAR, the BAR, and the LCR are decremented the same way and the operation ends in the same manner (Q register all zeros). However, the condition register setting is not changed.

FEMD 5-080 contains the circuit description.

Add or Subtract Zoned Decimal—AZ—SZ

- Decimal add A field data to B field data for add instruction with like signs and subtract instruction with unlike signs.
- Decimal subtract A field data from B field data for add instruction with unlike signs and subtract instruction with like signs.
- Signs are in zone portion of low order bytes.
- Length of A field is a numeric portion of Q code plus 1.
- B field is longer than A field by amount in zone portion of Q code.

Although the add zoned decimal and subtract zoned decimal operations have different operation codes, the execution of each operation is the same depending upon the signs of the two fields. For instance, an add instruction with unlike signs for the two fields (1 minus and 1 plus) actually subtracts the A field from the B field. Likewise, a subtract instruction with unlike signs actually adds the A field to the B field. In either instance, the operations are the same except for the add or subtract function of the ALU.

The operations begin with the low order position of each field and continues until the high order position of the B field is reached. First an A cycle removes the first A field byte from storage and retains it in the DRR. Then a B cycle removes the first B field byte from storage, transfers it to the B register, and adds or subtracts the registers in the ALU. This process continues until the end of the A field, which can be shorter than the B field, and then B cycles continue to the end of the B field.

If the result is in true form, nothing more needs to be done with it. But if the result is in complement form, the result must be recomplemented. Results are in complement form when:

- Operation has a subtract function (no A register complement) and the A field is larger than the B field.
- Result is minus zero.

Recomplement begins with the low order position of the B field and continues to the high order position. Continuous B cycles remove the bytes from storage and recomplement them in the ALU.

The numeric portion of the Q code plus 1 is the length of the A field. The B field is longer than the A field by the amount in the zone portion of the Q code.

After an A cycle has stored the first A field byte in the DRR and the AAR has been decremented the CPU enters into a B cycle. During the B cycle, the first B field byte is read from storage and is loaded into the B register. The A field byte is transferred from the DRR to the A register and, depending upon the operation code and the signs of the two fields, the two bytes are either decimal added or subtracted in the ALU (Figure 3-19). Store new enters the result into the SDR and read call/write call writes the new data into the B field storage location.

The BAR is decremented and the Q register is tested to see if the end of the A field (Q register numeric portion all zeros) or if the end of the B field (Q register all zeros) has been reached. If the numeric portion of the Q register is not all zeros, the CPU takes another A cycle and another B cycle to add or subtract the next characters. If the numeric portion is all zeros but the zone portion still contains bits, 'EA eliminate' is activated to block A cycles and the CPU takes a B cycle.

If the Q register is all zeros, a check is made to determine if the total is in true or complement form. If the operation function is decimal subtract (A register not complemented) and a carry occurs from the high order byte, 'recomplement cycle' is activated to start recomplementing. Under all other conditions, except a minus zero result, the 'op-end' trigger is turned on and the operation ends. A minus zero result, which is also recomplemented, is determined by the CR setting and is covered later.

During clock 1 and 2 of each B cycle, except for the first, the LCR is decremented (Figure 3-20). The LCR contains the length count which was stored there during the I-Q cycle. The result, which is latched into the ALU at 2CD time, is loaded into the Q register at clock 3 time. Until the Q register numeric portion is all zeros, the LCR is decremented with a 7 bit; after the numeric portion is all zeros, decrementing is done with a 3 bit (Figure 3-20).

During each A and B cycle, 'sign control' is activated to provide the EBCDIC code for the sign of each field. The sign is in bits 0-3 of the first byte of each field. EBCDIC sign for minus is 1101 and for plus is 1111. The CPU also recognizes the ASCII-8 code for minus (1011) but 'sign

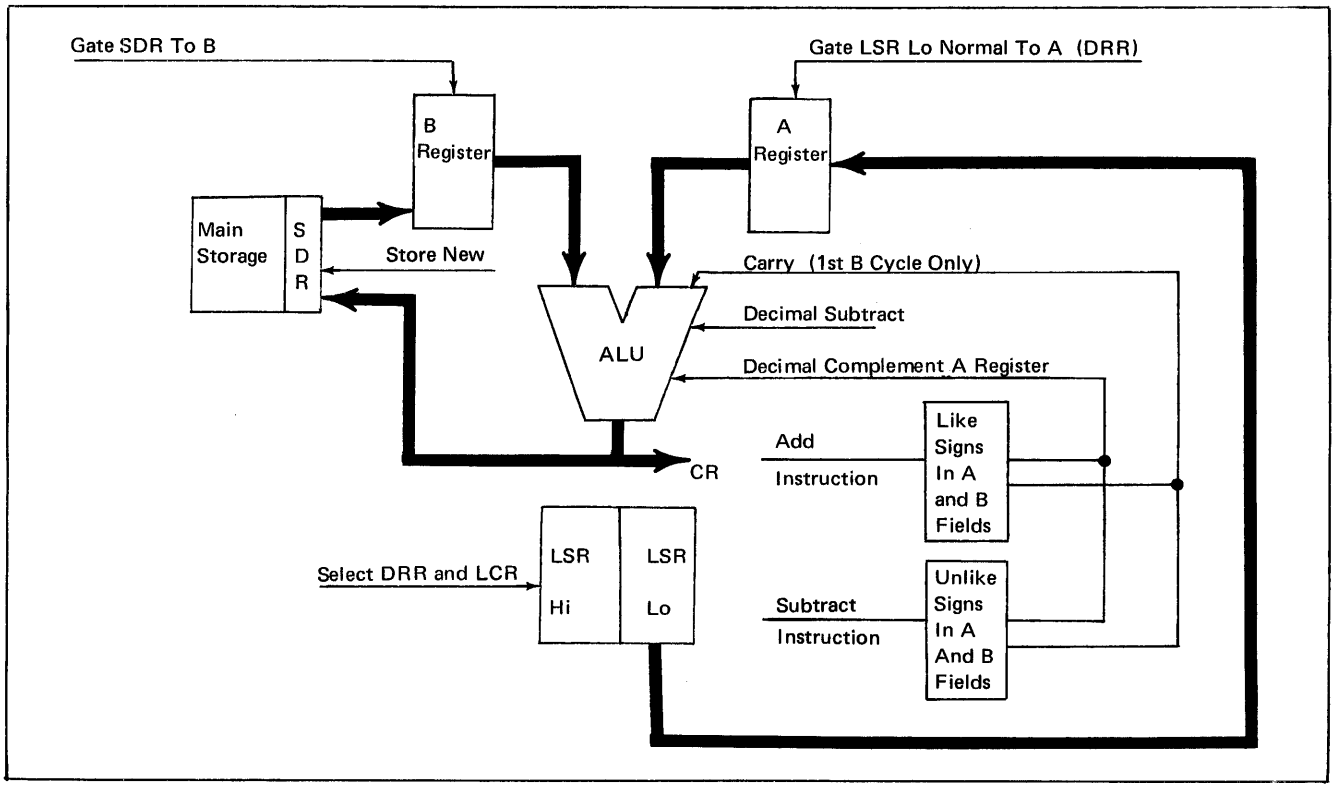


Figure 3-19. Add or Subtract Zoned Decimal–Add or Subtract Characters

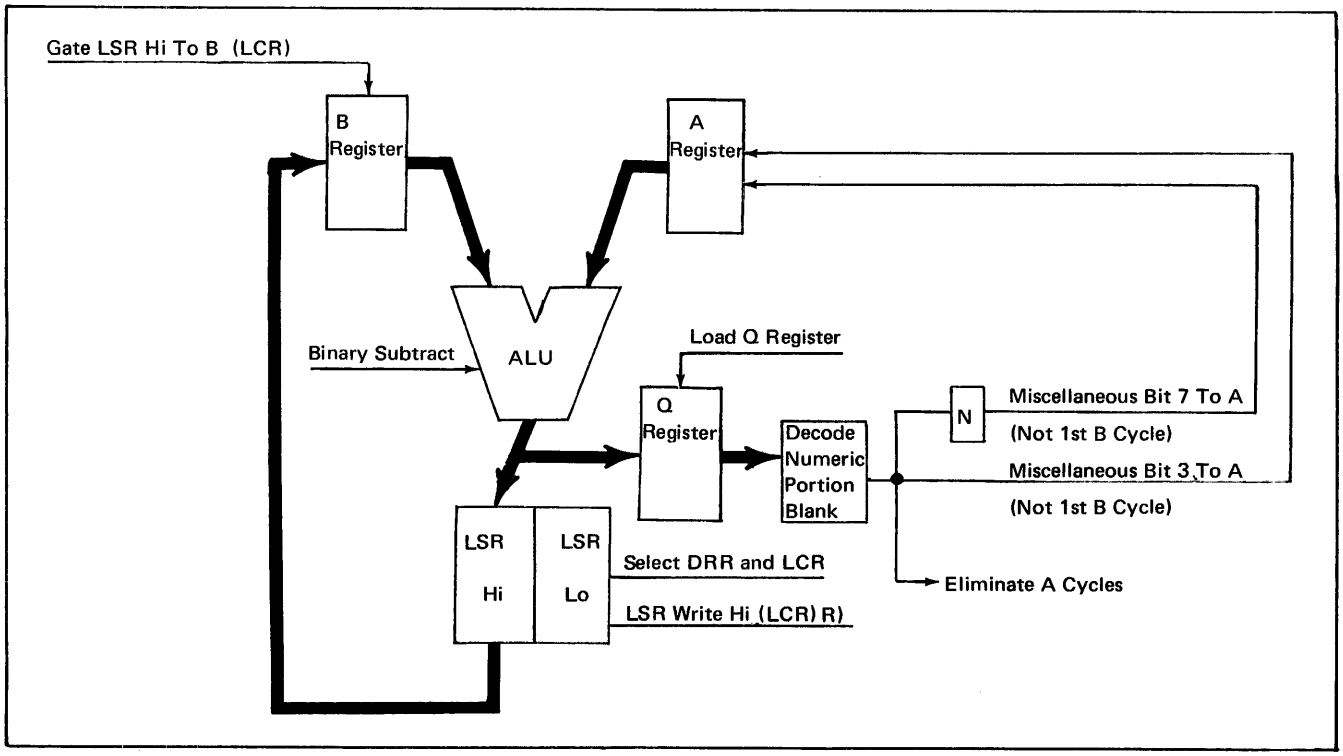


Figure 3-20. Decrementing Length Count–Unequal Length Fields

control' changes this to EBCDIC. After the first byte of each field, all zone bits (1111) are provided for each character. During the first B cycle, the sign of the B field is entered into storage.

During clock 1 and 2 of the first A cycle, the condition register is reset to equal. Then in the first B cycle, the 'CR lo/hi' latch is set by the result sign (lo for minus, hi for plus). However, if no numeric output occurs from the ALU (all zeros), the condition register remains set to equal. If, during any B cycle, a non-zero ALU output occurs, the result can no longer be equal and the equal condition is reset. The setting of the 'CR lo/hi' latch is then used to determine the CR setting.

If the CR equal condition has not been reset before the last B cycle and the 'CR lo/hi' latch is set to lo (minus) the result is minus zero. All zero results are considered plus so 'recomplement cycle' is activated to recomplement the results.

If the operation is an add function (decimal complement A register) and no carry occurs from the high order position, the CR decimal overflow condition is also set. This is an indication that the result is too large to be contained in the B field. Figure 3-21 shows the significance of all the CR settings.

FEMD 5-100 contains the circuit description.

Recomplement

Addressing for recomplement cycles is controlled by the ARR which contains the low order address of the B field (refer to I-H1 and I-L1 cycles). The ARR is decremented each recomplement cycle in the same manner that the AAR and BAR are decremented in other operations.

'EA eliminate' is active through the entire recomplement operation causing continuous B cycles. Each byte is read from storage and loaded into the B register (Figure 3-22). The A register has a 1 forced into it on the first recomplement cycle and is left with all zeros for the remainder of the cycles. Both the A and B registers are decimal complemented.

The length of the field is determined by the LCRR which was loaded during the I-Q cycle. Decrementing of the LCRR is the same as for the LCR in a decimal add or subtract operation (Figure 3-20).

Condition Register	Equal	Low	High	Decimal Overflow
ALU result	Result is zero	Result is minus	Result is plus	Result too large for field

Figure 3-21. Condition Register–Add or Subtract Zoned Decimal

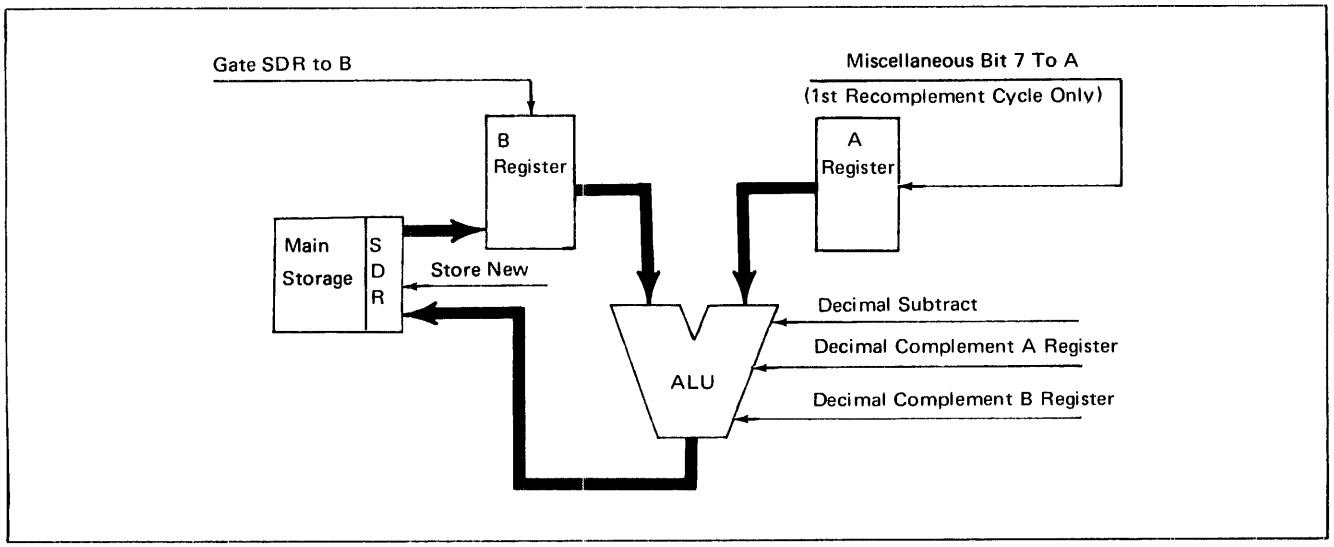


Figure 3-22. Recomplementing

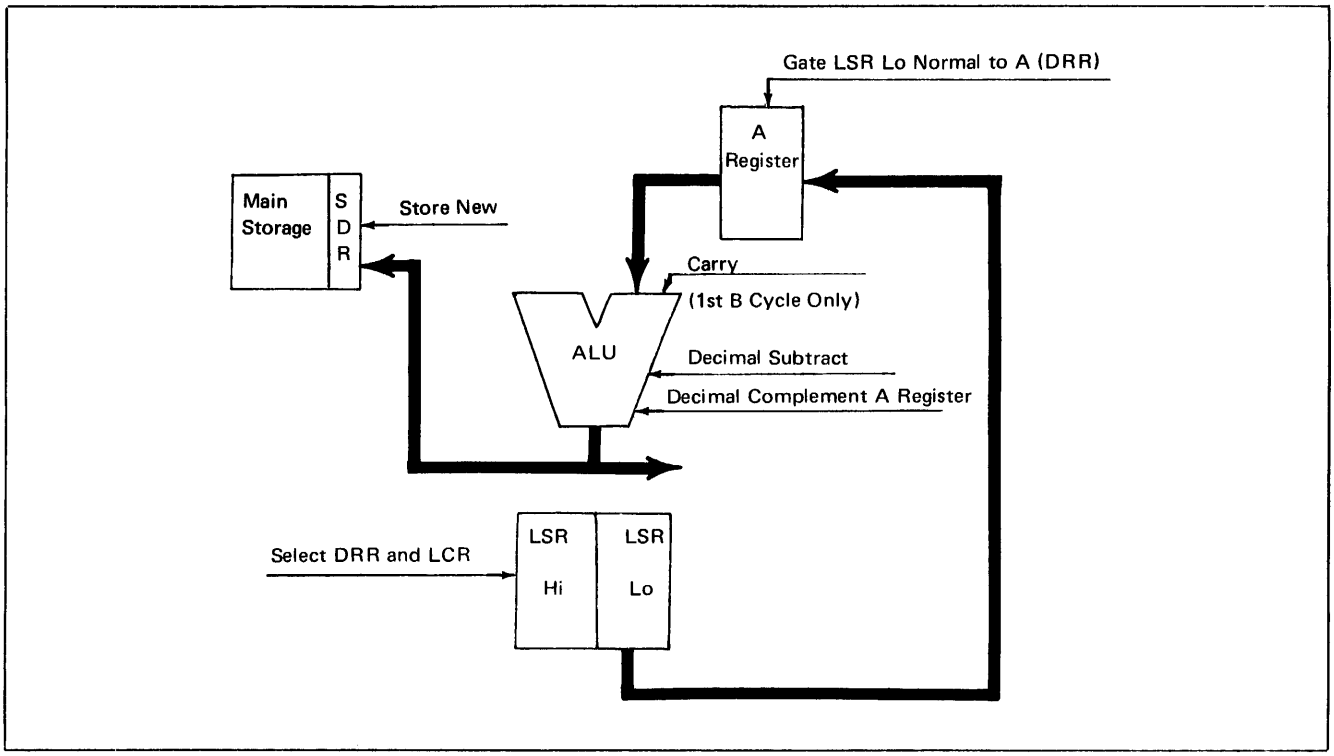


Figure 3-23. Zero and Add Zoned–Adding Characters

The Q register is tested each cycle to see if the end of the field has been reached (all zeros in the Q register). When the Q register is all zeros, the 'op-end' trigger is turned on and the operation ends.

Because recomplement is necessary only when the A field is larger than the B field or the result is minus zero, 'sign control' is activated to reverse the sign of the result. The 'CR lo/hi' latch is reset on the first cycle and the CR setting is determined in the same way as during decimal add or subtract.

FEMD 5-100 contains the circuit description.

Zero and Add Zoned–ZAZ

- Decimal add A field data to zeros and place result in B field.
- B field is longer than A field by amount in zone portion of the Q code.

The zero and add zoned operation is similar to an add zoned decimal operation except the function is always add, without consideration of the fields signs. Another difference is that the B field characters are not loaded into the B register (Figure 3-23). With the B register all zeros, the operation is the same as adding the A field to zero. The only other significant difference is that the A field sign is entered as the sign of the result instead of the B field sign.

The address registers and the LCR are decremented the same way and the operation ends in the same manner. Recomplementing is not necessary unless the result is minus zero. The CR settings are shown in Figure 3-21.

FEMD 5-100 contains the circuit description.

Edit–ED

- Replace hex 2/0 in B field with A field data.
- Skip other characters in B field leaving them as they were.
- Length of B field is Q code plus 1.

An edit operation inserts punctuation (decimal points, commas, or other symbols) into an amount field. Figure 3-24 shows an example of an edit operation. The A field represents the total nine-hundred seven dollars and fifteen cents. The B field is the pre-established edit pattern. The total is then moved into the edit pattern to replace those positions that contained a replaceable character (2/0).

The operation begins with the low order position of each field and continues until the high order position is reached. First, an A cycle removes the first A field byte from storage and retains it in the DRR. Then a B cycle removes the first B field byte from storage, transfers it to the B register, and checks to see if the character is a replaceable character. Only a hex 2/0 is recognized. If the character is 2/0, the A field character is stored in that location; if the character is not 2/0, the A field character is retained and the next B field character is checked.

The Q code plus 1 is the length, in bytes, of the B field. The A field characters are assumed to be decimal numeric, and their zone portions are all set to F before entering them into the B field. However, the sign of the A field before the operation is used to control the setting of the condition register. Figure 3-25 shows the significance of the condition register settings.

After an A cycle has stored the first A field byte in the DRR, the CPU enters into a B cycle. During the B cycle the A field byte is transferred from the DRR to the A register. The first B field byte is read from storage and is loaded into the B register. The 'AND' and 'OR' lines are activated to move the A register byte through the ALU (Figure 3-26). The B register is checked to see if it contains the character 2/0. If it does, 'store new' enters the ALU result into the SDR and 'read call/write call then writes the new data into

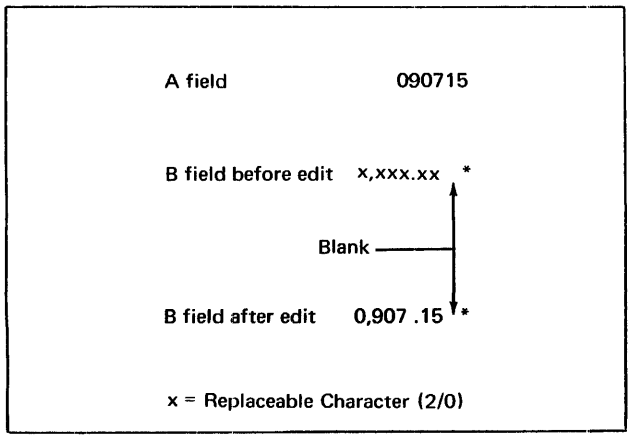


Figure 3-24. Edit

Equal	Low	High
A field is zero	A field is negative	A field is positive

Figure 3-25. Edit–Condition Register

the B field units storage location. Since the A field byte was stored, the machine takes another A cycle to read out the next A field character and store it in the DRR.

If the B register byte was not 2/0, the ALU output is not entered into the SDR and the B register byte is regenerated back into main storage. In this case, 'EA eliminate' is activated, the A field byte is retained in the DRR, and the machine takes another B cycle to read the next B field byte from storage.

During each B-cycle, except the first B-cycle, the LCR is decremented. The LCR contains the B field length count which was stored there during the I-Q cycle. The result, which is latched into the ALU at clock 2CD time, is loaded into the Q register at clock 3 time.

The Q register is tested each B cycle to see if the end of the field has been reached (all zeros Q register). If the Q register is all zeros, the 'op end' trigger is turned on and the operation ends. By not decrementing the LCR on the first B cycle, the B field length becomes one more than the Q code.

Figure 3-27 shows the cycles required to complete a typical edit operation. During the first A cycle, the A field low order byte, in this case a 5, is stored in the DRR. During the following two B cycles, as the asterisk and space are read from storage, the 5 is retained in the DRR. On the third B cycle, when the replaceable character is read from storage, the 5 replaces the 2/0 in the B field location. Another A cycle follows to read out the next A field character, and so on until the length count is reduced to zero.

During clock 1 and 2 of the first A cycle, the condition register is set to equal. The sign of the A field (which is contained in the zone portion of the low order byte) is checked while the byte is in the B register. If the sign is minus the CR low latch is turned on; if not the latch is left off. During each B cycle, after computing the A and B data at clock 3 and 4 time, the ALU output is sampled. If the ALU output is all zeros, the condition register remains set to equal and the equal condition takes precedence over the sign of the field. However, if an ALU output occurs during any B cycle the result can no longer be equal and the equal condition is reset.

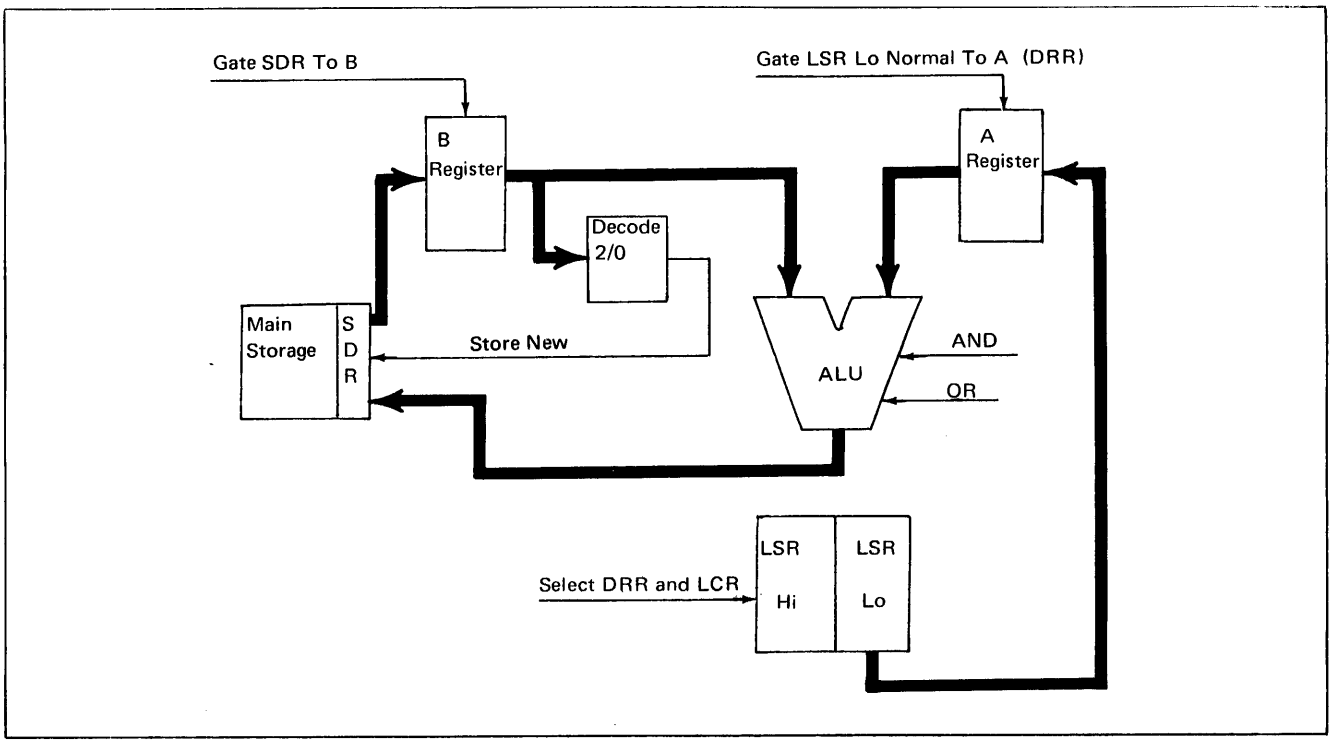


Figure 3-26. Edit–New Data to Storage

Cycle	A	B	B	B	A	B	A	B	B	A	B	A	B	A	B	B
B register	5	*	ß	x	1	x	7	.	x	0	x	9	x	0	,	x
A register		5	5	5		1		7	7		0		9		0	0
Data recall register	5	5	5	5	1	1	7	7	7	0	0	9	9	0	0	0
Regenerate	5	*	ß		1		7	.		0		9		0	,	
New data				5		1			7		0		9			0
Length count	9	9	8	7	7	6	6	5	4	4	3	3	2	2	1	0

x = Replaceable Character (2/0)

A field 090715
B field before edit x,xxx.x *
B field after edit 0,907 .15 *
blank

Note: Since the A and B registers are loaded each odd CD clock time, the figures shown apply only to clock 3 and 4 time when the main storage data is being analyzed.

Figure 3-27. Edit Cycles

Once the equal condition has been reset, the final high or low setting of the condition register is determined by the CR low latch. If the CR low latch is on, a CR low condition is indicated; if the CR low latch is off, a CR high condition is indicated.

During each A cycle, as the A field byte goes through the ALU, 'sign control' is activated. Thus, each byte is entered into the DRR as a decimal numeric character (zone bits all present).

FEMD 5-110 contains the circuit description.

Insert and Test Character—ITC

- Replace all characters to left of first significant digit in B field with A field character.
- Only numeric characters 1 through 9 are considered significant digits.

The insert and test character operation inserts a single A field character into each B field position to the left high order significant digit. Only numeric characters 1 through 9 are considered to be significant digits. Figure 3-28 shows an example of an insert and test character operation. The B field starting address is the high order position and the operation continues until either, the low order position of the field is reached or, a significant digit is found. The B field length is one more than the Q code.

After the A field byte has been stored in the DRR the CPU enters into a B-cycle. The B field high order byte is read from storage and is loaded into the B register. The A field byte is transferred from the DRR to the A register and the 'AND' and 'OR' ALU control lines are activated to move the A field byte through the ALU (Figure 3-29). If the B register contains a character other than numeric 1 through 9, the 'store new' enters the A field byte into the SDR and the read call/write call writes the new byte into the B field high order location.

A field character	*
B field before operation	0,907.15 *
B field after operation	**907.15 *

Figure 3-28. Insert and Test Character

The BAR is then incremented in the same manner that the IAR is incremented during I-cycles (Figure 3-3). 'EA eliminate' prevents the CPU from taking another A cycle. Another B cycle reads the next descending B field position from storage and it is checked in the same manner.

If the B field byte contains a significant digit the B field byte is regenerated back into storage and the 'op end' trigger is turned on and the operation ends. Meanwhile, the LCR is decremented each B cycle, except the first B cycle, and is transferred to the Q register. If no significant digit is found before the length count is reduced to zero, the all zero Q register ends the operation.

Each B cycle in which no significant digit is found, the address of the next B field byte is stored in the ARR for programming purposes. At the end of the operation, the ARR will contain the address of the first significant digit. If no significant digit is encountered, the ARR will contain the address of the byte to the right of the B field.

FEMD 5-120 contains the circuit description.

Move Hex Character—MVX

- Move half (zone or numeric) of A address byte to the numeric or zone portion of the B address byte.
- Do not change the other half of the B address byte.
- Bits 6 and 7 of Q code specify the portion of each byte used.

The move hex character operation moves a half byte of information from one core storage location to another. Either half of the A field byte (zone or numeric) may be placed in either portion of the B field byte (zone or numeric) without changing the other half of the B field byte. The type of move is determined by bits 6 and 7 of the Q code. Figure 3-30 shows an example of the four types of moves possible and gives the Q code bit structure for each type of move. Since each field is only one character in length, one A cycle and one B cycle are all that are required to complete the operation.

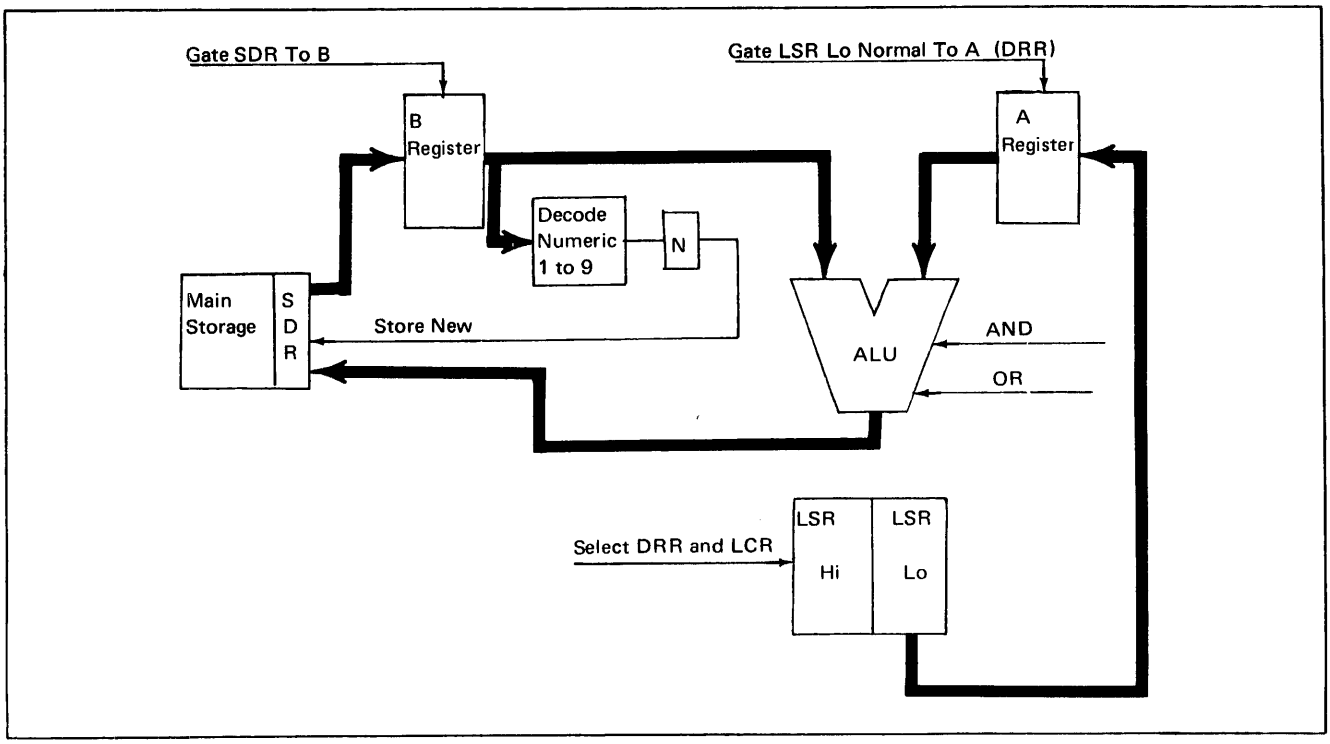


Figure 3-29. Insert and Test Character-New Data to Storage

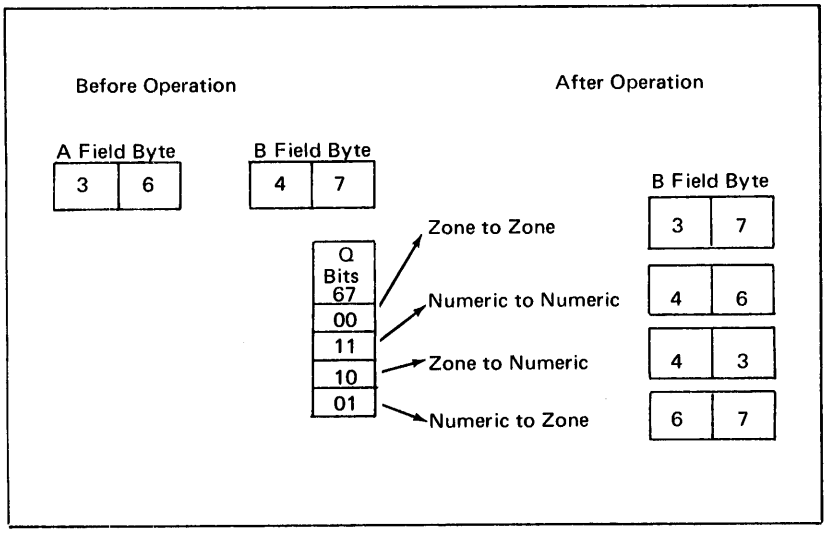


Figure 3-30. Move Hex Character

After an A cycle has stored the A field byte in the DRR the CPU enters into a B cycle. During the B cycle, bits 6 and 7 of the Q code control the data flow. Figure 3-31 shows that if the half byte is to be moved to the same relative position in the B field byte, the bits are transferred from the DRR to the A register in their normal positions. However, if the move is to the opposite portion of the B field byte, the A field byte enters the A register with the zone and numeric portions crossed. For example, a 3 bit enters the A register as a 7 bit, a 2 bit enters as a 6 bit and so forth.

The portion of the B register byte that is to be changed by the A field half byte determines the ALU controls. The 'AND' and 'OR' ALU control lines are activated only for that portion that is to be changed (Figure 3-32). The other half of the B register byte passes through the ALU without any ALU controls and the new byte is entered into storage. At clock 8 time the 'op end' trigger is turned on and the operation ends.

FEMD 5-070 contains the circuit description.

ONE ADDRESS INSTRUCTIONS

I-Cycles

- Load operation code into op register.
- Load Q code into Q register and DRR.
- Load B field address into BAR except for load address instruction.
- Load index register for load address instruction.
- Load B field address into ARR for branch instruction.

Single address instructions require a maximum B-field length of two bytes. Therefore, it is not necessary to maintain a field length count for them. Thus, the Q code is freed for use in controlling the functions necessary to execute the single address instructions. Use of the Q code is covered under the individual instructions.

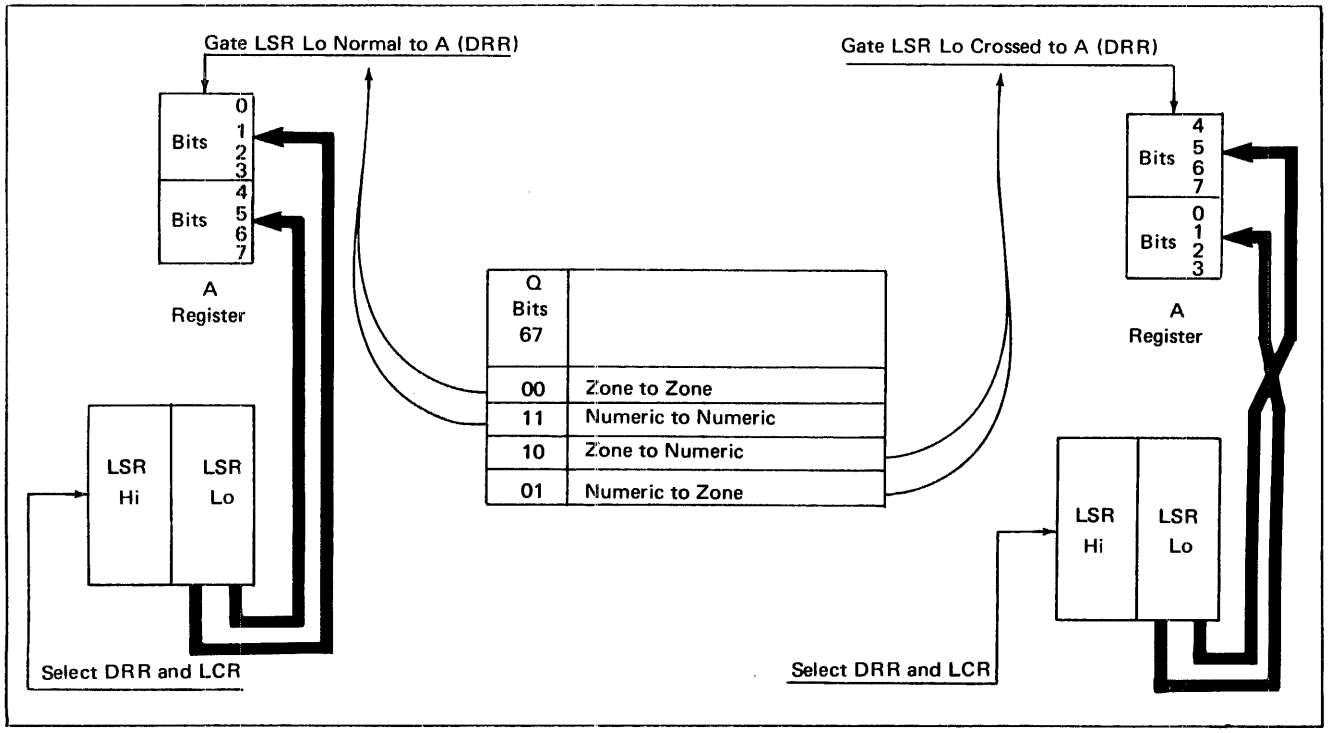


Figure 3-31. Hex Character to A Register Data Flow

I-cycles for single address instructions are either three or four cycles in length. First, an I-op cycle transfers the operation code from main storage to the op register. Second, an I-Q cycle transfers the Q code into the Q register and the DRR.

Two cycles, an I-H1 and an I-L1 cycle, are then used to load the B field address into the BAR. For branch instructions, the B field address is also loaded into the ARR. For a load address instruction, an index register is loaded instead of the BAR. This is covered under 'Load Address'.

If indexing is used, a single I-X1 cycle replaces the I-H1 and I-L1 cycles.

I-op and I-Q cycles are the same as in 2 address instructions except the Q code is stored in the DRR instead of the LCR and LCRR. I-H1 and I-L1 cycles are the same unless the instruction is a load address instruction (refer to 'Load Address').

The need for an I-X1 cycles is determined by (1) either op bit 0 or 1, but not both, or (2) bit 2 or 3, but not both. The I-X1 cycle description and the index register selected are the same as in two address instructions.

Any additional considerations made during the I-cycles are covered under the individual operations. FEMD 5-010, 5-030, and 5-040 contain the circuit descriptions.

Move Logical Immediate—MVI

- Move the Q code byte to the B address storage location.

The move logical immediate operation moves the byte of data which is contained in the Q code portion of the instruction to the B address storage location. Since only one byte is being moved, the operation is executed with a single B cycle.

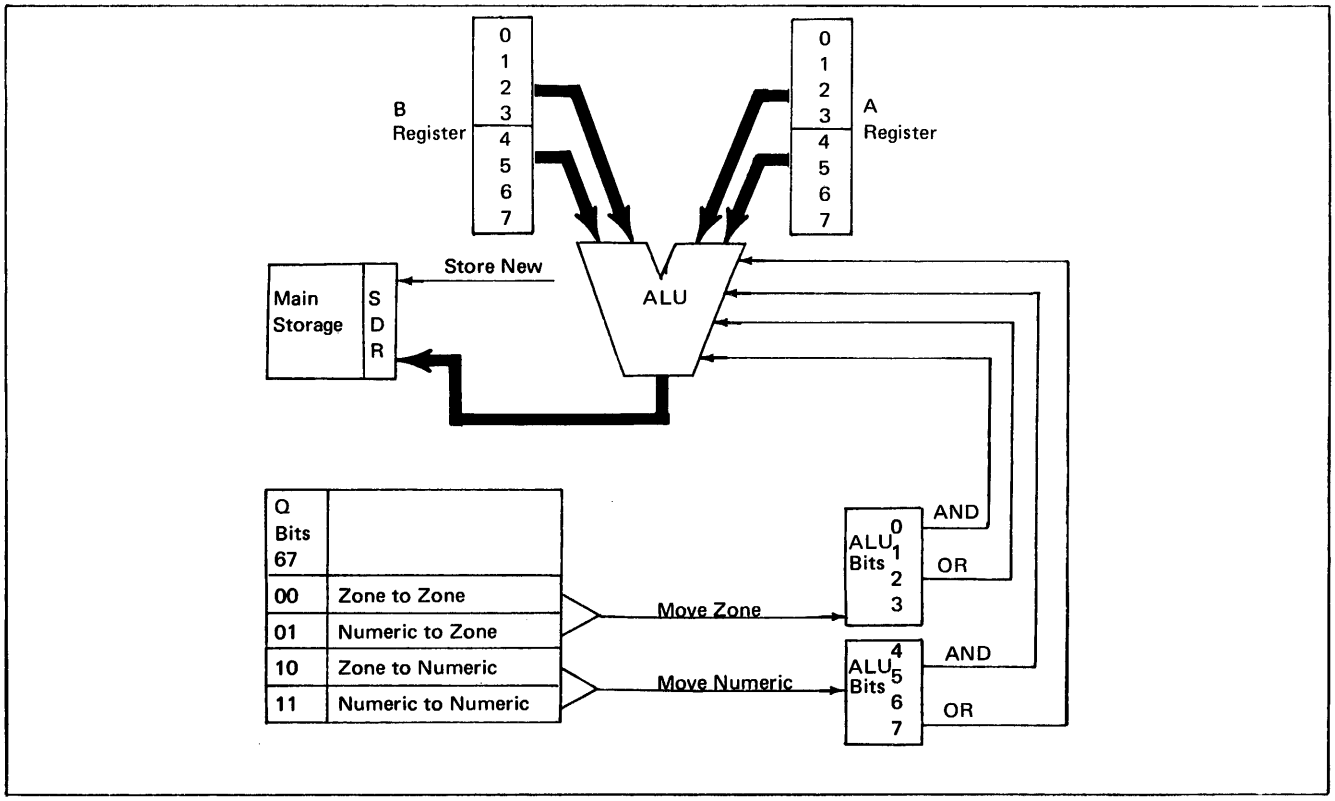


Figure 3-32. Hex Character to Storage Data Flow

During the B cycle, the storage location is addressed by the BAR, and at clock 3 and 4 time the DRR is transferred to the A register (Figure 3-33). The DRR contains the Q code which was stored there during the I-Q cycle. The data in storage, if any is present, is not transferred to the B register and the A register is binary added to the zeros in the B register. 'Store new' enters the result into the SDR and read call/write call writes the byte into the B address location. The 'op-end' trigger is then turned on and the operation ends.

FEMD 5-090 contains the circuit description.

Compare Logical Immediate—CLI

- Compare Q code with byte in B address storage location.

The compare logical immediate operation compares the byte of data which is contained in the Q code portion of the instruction with the B address storage location byte. Since only one storage position is involved, the operation requires a single B cycle.

The operation is similar to a move logical operation except the B address byte is loaded into the B register and the Q code byte is binary subtracted from it (Figure 3-34). The results are not entered into storage but are used merely to set the condition register. Figure 3-35 shows the significance of the CR settings.

FEMD 5-090 contains the circuit description.

Set Bits On Masked—SBN

- Place bits that are present in the Q code into the B address storage location.
- Do not change the remainder of B address byte.

The set bits on masked operation, turns on the bits in the B address storage location that correspond to the Q code bits. Bits that were already on in the B address byte are left on. Figure 3-36 gives an example of a set bits on masked operation.

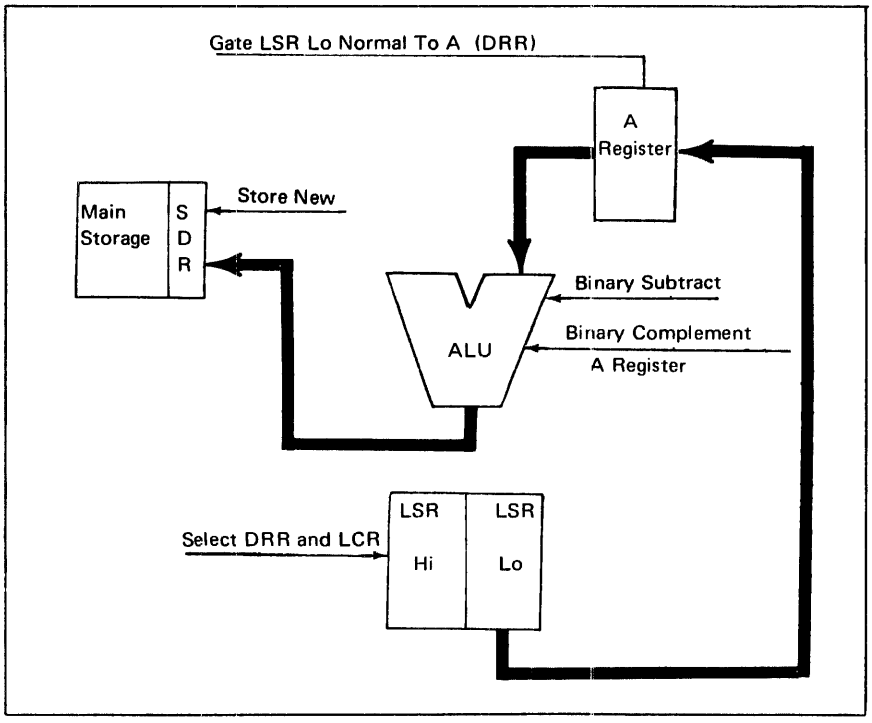


Figure 3-33. Move Logical Immediate Data Flow

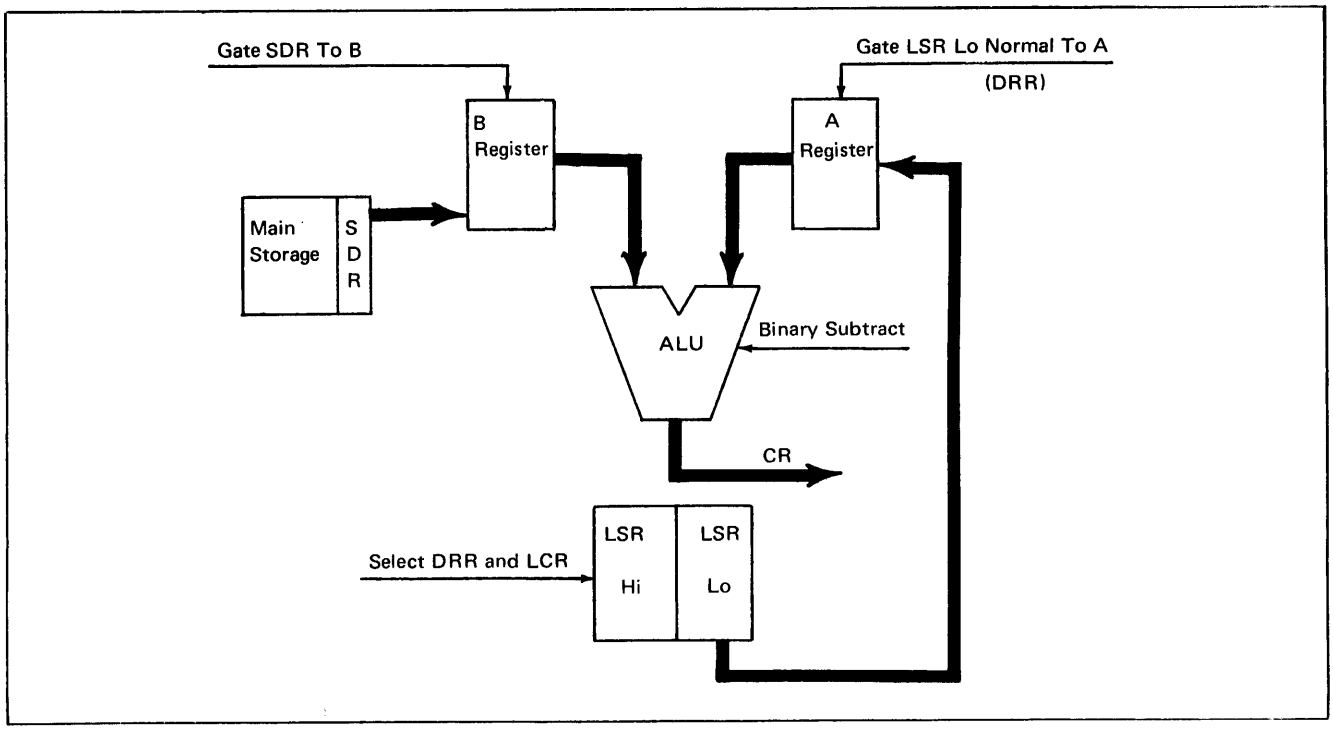


Figure 3-34. Compare Logical Immediate–Comparing Characters

Equal	Low	High
B address and Q bytes are equal	B address byte is lower than Q byte	B address byte is higher than Q byte

Figure 3-35. Condition Register–Compare Logical Immediate

Q Code Byte	11000111
B Address Byte	10010010
New B Address Byte	11010111

Figure 3-36. Set Bits On Masked

The operation requires a single B-cycle and 'ORs' the Q code byte, which was stored in the DRR during the I-Q cycle, with the B address byte (Figure 3-37). Store new enters the result into the SDR and read call/write call writes the byte into storage. The op-end trigger is turned on and the operation ends.

FEMD 5-050 contains the circuit description.

Set Bits Off Masked—SBF

- Remove bits that are present in the Q code from the B address storage location.
- Do not change the remainder of the B address byte.

The set bits off masked operation turns off the bits in the B address storage location that correspond to the Q code bits. The rest of the bits in the B address byte are left unchanged. Figure 3-38 gives an example of a set bits off masked operation.

Q code byte	11000111
B address byte	10010010
New B address byte	00010000

Figure 3-38. Set Bits Off Masked

The operation requires a single B-cycle and 'ANDs' the B address byte with the binary complement of the Q code (Figure 3-39). The Q code was stored in the DRR during the I-Q cycle of the operation. Store new enters the result into the SDR and read call/write call writes the byte into storage. The 'op-end' trigger is turned on and the operation ends.

FEMD 5-050 contains the circuit description.

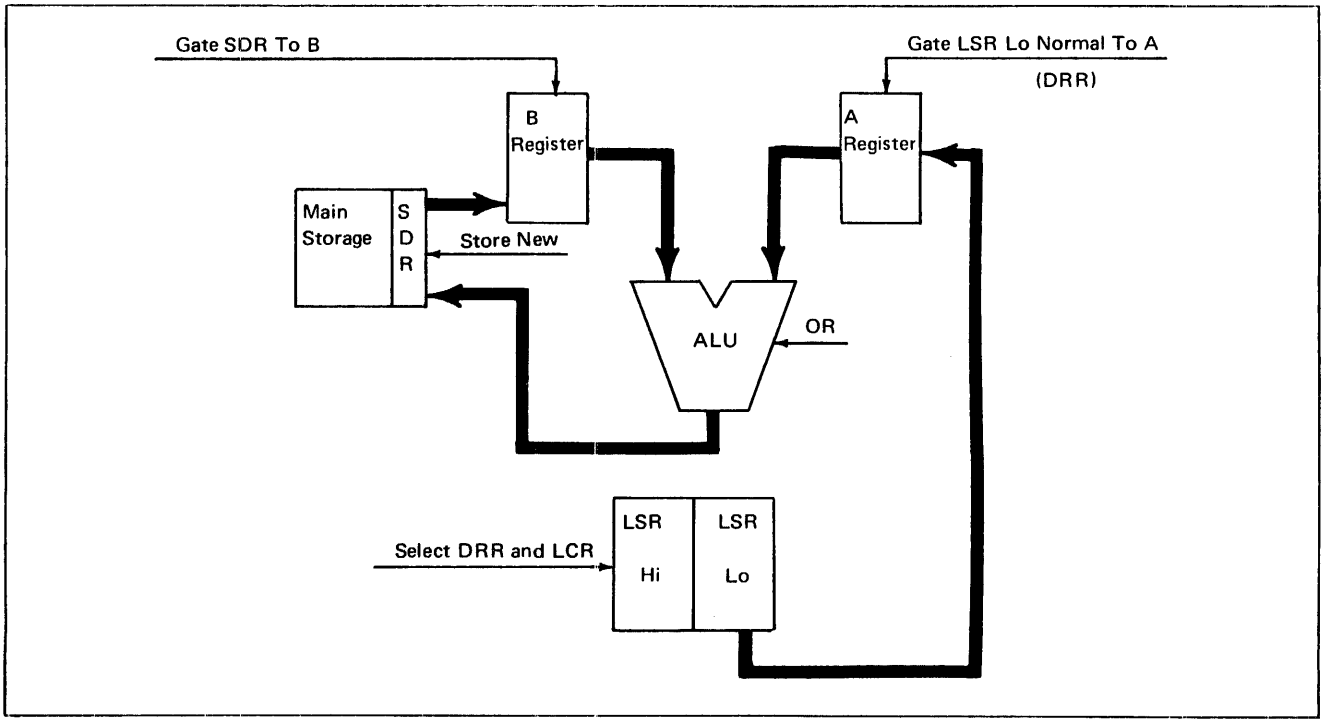


Figure 3-37. Set Bits On Masked Data Flow

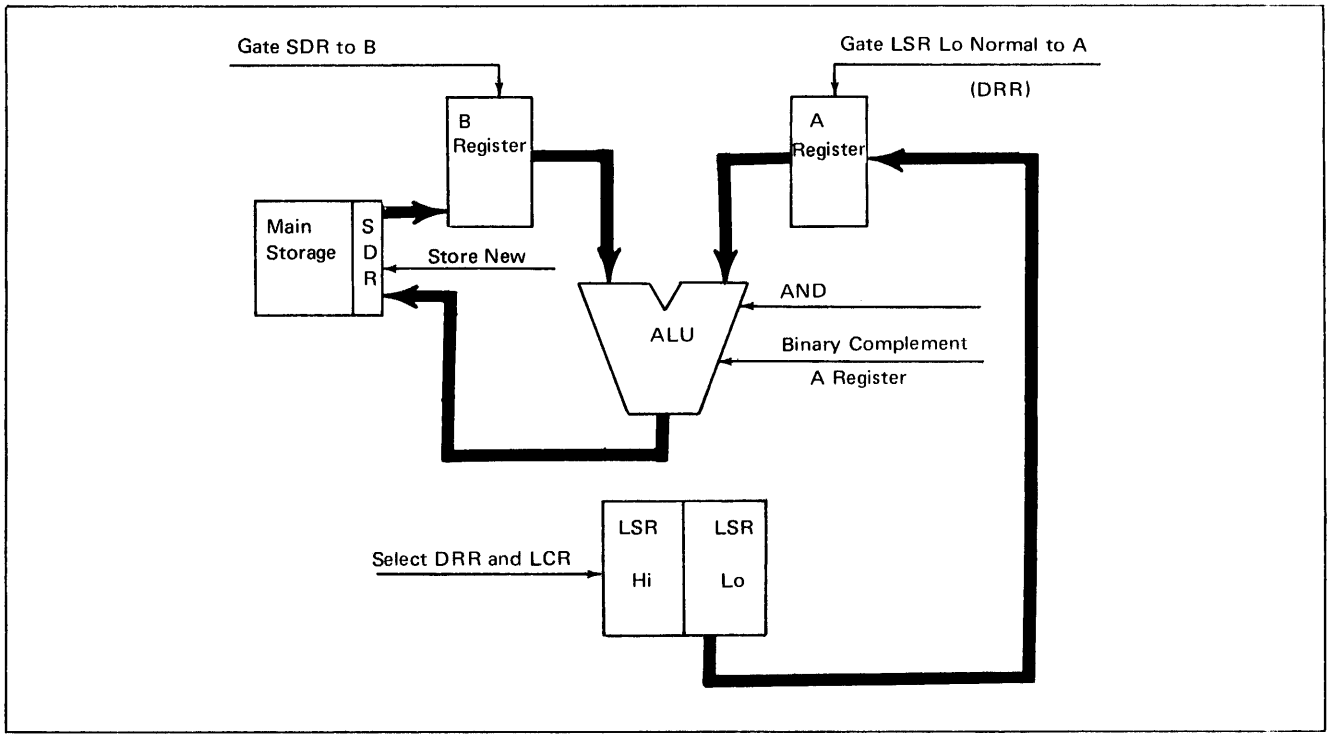


Figure 3-39. Set Bits Off Masked Data Flow

Test Bits On Masked—TBN

- Activate 'CR test false' if bits present in the Q code are not all present in the B address storage location.

The test bits on masked operation tests to determine if all bits present in the Q code are also present in the B address storage location. If they are not, the 'CR test false' latch is turned on.

The operation requires a single B-cycle and uses the 'OR' control line in the ALU (Figure 3-40). The Q code is transferred from the DRR to the A register and the B-field byte is loaded into the B register. Any bit in the A register that is not present in the B register gives a 'test false' output (Figure 3-40). The results are not written into storage but are used merely to set the condition register.

FEMD 5-050 contains the circuit description.

Test Bits Off Masked—TBF

- Activate 'CR test false' latch if any bits present in the Q code are also present in the B address storage location.

The test bits off masked operation tests to determine if all bits present in the Q code are absent from the B address storage location. If they are not, the 'CR test false' latch is turned on.

The operation requires a single B cycle. The Q code is transferred from the DRR to the A register (Figure 3-41). The A register is binary complemented and the AND control line in the ALU is used to give a 'test false' output for any bit in the Q code which has a corresponding bit in the B address byte. The results are not written into storage but are used merely to set the condition register.

FEMD 5-050 contains the circuit description.

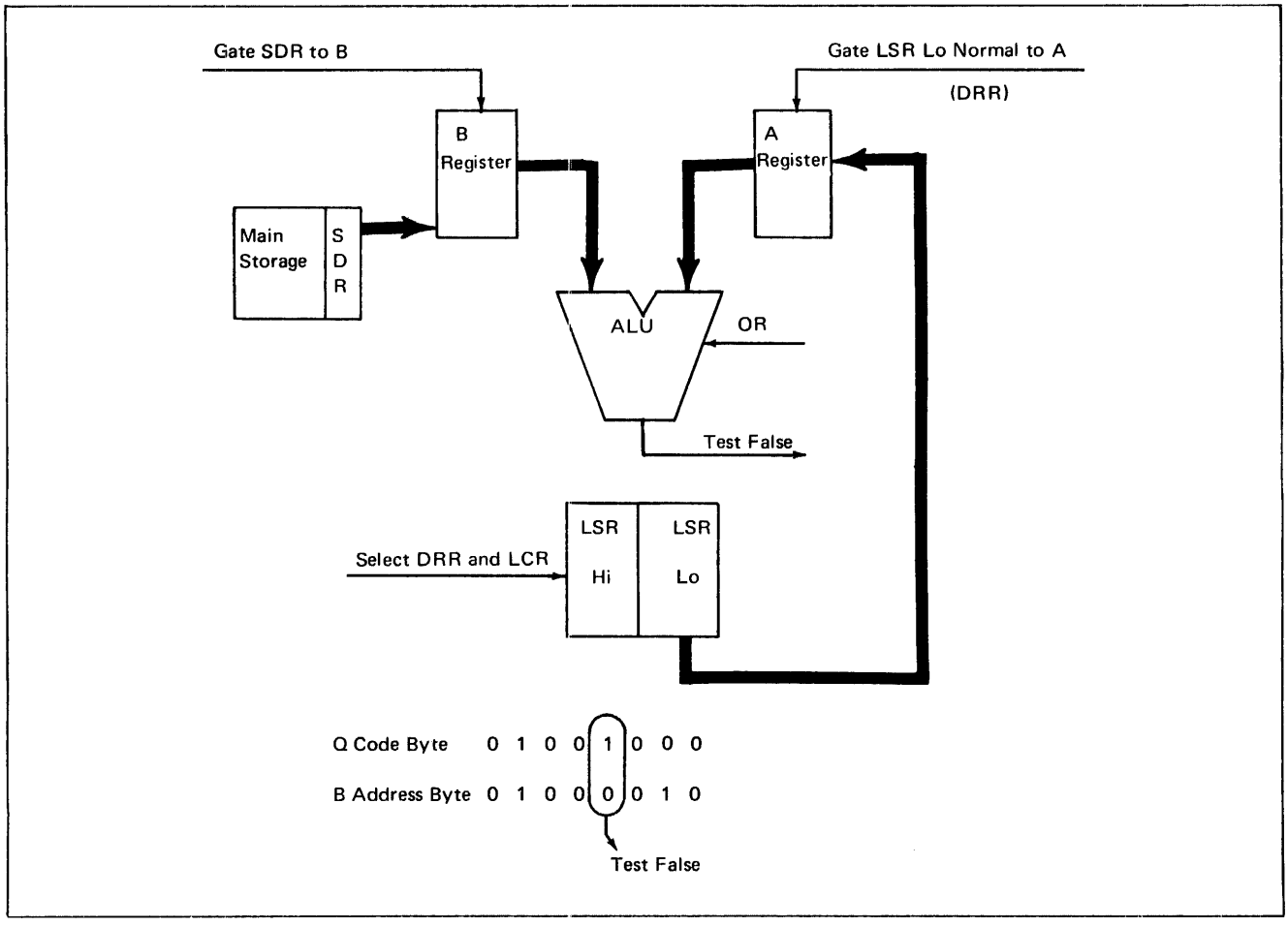


Figure 3-40. Test Bits On Masked

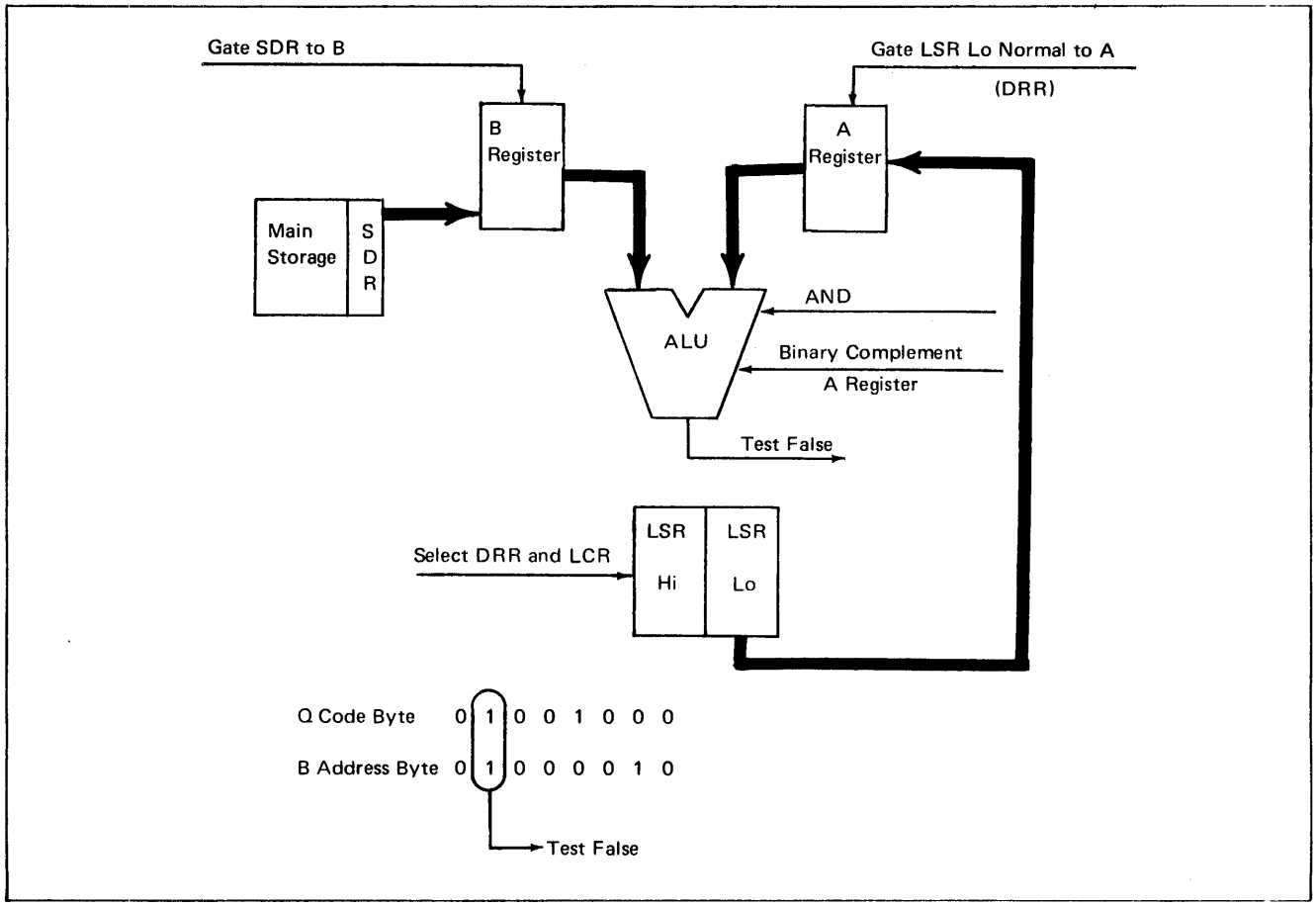


Figure 3-41. Test Bits Off Masked

Store Register—ST

- Place the contents of the register selected by the Q code into the B field storage location.

  The store register instruction stores an LSR in the B field storage locations. LSR selection is divided into two different groups depending upon the presence or absence of Q bit 0 (Figure 3-42).

Q Code Bits	Register Selected With Q Bit 0	Register Selected With No Q Bit 0
1	Interrupt 1-IAR	P2-IAR
2	Interrupt 2-IAR	P1-IAR
3	Interrupt 3-IAR	IAR
4	Interrupt 4-IAR	ARR
5		PSR
6		XR2
7		XR1

Note: With Q bit 0 and no other Q bits, Interrupt 0-IAR is selected.

Figure 3-42. LSR Selection

Since the LSRs are two bytes long, the store register instruction requires two B-cycles. During the first B-cycle, the Q register selects the LSR and the low order position is transferred to the A register (Figure 3-43). The B register is left all zeros so the A register is binary added to zero to move the LSR byte through the ALU. 'Store new' enters the byte in the SDR and 'read call/write call' writes it into storage.

The BAR is decremented and in the second B cycle, the high order byte of the LSR is moved (Figure 3-43). The 'op-end' trigger is then turned on and the operation ends.

FEMD 5-060 contains the circuit description.

Load Register—L

- Place the contents of the B field into the register selected by the Q code.

The load register operation loads an LSR with the contents of the B field storage locations. LSR selection is the same as for a store register operation (Figure 3-42).

During the first B cycle, the Q register selects the LSR and the first B field byte is passed through the ALU without any ALU controls (Figure 3-44). The ALU output is then written into the low order position of the LSR.

The BAR is decremented and in the second B cycle, the next byte is written into the high order position of the selected LSR. The 'op-end' trigger is turned on to end the operation.

If the LSR selected by the Q code is the PSR (Q bit 5 and not bit 0), an additional function is performed. Since the PSR low order is used as the CRR, the CR is also set by the ALU output during the first B cycle. Figure 3-45 shows the CR positions set by the ALU output.

FEMD 5-060 contains the circuit description.

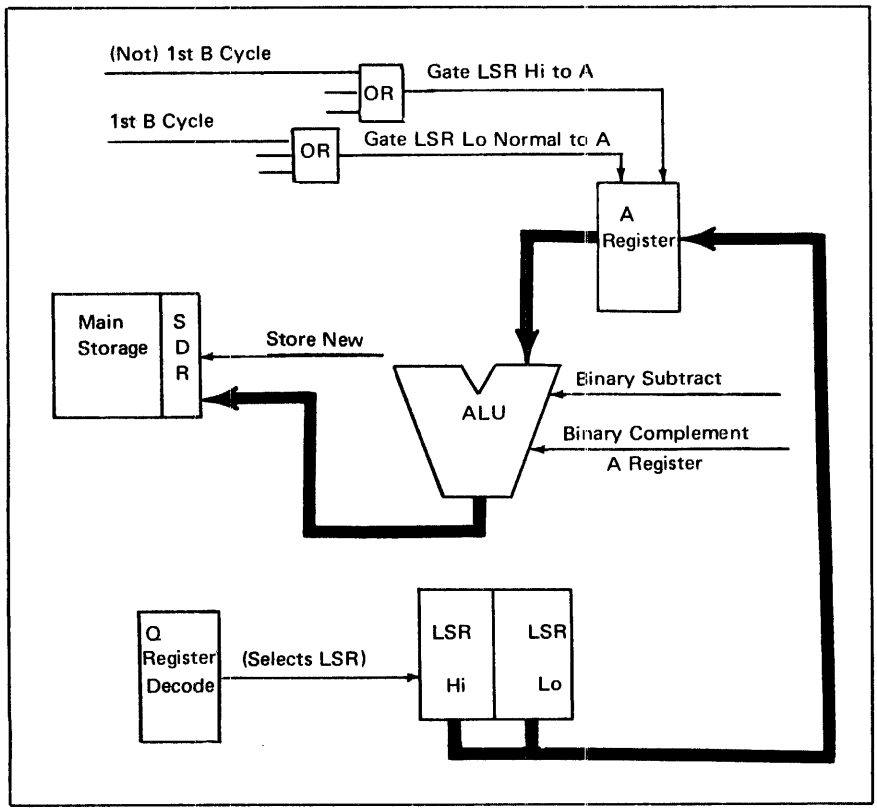


Figure 3-43. Store Register

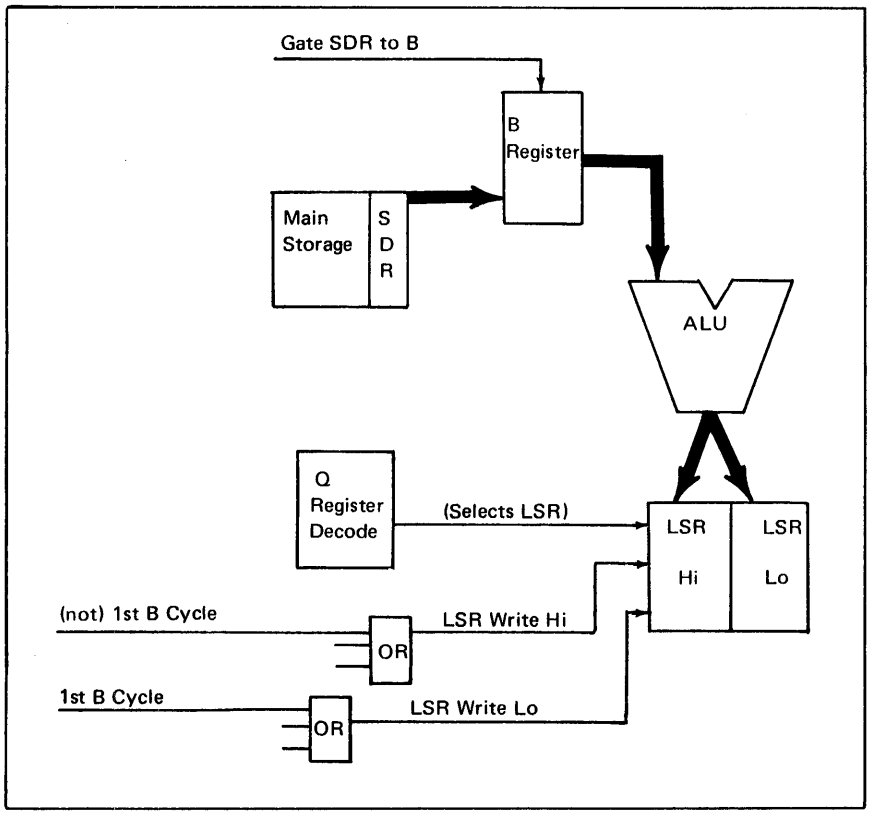


Figure 3-44. Load Register

ALU Output Bits	CR Results
7	Equal
6 not 7	Low
not 6 not 7	High
2	Binary Overflow
3	Test False
4	Decimal Overflow

Figure 3-45. Load PSR–CR Settings

Add to Register–A

- Add B field contents to register selected by the Q code.
- Put results in selected register.

The add to register operation adds the B field to an LSR and loads the result into the LSR. LSR selection is the same as for a store register operation (Figure 3-42).

During the first B cycle, the Q register selects the LSR and the low order position of the LSR is transferred to the A register (Figure 3-46). The first B field byte is loaded into the B register and binary added to the A register. The results are written into the low order position of the LSR.

The BAR is decremented and the process is repeated for the high order position of the LSR. The 'op-end' trigger is turned on to end the operation.

The results of the addition are also used to set the condition register. Figure 3-47 shows the significance of the CR settings.

FEMD 5-060 contains the circuit description.

Equal	Low	High	Binary Overflow
Result is zero	No Carry and non-zero result	Carry and non-zero result	Result too large for register (no high order carry)

Figure 3-47. CR Settings–Add to Register

Load Address–LA

- If instruction format is four bytes, load two byte address into index register selected by bits 6 and 7 of the Q code.
- If instruction format is three bytes, add last byte to index register selected by the op code and load the result into register selected by Q code.

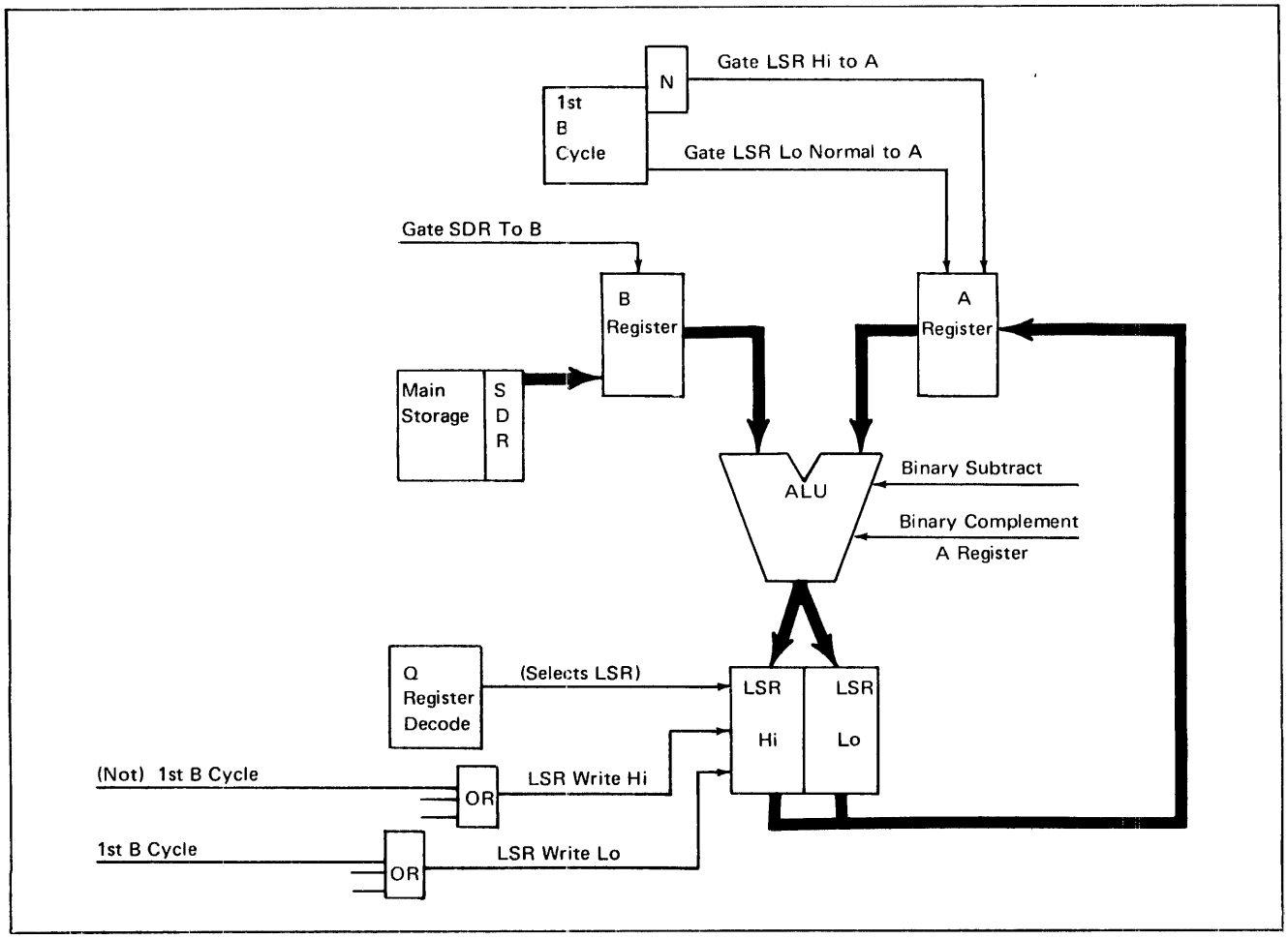


Figure 3-46. Add to Register

Q Code Bit	Register Selected
6	XR-2
7	XR-1

Figure 3-48. Load Address–Index Register Selection

The load address instruction performs one of two possible operations, depending on the instruction length. If the instruction is four bytes long (Figure 3-49), the last two bytes of the instruction are taken from storage and loaded into the index register selected by bits 6 and 7 of the Q code (Figure 3-48). If the instruction is three bytes long (Figure 3-50), the last byte of the instruction is taken from storage, added to the contents of the index register selected by bits 0-3 of the op code, and then loaded into the index register selected by bits 6 and 7 of the Q code.

A four byte format requires one I-H1 cycle and one I-L1 cycle. During the I-H1 cycle bits 6 and 7 of the Q code

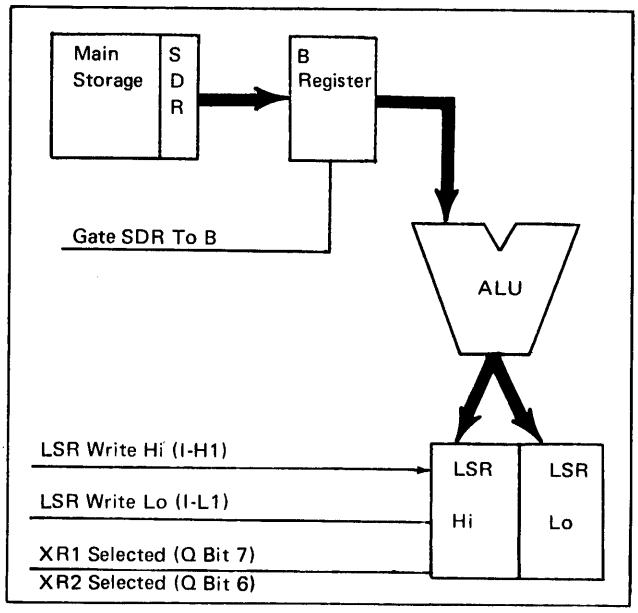


Figure 3-49. Load Address Data Flow–Not Indexed

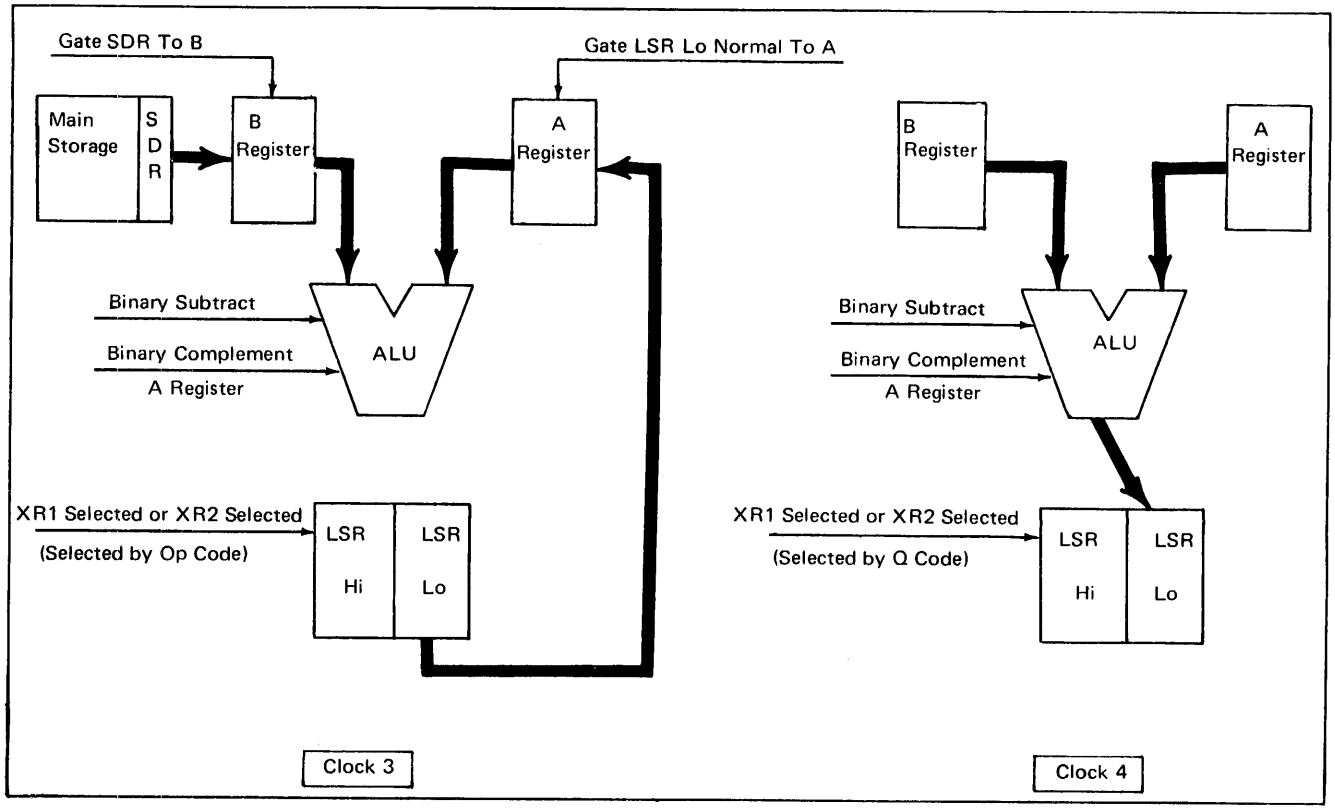


Figure 3-50. Load Address Data Flow–Indexed

select one of the two index registers. Bit 7 only being on selects XR1, bit 6 only being on selects XR2. Data is transferred from the storage position addressed by the IAR to the B register, through the ALU, and into the high order position of the selected index register. The IAR is incremented and during the I-L1 cycle the process is repeated for the low order position of the index register. A three byte format requires going through only one IX cycle. Data is transferred from the storage location address by the IAR to the B register and at clock 3 time is added to the contents of the selected index register (bits 0 through 3 of the op code). Bits 6 or 7 of the Q code selects one of the index registers at clock 4 time and the sum in the ALU is loaded into that register. The IAR is incremented for the next operation.

FEMD 5-109 contains the circuit description.

Branch On Condition—BC

- Condition register is tested for the condition specified in Q code.
- Branch to address is placed in ARR.
- ARR/IAR interchange if tested condition is satisfied.

The branch on condition operation loads the two byte branch to address into the ARR. If the condition specified in bits 2 through 7 of the Q code is satisfied (Figure 3-51), an IAR/ARR interchange occurs at op end. The ARR is then used as the IAR.

Bit 0 of the Q code is used to specify if the branch is to be performed on condition true or condition false. If bit 0 is on and at least one of the conditions specified by the Q code is present, the branch is performed. If bit 0 is off and all conditions specified by the Q code are missing, the branch is performed.

Q Bit	Condition Tested
0	Presence of Condition
Not 0	Absence of Condition
7	Equal
6	Low
5	High
4	Decimal Overflow
3	Test False
2	Binary Overflow

Figure 3-51. Branch On Condition—Q Code

During the I-Q cycle, the Q code data is transferred from storage, through the B register, and into the ALU. The contents of the condition register is decoded and enters the ALU through the A register. An ALU AND function is performed (both input bits must be the same to get an output), and the output is checked for non-zero. The result is placed in the Q register. B register bit 0 is used to determine if an ALU sum of zero or not-zero is needed to satisfy the branch condition. Figure 3-52 shows the function of each Q code bit when testing the condition register. If the branch condition is satisfied, the IAR/ARR interchange latch is set (Figure 3-53). The LSR switching does not occur, however, until op end.

During the I-H1 cycle, the ARR is selected and the high order position of the branch to address is transferred from storage through the B register, ALU and into ARR high. The IAR is incremented and the process is repeated for the low order position.

FEMD 5-130 contains the circuit description.

COMMAND INSTRUCTIONS

- Load operation code into op register.
- Q code used to define command.
- Control code is third byte of instruction and contains additional information pertaining to the command.

I-cycles for command operations are three cycles in length; first, an I-op cycle transfers the operation code from main storage to the op register. Second, an I-Q cycle transfers the Q code into the Q register and DRR. If the operation is a branch or jump, the condition register is also tested for true or false. Third, an I-R cycle is then used to transfer from storage the control code needed to execute the command. The details for use of the control code are covered under specific operation descriptions.

Jump On Condition—JC

- Condition register is tested for the condition specified in the Q code.
- If the tested condition is satisfied, control code is added to IAR for next sequential instruction.

The jump on condition operation is similar to branch on condition except for the instruction address modification. If the condition register contents satisfy the condition specified in the Q code (Figure 3-51), the control code byte is added to the IAR.

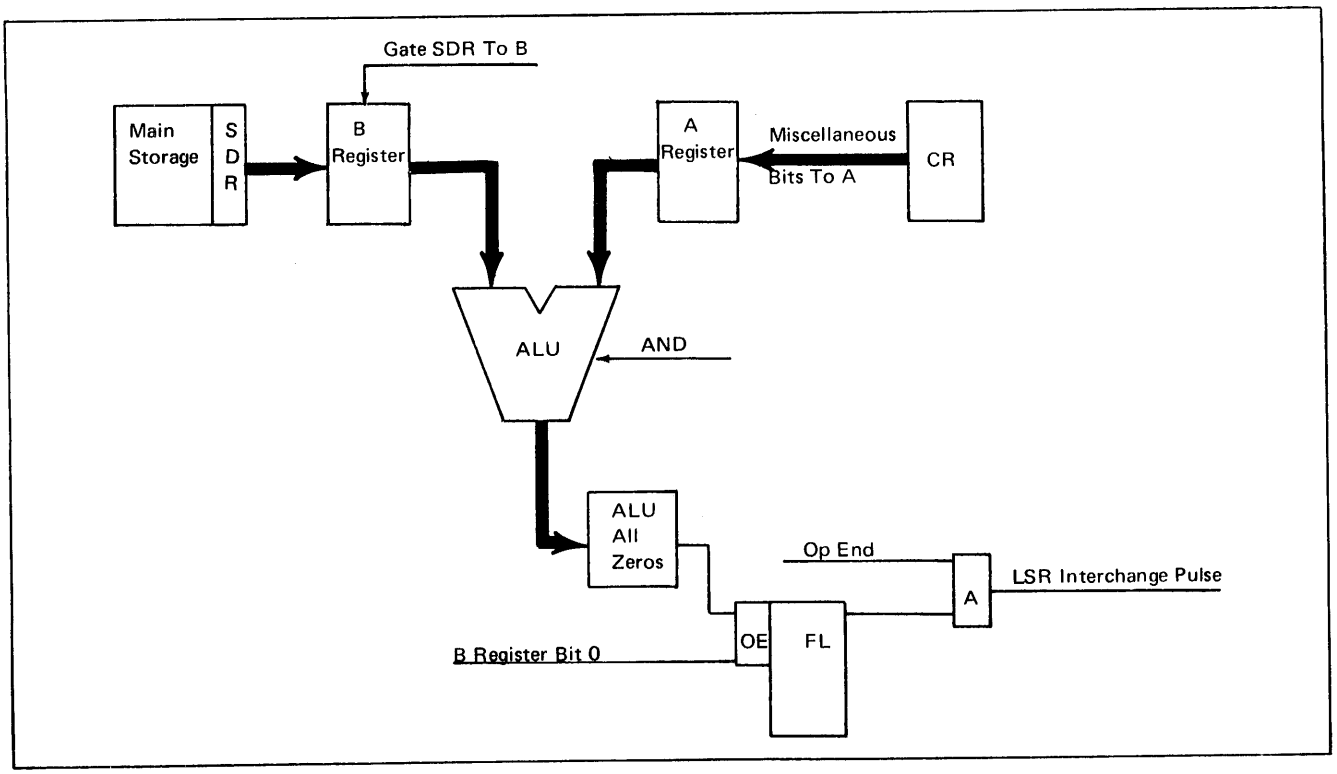


Figure 3-52. Branch On Condition–I-Q Data Flow

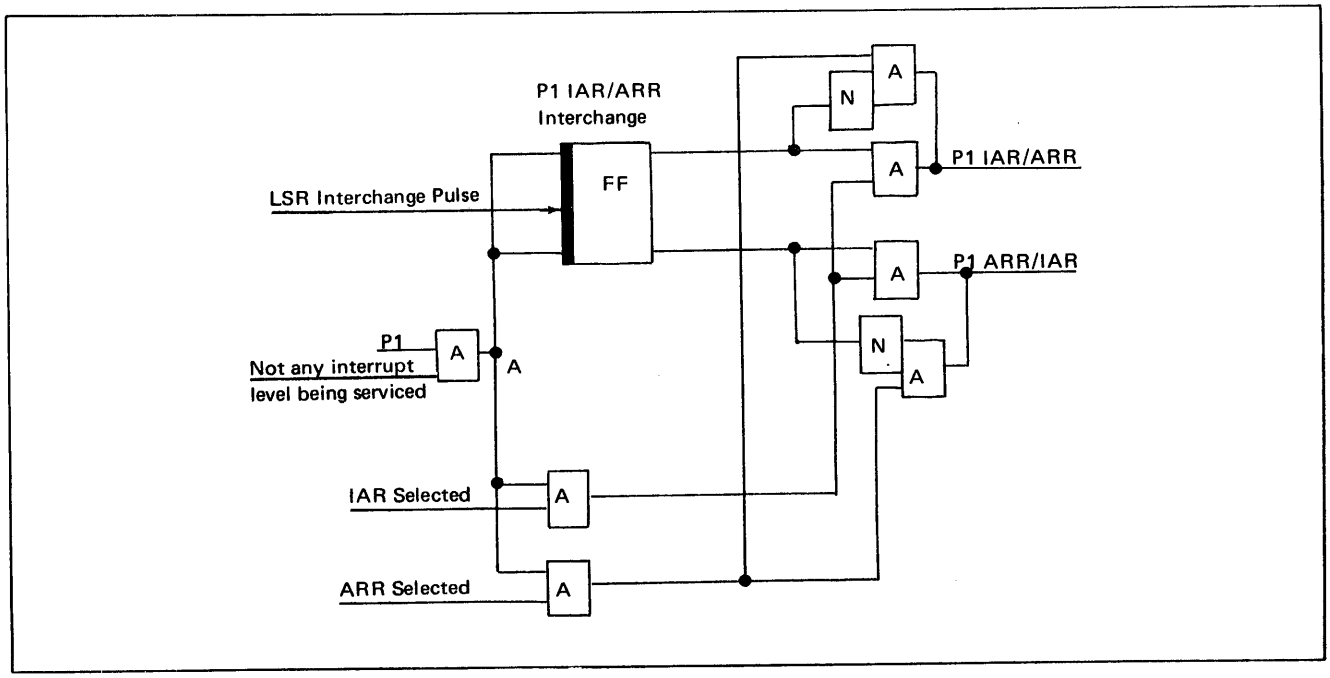


Figure 3-53. IAR/ARR Selection Control

During the I-Q cycle, the Q code is transferred from storage, through the B register and into the ALU. The condition register is decoded into the A register and enters the ALU. An ALU AND function is performed (both input bits must be the same to get an output) and the output is checked for non-zero.

Bit 0 of the Q register is used to specify if the jump is to be performed on condition true or condition false. If bit 0 is on and any one of the conditions specified by the Q code is present, the jump is performed. If bit 0 is off and all conditions specified by the Q code are missing, the jump is performed.

During the IR cycle, if the jump condition was satisfied, the IAR is selected and the control code is added to that register.

FEMD 5-140 contains the circuit description.

Halt Program Level (Basic Machine)

- Prevents execution of the next sequential instruction.
- Loops on halt instruction until the start key is pressed.
- Instruction format bytes two and three displayed on the console.

The halt program level instruction prevents execution of the next sequential instruction. During the I-Q cycle, the tens position of the halt identifier (instruction byte two) is displayed in the console display (Figure 6-7) and program interlock is forced. During the I-R cycle the units position of the halt identifier (instruction byte three) is displayed. I-R cycle and program interlock activates I-R program back-up. Program back-up decrements the IAR by two ('force bit 6 to A') and this loop continues until the system start key is pressed. Pressing the start key eliminates the program interlock and the next sequential instruction is executed.

If a halt is executed during an interrupt level, program interlock is blocked and the IAR is advanced in the normal manner.

FEMD 5-210 contains the circuit description.

Halt Program Level (Dual Programming Feature)

- Prevents execution of the next sequential instruction.
- Loops on halt instruction until halt reset key is pressed.
- Instruction format bytes two and three displayed on console.
- IAR is modified to the halt command starting address.
- Next instruction is taken from the alternate program IAR.

The halt program level instruction prevents execution of the next sequential instruction and the halt identifiers (instruction bytes two and three) are displayed in the console display (Figure 6-7). During the I-Q cycle, program interlock is forced and during the I-R cycle, I-R program back-up is activated. Program back-up decrements the IAR by two ('force bit 6 to A') and 'program 2' is activated. The next instruction address is selected from the program 2 IAR. Execution of the original program is resumed if; (1) the halt reset key for that program level is pressed, or (2) a halt program level instruction occurs while in program level 2. The program will return to the starting address of the original halt program level instruction.

If a halt program level instruction is executed during an interrupt level program, program interlock is blocked and the IAR advances to the next sequential instruction.

FEMD 5-210 contains the circuit description.

I/O INSTRUCTIONS

I/O instructions consist of two types of instruction, one address instructions and command instructions. The Q code contains the address of the I/O device, the code for the primary or secondary unit involved, and the function to be performed (read, write, control). Where applicable, the control code contains additional information for the device (space, stacker selection, etc.).

I-cycles follow the same cycle pattern as in other 1 address or command instructions. The I-cycle link with the I/O attachments is covered under the individual operations.

Start I/O—SIO

- Start an I/O device.
- Q code contains device address and function to be performed (read, punch, print).
- Control code contains additional instructions for device (space, stacker selections, etc.).

The start I/O instruction starts the mechanical function of any I/O device. The particular device selected and the

function to be performed are determined by the Q code of the instruction. The device address (DA) is contained in bits 0-3 of the Q code (Figure 3-54). Bit 4 of the Q code contains the modifier (M) bit which selects the primary or secondary unit of the device. Bits 5-7 contains the N field which is the function to be performed by the device. Three N codes are common to all devices, but the remainder are assigned by the individual devices (Figure 3-55).

The control code byte of the instruction further defines the device function. For instance, it may define the stacker pocket selected in the MFCU or it may define the type of spacing for the line printer. Use of the control code varies with the individual devices. Refer to the theory of operations manual for the various I/O attachments.

CPU control of the operation ends with the I-R cycle of the instruction. If the addressed device is not busy or does not need attention and the Q-byte and control byte reach the device without error the CPU continues with the next sequential instruction. If the device cannot execute the command, the program loops on the SIO instruction until the device is no longer busy or until the operator has corrected the attention condition.

Once the device has accepted the instruction, the operation is performed by the attachment circuitry. Whenever the device needs data from storage (write, print, punch) or has data to send to storage (read) the attachment breaks into the program to use the number of cycles it requires.

FEMD 5-150 contains the circuit description.

Device Address		M Bit	
Bits 0123	Device	Bit 4	Unit
0000	CPU Console		
0001	Keyboard		
0011	Spare		
0111	Spare		
1010	Disk - Primary Spindle	*	Primary/Secondary Drive
1011	Disk - Secondary Spindle	*	Primary/Secondary Drive
1110	Line Printer	*	Primary/Secondary Carriage
1111	MFCU	*	Primary/Secondary Feed

* 0 - Primary Unit
  1 - Secondary Unit

Figure 3-54. DA and M-Bit Assignment

N Field	
Bits 567	Function
000	Control or Equivalent
001	Read or Equivalent
010	Write or Equivalent
*	

* Additional codes determined by the individual devices.

Figure 3-55. N Code Assignment

I-Cycle

At clock 5 time of the I-Q cycle, when the Q code is latched in the ALU latches, the ALU output is sent to the devices on the DBO. This Q byte bypasses the DBO translator.

Each device attempts to decode the address. If any device recognizes the address and if the Q byte contains a valid N code, the device activates either I/O condition A or I/O condition B (Figure 3-56).

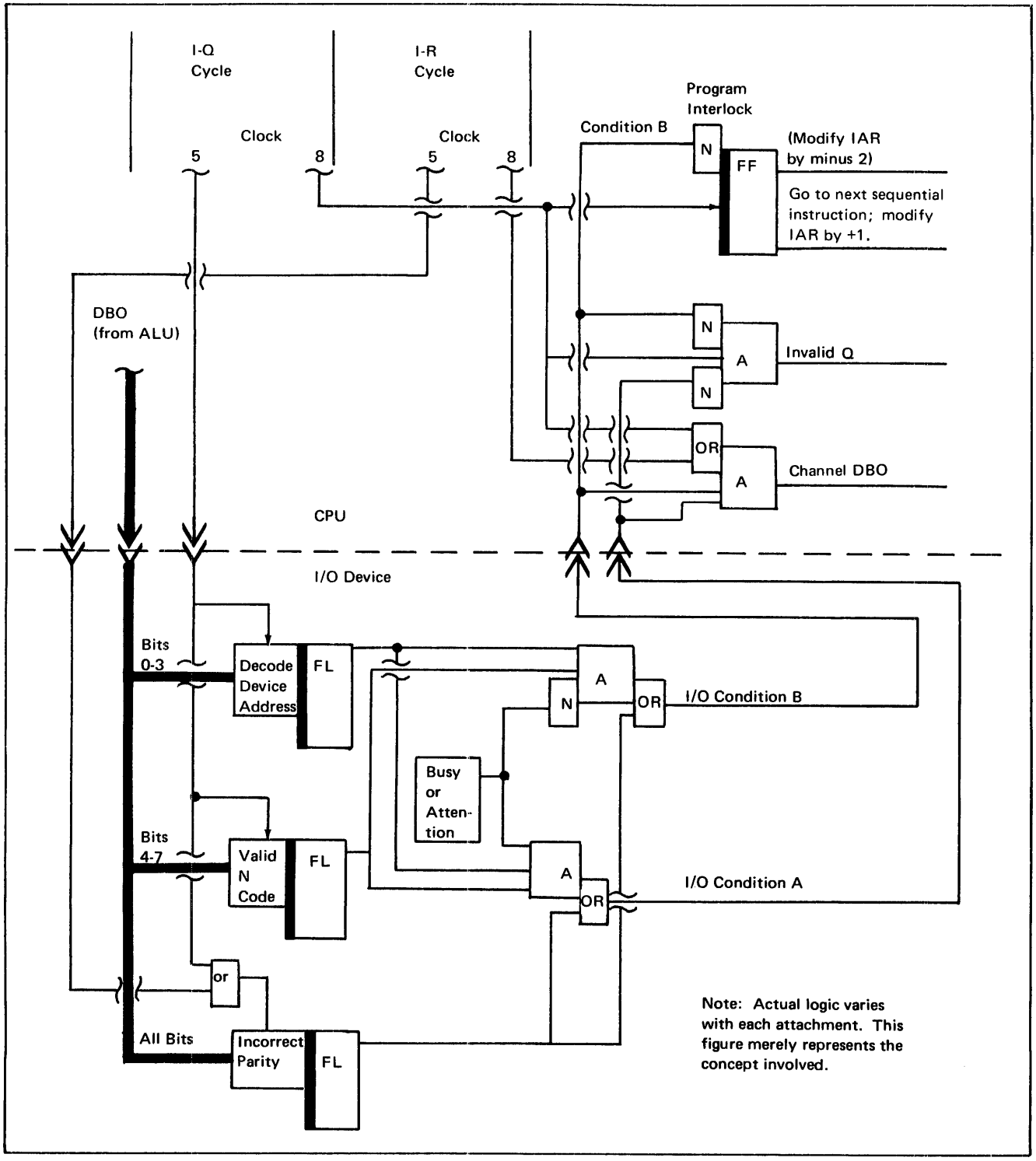


Figure 3-56. SIO–CPU/Device Control

If the device is 'busy' or 'not ready and no errors', 'I/O condition A' only is activated and at clock 8 time of the I-Q cycle, the CPU activates 'program interlock'. If the device is able to execute the instruction, I/O condition B only is activated blocking the gate to the 'program interlock' trigger. If device is 'not ready with errors', I/O condition B is activated and device will no-op instruction.

IAR modification is dependent upon the 'program interlock' trigger. During I-R cycle clock 5 and 6 time, if program interlock is inactive, 1 is added to the IAR to increment it in the normal manner (Figure 3-57). If, however, 'program interlock' is active, the IAR is decremented by 2 to retry the instruction. During clock 7 and 8 the ALU controls remain the same as clock 5 and 6 to modify the high order position of the IAR.

During the I-Q and I-R cycles, the DBO is also parity checked (Figure 3-57). If a parity check occurs the processor is signaled by activating both I/O condition A and I/O condition B. Figure 3-58 shows the significance of all settings for the I/O condition A and I/O condition B lines.

Line Activated By Any Device	Significance
'I/O condition B' only	Correct address, valid N code, device not busy and does not need attention—instruction accepted.
'I/O condition A' only	Correct address, valid N code, device busy or needs attention—instruction rejected.
Both lines	Incorrect parity—causes processor check and DBO parity check.
Neither line	Invalid address or N code--causes processor check and invalid device address.

Figure 3-58. SIO–Device Response

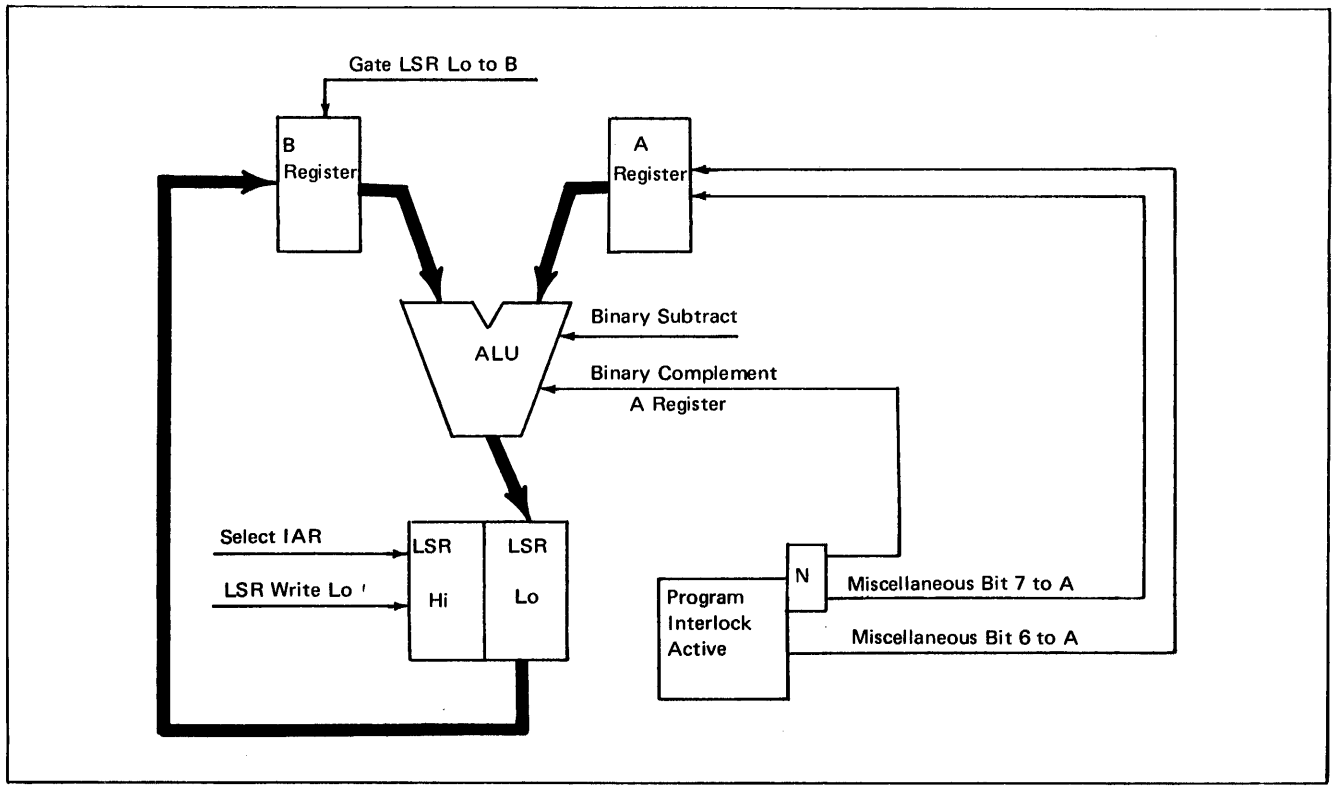


Figure 3-57. SIO–I-R Cycle; Modify IAR Low

Cycle Steal Priority (CSP)

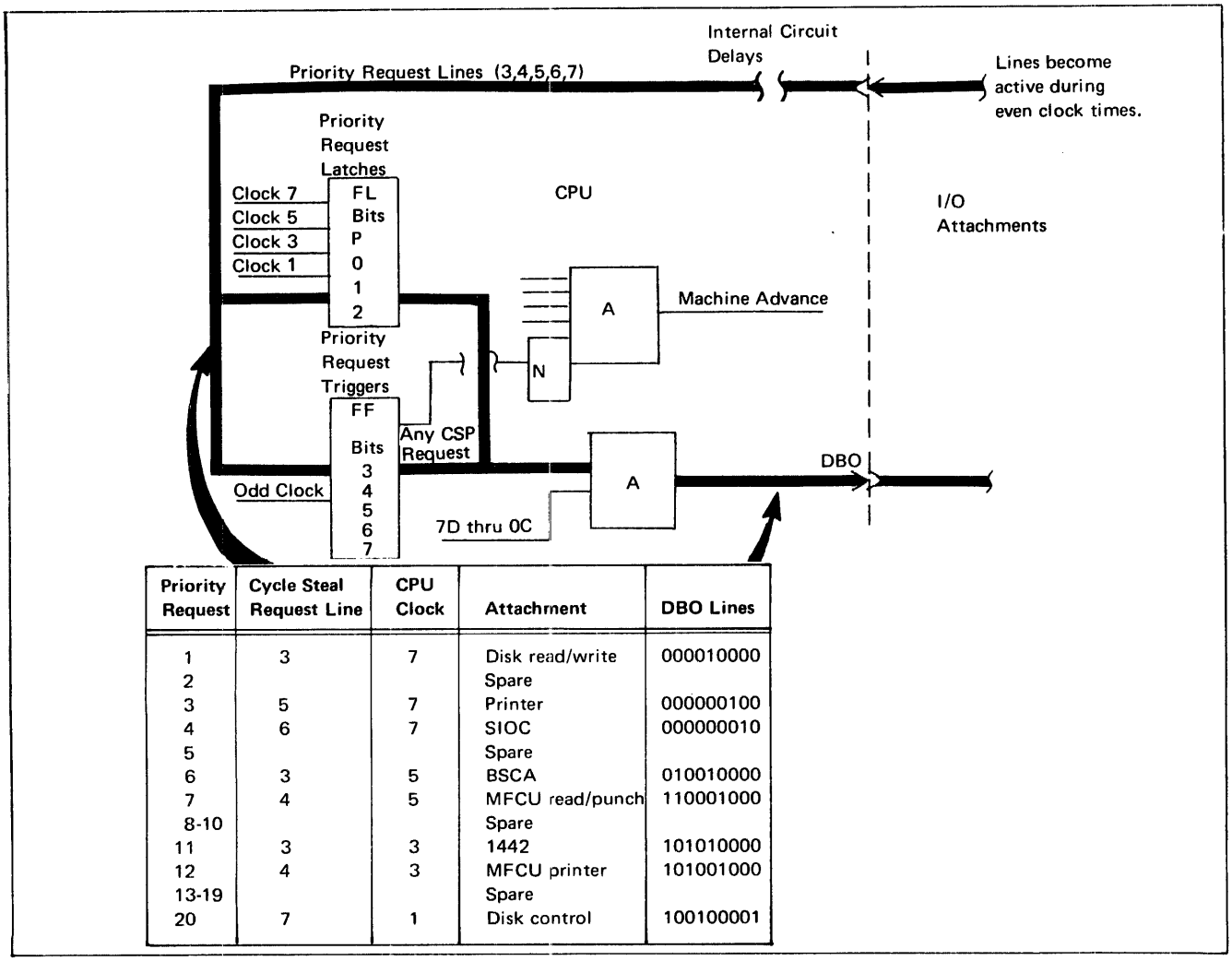
- Device requests cycle.
- CPU assigns cycle by device priority.
- I/O cycle can occur between any two CPU cycles.

Whenever an I/O device reaches a point in its mechanical operation where it needs data from storage (write, print, punch) or has data to send to storage (read) the device requests an I/O cycle. An I/O cycle request can occur during any cycle and is always granted by the CPU. More than one I/O device may request an I/O cycle at the same time so each device is assigned a particular cycle steal priority.

Cycle steal requests are generated by the attachments at even clock times. Because of the internal circuit delays, these lines are not sampled until the next clock pulse.

During the CPU odd clock times, requests for cycle steals enter the CPU from the attachments on the 'priority request' lines (Figure 3-59). These requests are entered into the 'priority request' latches and triggers. If more than one device requests an I/O cycle during the same clock time, the bit triggers with the highest priority prevent the lower priority triggers from being turned on. A request at a later clock time resets the triggers and latches for any previous request.

At clock 7D time, the bit structure for the highest priority device among those requesting a cycle is sent to the devices on the DBO, bypassing the DBO translator (Figure 3-59). 'Any CSP request' blocks the 'machine advance' pulse preventing the CPU from advancing to the next CPU cycle.



Priority Request	Cycle Steal Request Line	CPU Clock	Attachment	DBO Lines
1	3	7	Disk read/write	000010000
2			Spare	
3	5	7	Printer	000000100
4	6	7	SIOC	000000010
5			Spare	
6	3	5	BSCA	010010000
7	4	5	MFCU read/punch	110001000
8-10			Spare	
11	3	3	1442	101010000
12	4	3	MFCU printer	101001000
13-19			Spare	
20	7	1	Disk control	100100001

Figure 3-59. I/O Cycle Request–Priority Assignment

Odd parity is maintained for the priority bits on DBO. A P-bit is available on DBO when the parity latch is off. Thus, if no request is received a P-bit is available on DBO at clock 7D time. For clocks 1, 3, and 5, each device turns on 2 bit positions to be sent on DBO (Figure 3-59). The parity latch is off and thus provides the needed P-bit. During clock 7, only one bit position is turned on by the requesting device, and requires the P-bit latch be turned on to eliminate unneeded parity bit.

I/O Cycle

- Device controls data flow and functional unit control lines within CPU.

I/O cycles follow the same general data manipulation procedure as CPU cycles. That is:

- Clock 0 - address storage
- Clocks 1 and 2 - miscellaneous (generally LSR alteration)
- Clocks 3 and 4 - compute (data manipulation between CPU and I/O device)
- Clocks 5 through 8 - address register modification.

If an I/O cycle does not require all of these functions, the I/O device blocks them by controlling the CPU data flow control lines. The following examples represent a method of transferring data between the CPU and I/O devices. The actual method used depends upon the result desired by the individual attachments.

A device is given the cycle steal assignment at clock 7D prior to the actual I/O cycle. In order to have SAR loaded at clock 0 of the I/O cycle, the assigned device must raise the proper LSR select lines at clock 8 prior to the I/O cycle. The LSR select buffer will select the LSR during clock 0. The device can select LSRs at four times during the I/O cycle. It may select the same LSR, a different LSR, or no LSR (Figure 3-60).

The A register is loaded at odd CD clocks with the information that the device puts on DBI at even clocks. This data may be translated from card code to hexadecimal during clocks 2 and 3.

The B register is loaded at odd CD clocks with the following data:

Clock	B Register Data
1CD	00
3CD	Contents of SDR or 00 Controlled by attachment
5CD	Selected LSR low byte or 00 if no LSR is selected
7CD	Selected LSR high byte or 00 if no LSR is selected

ALU output during the I/O cycle is the contents of the B register minus the contents of the A register or the contents of the B register plus the contents of the A register. This is determined by the device through 'chan bin sub'. ALU output is the following data at the following clock times:

Clocks	ALU Out Data
2D to 4C	± DBI
4D to 6C	B Register ± DBI
6D to 8C	LSR low ± DBI
8D to 2C of next cycle	LSR high ± DBI

DBO equals the ALU outputs as latched except during clocks 7D to 0C. This allows the device to use ALU out data.

If the device needs data from storage (MFCU punch) the device does not block the 'gate SDR to B' line and the data is transferred from storage to the B register (Figure 3-60). No data is entered into the A register from DBI so the B register is binary added to the blank A register to effectively move the B register through the ALU. The data is latched into the ALU latches and is available to the I/O device on DBO. The device activates the 'chan in transl out' line if the byte needs to be translated to card code as in an MFCU punch operation (Figure 3-60).

Not all I/O cycles move the data unchanged through the ALU. The following example shows the function of each cycle taken by the line printer to print a character. The objectives of the three cycles in the example are:

1. Remove the data byte to be printed from storage and retain it.
2. Remove the chain image character from storage and compare it with the data byte to see if the chain is in the correct location to print the character.

3. Place the value 4/0 into the data byte location as an indication that the character has been printed.

In the first print cycle steal, the LPDAR (line printer data address register) addresses storage. The byte of data is added to zeros in the A register to move the byte through the ALU and is sent to the printer attachment where it is retained.

In the second cycle, the LPIAR (line printer image address register) addresses storage and the chain image character is read out and placed in the B register. The byte of data transferred to the attachment during cycle 1 is sent back to the CPU from the printer attachment and enters the A register from DBI. The printer attachment activates the 'chan in bin sub' line to subtract the A register from the B register. The result is sent to the printer attachment. If the result is zero, the two characters are the same and the printer prints the character.

In the third print cycle, the LPDAR again addresses storage and the printer attachment activates the 'I/O block SDR' line to prevent the character from entering the B register.

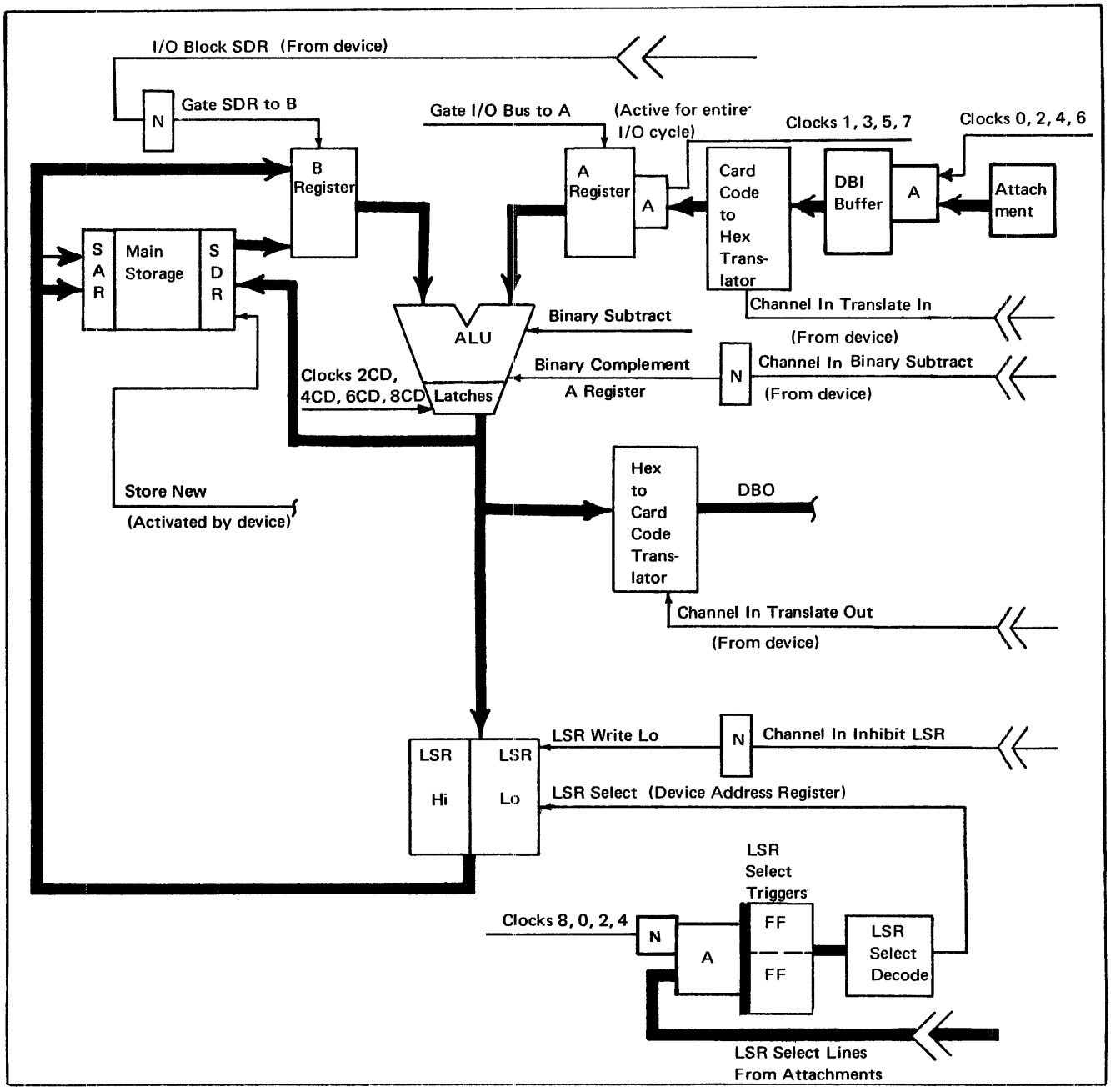


Figure 3-60. I/O Cycle Data Transfer (Part 1 of 2)

The printer attachment sends the hex value 4/0 to the CPU on DBI. This value is added to the zeros in the B register to move the 4/0 through the ALU. The printer attachment activates store new to enter the 4/0 into storage.

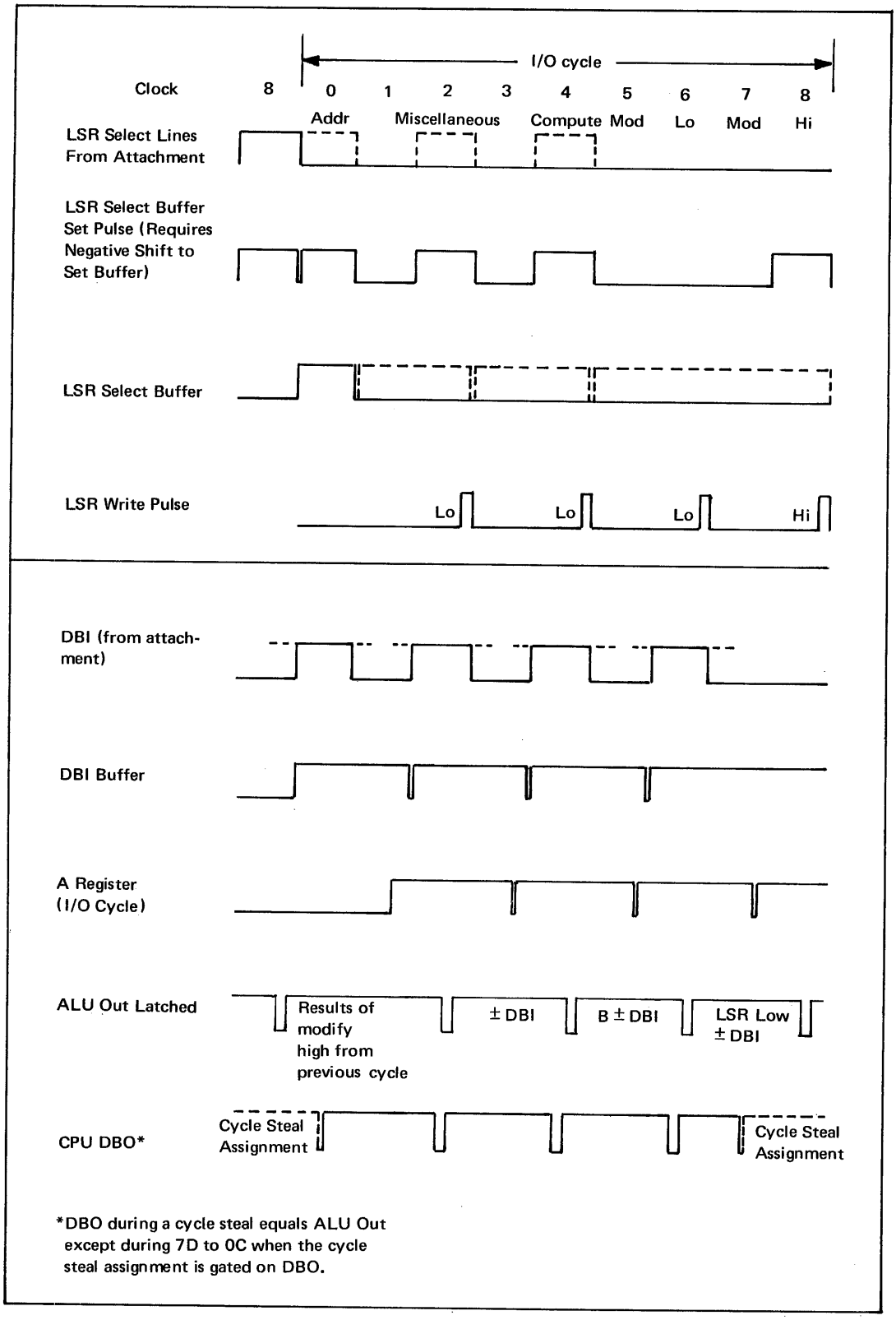


Figure 3-60. I/O Cycle Data Transfer (Part 2 of 2)

Address modification is the same as in a CPU cycle except that register selection, ALU controls, and the modification amount are all controlled by the I/O device. During clock 5 and 6, the LSR is selected in the same manner as during clock 0 (Figure 3-61). The amount the register is to be increased or decreased by is entered into the A register from DBI. Incrementing or decrementing is determined by controlling the 'bin comp A reg' line (Figure 3-62).

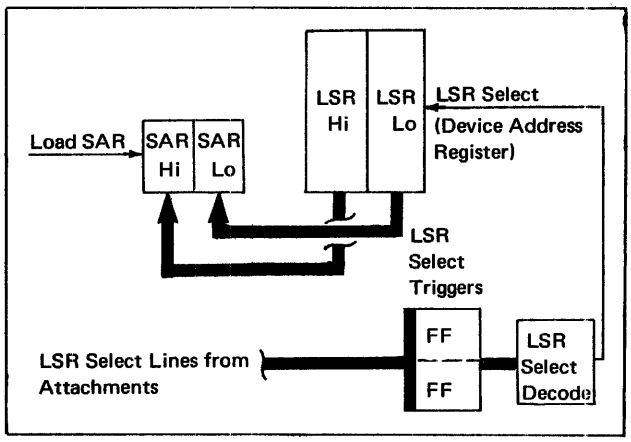


Figure 3-61. I/O Cycle–Storage Addressing

Address modification does not take place in all I/O cycles. For instance, during the third print cycle taken to print a character the printer attachment must address the same storage location as during the first print cycle. In all three print cycles, the printer attachment sends the value 12 to the CPU on DBI. This 12 is added to the LPDAR in all three cycles. However, in the first two cycles, incrementing is blocked by activating the 'chan in inh LSR' line to prevent the results from being written into the LPDAR.

Interrupt

- Interrupts main program with separate program.
- Interrupts occur only between instructions.

Some I/O attachments operate by means of an interrupt routine. An interrupt differs from a cycle steal by interrupting the main program with a separate program routine. For this reason, an interrupt can occur only at the completion of an instruction.

Each interrupt level has a separate IAR and ARR in the CPU so the IAR and ARR for the main program are not

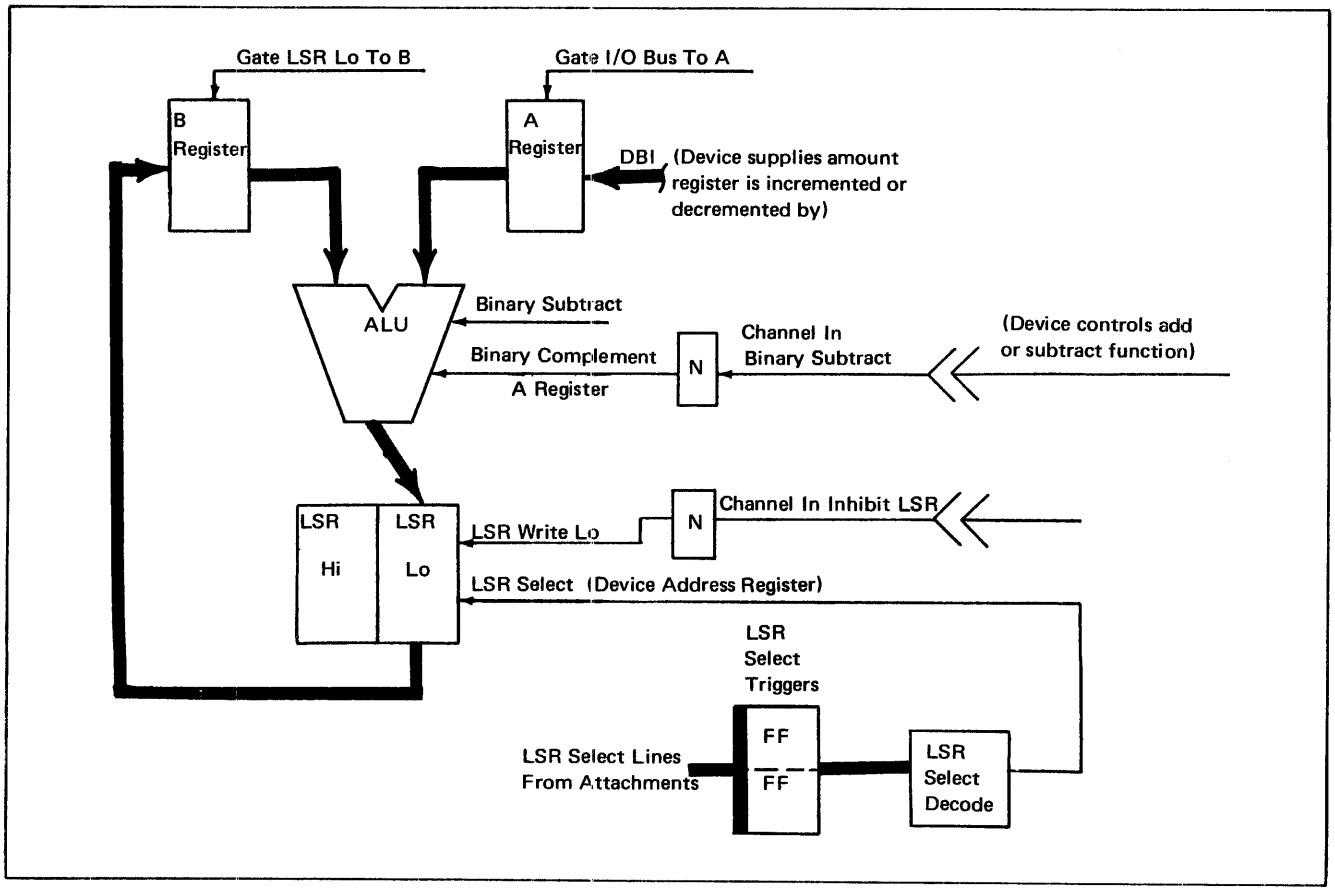


Figure 3-62. I/O Cycle–Address Register Modification

disturbed. Any other registers (CR or index registers) used during the interrupt must be stored at the beginning and re-established at the end of each interrupt routine.

The interrupt routine being performed is established by the interrupt priority latches. As in cycle steal, the highest interrupt level device takes precedence over lower level devices. Thus, it is possible for an interrupt routine to interrupt a routine of a lower priority device. However, each device maintains its interrupt request until it is satisfied, so the lower priority device finishes its routine upon completion of the higher level routine.

The stored program controls the ability of a device to interrupt by enabling and disabling the device through SIO instructions. Once an interrupt has occurred, the interrupt routine is also ended by an SIO instruction.

During the I-Q cycle, device selection occurs in the same manner as any SIO instruction. Then at clock 5 of the I-R cycle, the control code is sent to the device attachment on DBO (Figure 3-63). The control code is decoded by the device attachment to turn on the 'interrupt enable' latch. This latch remains on until a disable control code is sent in another SIO instruction.

If the device has a need to interrupt, the 'interrupt request' latch is turned on. At the end of the operation being performed in the CPU interrupt poll is sent to the device. This activates the 'DBI bit' line to turn on the interrupt latch for

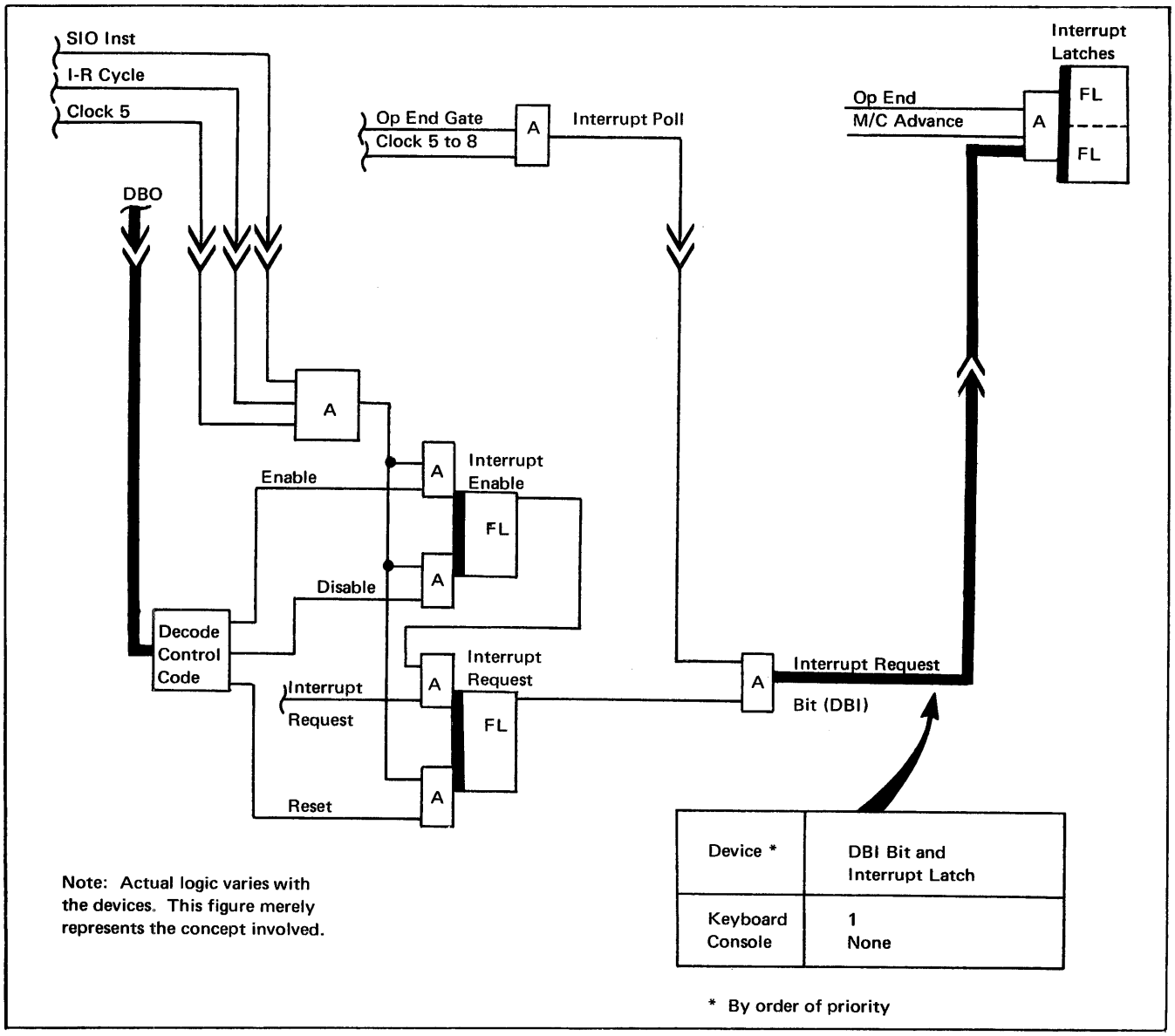


Figure 3-63. Interrupt Control

the device in the CPU (Figure 3-63). If more than one device is requesting an interrupt, only the highest priority interrupt latch will be turned on.

With any interrupt latch on, the selection of the normal IAR/ARR (P1 or P2) is blocked and the IAR/ARR for the active interrupt level latch is selected (Figure 3-64). The interrupt request latch in the device attachment stays on until an SIO with the proper control code resets it (Figure 3-63).

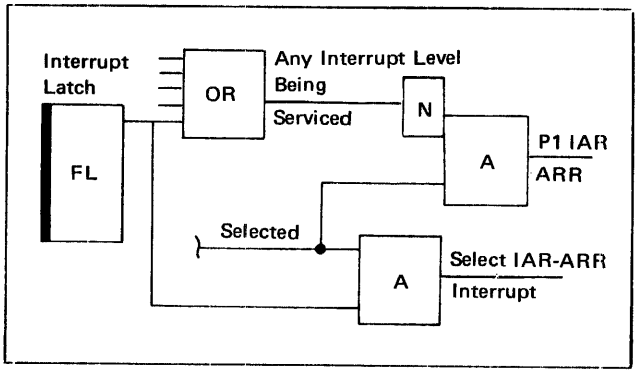


Figure 3-64. Interrupt–IAR/ARR Selection 53550

FEMD 5-230 contains the circuit description.

Load I/O–LIO

- Moves two bytes from storage to register selected by I/O attachment.
- Follows command format if device is busy or needs attention.
- A Q code of 0/0 results in a no op condition.

The load I/O instruction is a single address instruction that can be executed only if the addressed device is not busy and does not need attention. If the instruction cannot be executed it follows a command format and loops on the instruction in the same manner as an SIO instruction.

When it can be executed, the load I/O instruction removes two bytes from storage and loads them into a register selected by the device attachment (Figure 3-65). The register may be located in the attachment or may be an LSR in the CPU. In either case, two B-cycles are required to remove the bytes from storage.

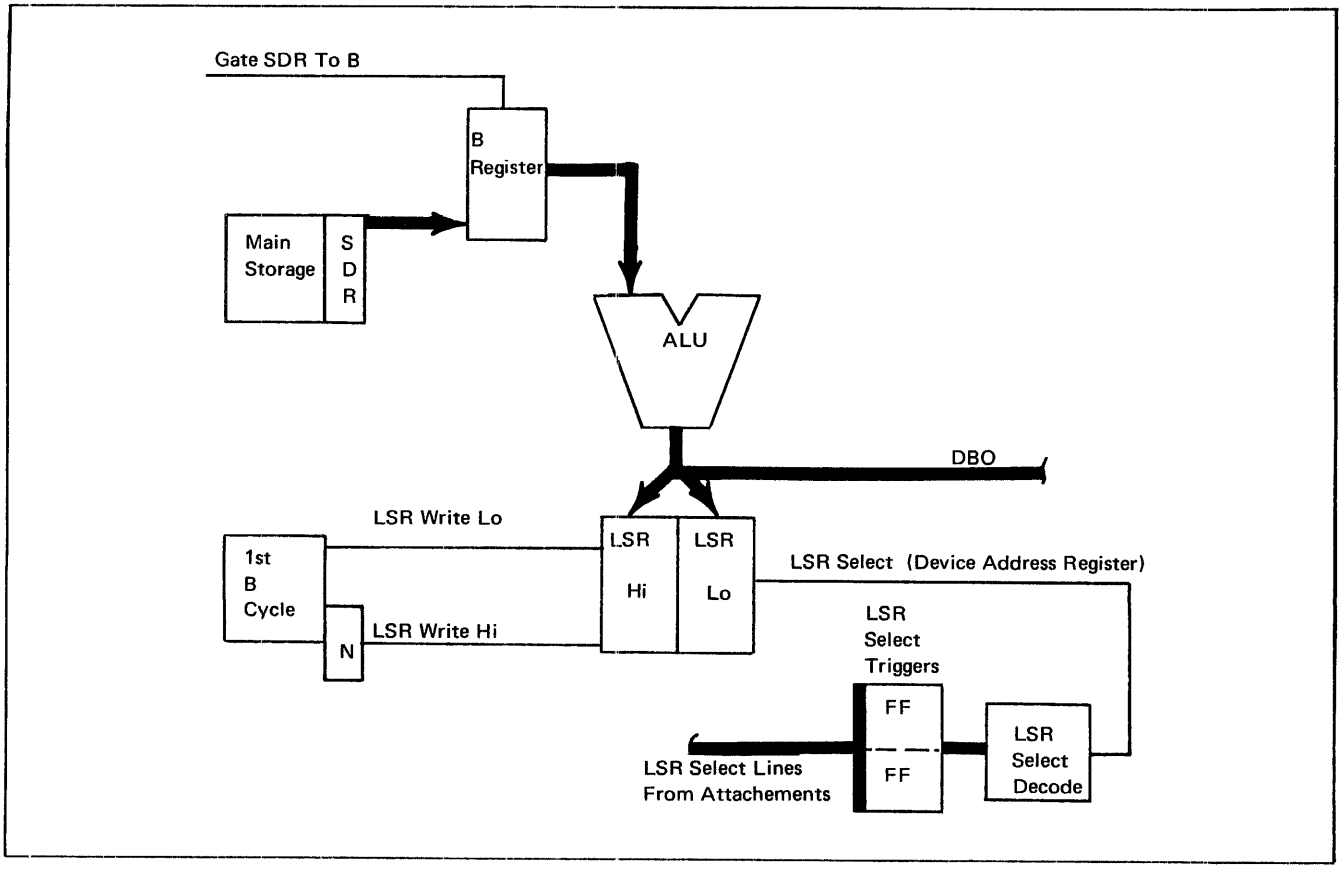


Figure 3-65. Load I/O Data Flow

The device address and M fields are located in bits 0-4 of the Q byte just as in any I/O instruction. The N field (bits 5-7) contains the register to be selected by the device attachment. Refer to the theory of operations manual for the individual attachments for the register selection codes.

During the I-Q cycles, device selection and 'I/O condition A' and 'I/O condition B' line control is the same as in a start I/O instruction. If program interlock is activated, an I-R cycle is forced. During the I-R cycle, because program interlock is active, I-R program back-up is activated to loop the instruction in the same manner as in an SIO instruction.

If the device attachment can execute the instruction, the B field address is loaded into the BAR with either I-H1 and I-L1 cycles or an I-X1 cycle. After the I cycles, the CPU enters into the first of two B-cycles.

During the first B cycle, the first byte is loaded into the B register and is passed through the ALU with no ALU controls (Figure 3-65). If the device attachment selects an LSR, the ALU output is written into the low order position of the selected LSR. If no LSR is selected, the ALU output is available on DBO to be entered into a register selected in the device attachment.

The BAR is decremented during clocks 5 to 8 times and in the second B cycle, the data byte is entered into the high order position of the selected register. The op-end trigger is turned on and the operation ends.

FEMD 5-160 contains the circuit descriptions.

Sense I/O—SNS

- Moves two bytes from register selected by I/O attachment to storage.
- Instruction executed even if device is busy or needs attention.
- Q code of hex 0/0 senses console address/data switches.

The sense I/O instruction moves two bytes of data from a register selected by the device attachment to main storage. The register may be located in the attachment or may be an LSR in the CPU. The instruction is executed regardless of whether the device is busy or not.

The device address and M fields are located in bits 0-4 of the Q byte just as in any I/O instruction. The N field (bits 5-7) contains the code for the register to be selected by the device attachment. Refer to the theory of operations manual for the individual attachments for the register selection codes.

If the Q code contains a hex 0/0, the two bytes of data set up in the console address/data switches are moved to the address specified in the B field address and that address minus one. The first B cycle moves the data set up in the two rightmost switches, and the second B cycle moves the data set up in the two leftmost switches.

During the I-Q cycle, device selection is the same as in a start I/O instruction except that the device attachment activates the I/O condition B line whether it is busy or not. Valid address checking and valid N code checking remains the same.

The B field address is loaded into the BAR in the same manner as in all one address instructions. After the I-cycles, the CPU takes two B-cycles to store the registers in the B field storage location.

If the register selected by the attachment is an LSR, the CPU activates 'gate LSR low normal to A' during clock 3 and 4 of the first B-cycle (Figure 3-66). The low order of the selected LSR is then loaded into the A register where it is binary added to the B register. Since the B register contains all zeros, the result is the same as the LSR low order byte. 'Store new' gates the result into the SDR and 'read call/write call' gates the byte into storage.

The BAR is decremented and the LSR high order byte is transferred in the second B cycle (Figure 3-66). The op-end trigger is turned on and the operation ends.

If the register selected by the attachment is not an LSR, gate I/O bus to A is activated in both B cycles and the bytes enter the A register from DBI.

FEMD 5-170 contains the circuit description.

Test I/O and Branch—TIO

- Test for specified I/O condition.
- Use branch-to address for next instruction if condition exists.
- N code specifies condition tested for.

The test I/O and branch instruction is a one address instruction that tests an I/O device for a specified condition and branches if that condition exists. The instruction is the same as a normal branch except that the I/O device is tested instead of the CR.

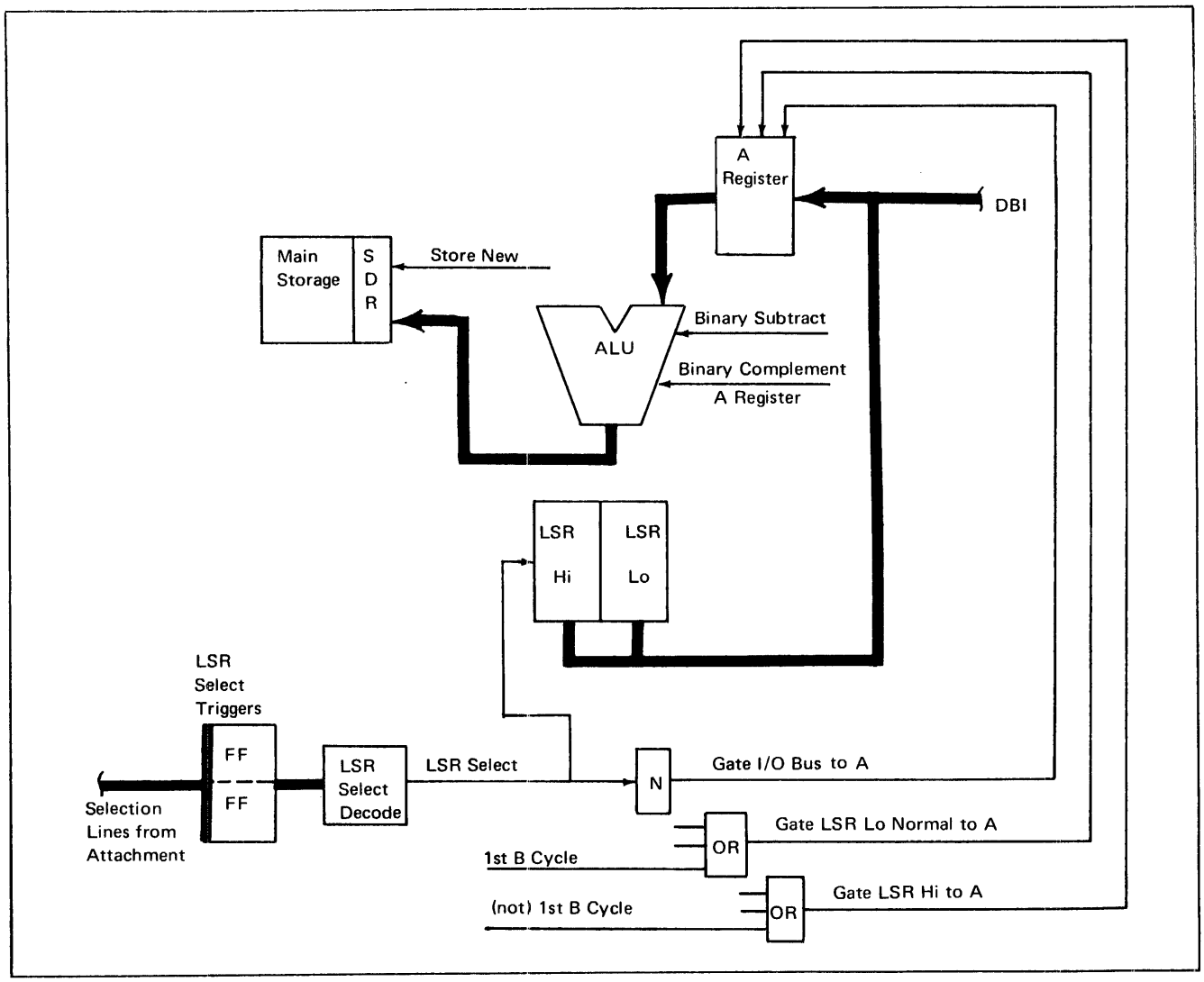


Figure 3-66. SNS I/O Data Flow

The device address and M fields are located in bits 0-4 of the Q byte just as in any I/O instruction. The N field (bits 5-7) contains the code for the condition being tested for. The conditions vary with each attachment. Refer to the theory of operations manual for the individual attachments for the condition codes.

Branching is determined by the device control of the 'I/O condition A' and 'I/O condition B' lines. Figure 3-67 shows the significance of the line settings.

During the I/Q cycle, device selection is the same as in other I/O operations. The device attachment also tests for the condition specified in the N field (Figure 3-68). If the

Line Activated By Any Device	Significance
'I/O condition B' only	Correct address, valid N code, condition for branching not met—proceed with next sequential instruction.
'I/O condition A' only	Correct address, valid N code, condition for branching met—branch to new address.
Both lines	Incorrect parity—causes processor check and DBO parity check.
Neither line	Invalid address or N code—causes processor check and invalid device address.

Figure 3-67. TIO—Device Response

condition is met, the attachment activates the 'I/O condition A' line. With the 'I/O condition B' line inactive at clock 8 time, the branch condition is latched in the CPU.

The ARR is loaded with I-H1 and I-L1 cycles or an I-X1 cycle the same as in a normal branch operation. The 'LSR intchg pulse' is then activated to switch the IAR and ARR (Figure 3-68). Refer to Branch on Condition for the IAR/ARR interchange explanation.

A TIO with the device address and M code of 0 will test the dual programming control switch for the condition specified in the N code. Refer to 'Dual Program Feature' in Chapter 4 for an explanation of the N code function.

FEMD 5-180 contains the circuit description.

Advance Program Level—APL

The advance program level instruction is a command instruction used primarily with the dual program feature. Therefore, the operation is covered under Dual Program Feature in Chapter 4. However, for programming compatability, machines without the dual program feature will accept the instruction. The following description applies to machines without dual program feature.

- Test for specified I/O condition.
- Loop on APL instruction until condition no longer exists.
- N code specifies condition tested for.
- Automatic APL is same as no-op.

The advance program level instruction tests for the same conditions as a test I/O and branch operation. The N code is the same as in TIO and varies with each attachment. If the advance condition is met, the attachment activates the 'I/O condition A' line just as in the TIO.

With the 'I/O condition B' line inactive, program interlock is activated in the I-Q cycle the same as in a start I/O instruction. During the I-R cycle, with program interlock active, 'IR prog back-up' decrements the IAR by 2 just as in the SIO instruction. When the advance condition is not met, and 'I/O condition B' is activated by the attachment, the IAR is incremented in the normal manner.

An all zero Q code is recognized as an automatic advance instruction. With this condition the 'IR program back-up' line is blocked and the IAR is incremented in the normal manner. Thus, an automatic advance instruction is equivalent to a no-op operation.

FEMD 5-200 contains the circuit description.

Initial Program Load—IPL

- Program load key initiates a system reset cycle.
- IPL activates data bus out 7 for MFCU read.
- IPL activates data bus out 5 for Disk read.

The initial program load operation is started by pressing the program load key. Pressing this key initiates a system reset cycle and selects the MFCU or file by activating a Data Bus Out line. DBO bit 7 selects the MFCU while DBO bit 5 selects the Disk. The input device selected is determined by the program load selector switch on the operators console (Figure 6-7).

After the input device is selected, it activates the select line to add all zeros into its DAR and the IAR. The data bus out line causes the device to load one card or block into main storage beginning at address 00 by using the normal cycle steal sequence. After one record or card has been transferred to storage, the IPL line is reset with I/O condition B giving a machine cycle advance. Since the IAR was reset to zero the first I op is in location 0000.

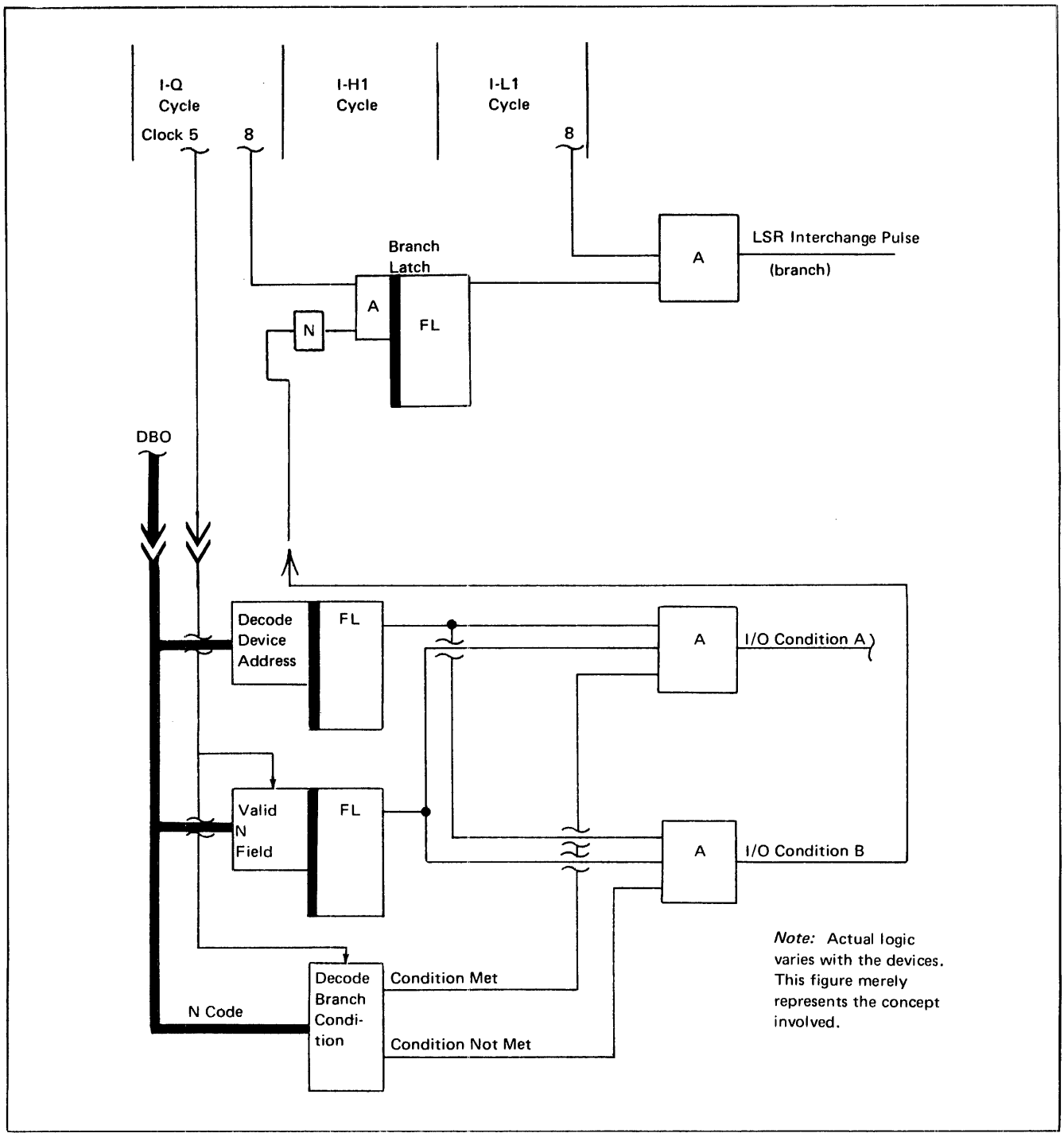


Figure 3-68. TIO-CPU/Device Control

DUAL PROGRAM FEATURE (DPF)

- Provides ability to execute two independent programs on a time-sharing basis.
- Feature is enabled/disabled by SIO instruction.
- Transfers to alternate program level when current program receives I/O busy.
- Transfers to alternate program level on advance program level (APL) instruction.

If while executing a program of instructions, an I/O device is busy, the dual program feature (DPF) can provide the ability to process another set of instructions. An SIO instruction with Q code of 00 controls the dual program feature. Control code bit 5 being on enables DPF. Control code bit 5 being off disables DPF. This instruction may be issued in either program level or any interrupt level, and enables/disables all program level advance instructions until another SIO changes the condition. An actual program level advance cannot, however, be executed during an interrupt and will result in the instruction being ignored.

Program interlock (I/O busy) activates 'I-R program back-up' and, during the I-R cycle, the IAR is decremented by two (Force bit 6 to A). Op end and program interlock AND together to change the state of the program level control flip-latch (program 1/program 2).

There is a separate set of LSRs for each program level. The LSRs shared by both levels are: the AAR, BAR, and the LCR/DRR.

Test I/O

A test I/O instruction with a device address of 0 and M bit of 0 compares the N code with the DPF switch setting (Figure 4-1). If the control switch setting is the same as

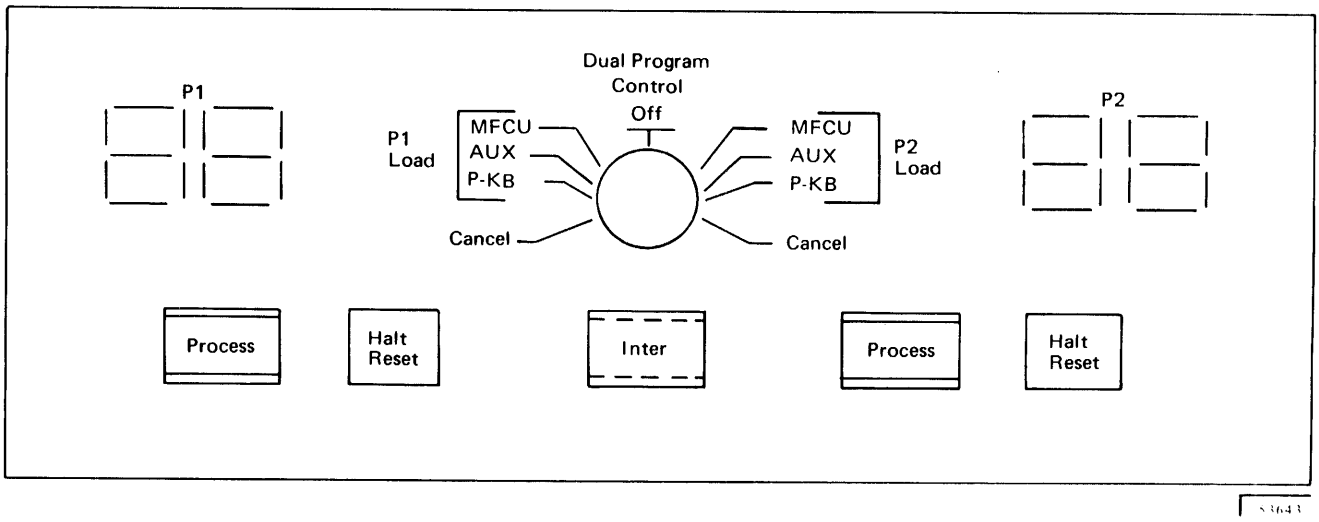


Figure 4-1. Dual Program Control Switch

that specified in the N code, the next address will be the specified branch to address. If the compare is unequal, the next sequential instruction will be executed. The function of the N code is shown in Figure 4-2.

During the I/Q cycle, the console DPF switch setting is compared with the condition specified in the N code. If the condition is met, 'DPF branch condition' is activated.

The ARR is loaded with I-H1 and I-L1 cycles or an I-X1 cycle the same as in a normal branch operation. 'DPF branch condition' activates the 'LSR interchange' pulse to switch the IAR and ARR.

FEMD 5-180 contains the circuit description.

Advance Program Level—APL

The advance program level (APL) instruction allows use of the dual program feature. The APL instruction is similar to the Test I/O and Branch instruction.

- Test I/O for specified I/O condition.
- Program advance if condition satisfied.
- N code specifies condition tested for.

The advance program level instruction tests for the same conditions as a test I/O and branch operation. The N code specifies the condition tested for, and varies with each I/O device attachment. If the advance condition is met, the I/O device attachment activates I/O condition A just as in the TIO.

With 'I/O condition B' inactive, program interlock is activated in the I-Q cycle. During the I-R cycle, 'I-R program backup' decrements the IAR by 2. The PSR contains the length count recall and condition recall data for the current program level and will be used when returning to this program level.

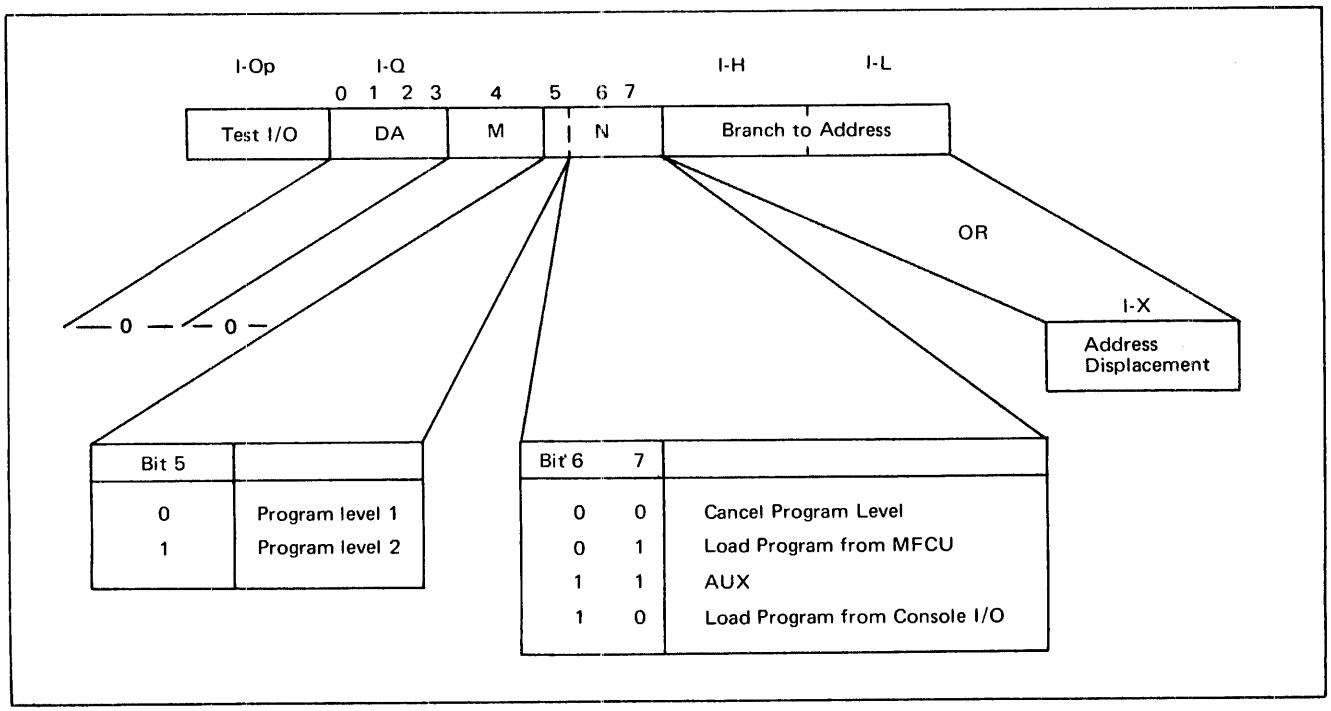


Figure 4-2. Test I/O Instruction Format (DPF)

BINARY SYNCHRONOUS COMMUNICATIONS ADAPTER

- Provides the system with the ability to function as a point-to-point or multipoint processor terminal.
- System can both transmit and receive during single communication.
- Data is transferred either in EBCDIC or ASCII.

The IBM Binary Synchronous Communications Adapter (BSCA) provides System/3 with the ability to function as a point-to-point (switched or leased) or multipoint processor terminal. Operation is half duplex synchronous, serial by bit and serial by character over communication facilities. Operation of BSCA is fully controlled by a combination of System/3 stored program instructions and BSCA responses to line control characters. With BSCA, the system can both transmit and receive, but half duplex operation prevents simultaneous transmission and reception of data.

The data set receives the data serially by bit and serially by character from the communications line during receive operations and presents the bits to the communications adapter. During transmit operations the communications adapter receives characters from storage serially, then makes them available serially by bit and serially by character to the data set. The data set places each bit on the communications line as soon as it receives the bit from the BSCA.

Data can be transferred either in Extended Binary Coded Decimal Interchange Code (EBCDIC) or American National Standard Code for Information Interchange (ASCII). Only units using the same code can communicate with each other.

Three local storage registers (two of which are located in the adapter) are provided for the adapter; the current address register, the transition address register, and the stop address register. These registers hold the storage addresses of data or line control characters at which certain actions are to occur, or the address of the next byte to be transmitted or received.

See the *IBM System/3 Binary Synchronous Communications Adapter Field Engineering Maintenance Diagrams Manual,* SY31-0258 for additional information on BSCA.

SERIAL INPUT/OUTPUT CHANNEL ATTACHMENT

- Provides a means of attaching additional input/output devices and special units for which attachment circuitry is not incorporated in the system.
- Control unit of attached device must be compatible with SIOC.
- Only one control unit can be attached to SIOC at any one time, but any number of devices can be attached to control unit.

The IBM Serial Input/Output Channel Attachment (SIOC) feature provides a means for attaching additional devices to System/3. These devices may be additional I/O devices for which attachment circuitry is not provided in the system or special units that may be requested by the customer. Only one control unit can be attached to SIOC at any one time and this unit must be compatible with SIOC. Any number of devices can be attached to the control unit, but only one of these devices may operate at a time.

SIOC provides an intermediate control unit between the system I/O unit and the device control unit. This intermediate control unit produces the necessary signals to control the device control unit from information furnished to the SIOC by:

- Instructions from the processing unit.
- Control bytes stored in registers in the SIOC by the processing unit.
- Information supplied by the device control unit.

The operation of SIOC requires that certain I/O instructions be performed to prepare the program and attachment for operation. The individual I/O devices that are attached to SIOC are identified by four identification lines.

The SIOC operates in interrupt mode on interrupt level 4. Each time the I/O device requires some special service from the processing unit, it interrupts the unit. Interrupts must be enabled for the I/O device before the SIOC can interrupt the processing unit.

See the *IBM System/3 Serial I/O Channel Attachment Field Engineering Maintenance Diagrams Manual*, SY31-0275 for additional information on the SIOC.

5471 PRINTER KEYBOARD ATTACHMENT

The IBM 5471 Printer Keyboard can, through the use of the 5471 Printer Keyboard Attachment, be attached to System/3. The printer keyboard is mounted on the system table top with a forms stand located on the floor behind it. The keyboard and the printer are not physically linked in that pressing a key does not automatically cause a character to be printed. The printer and keyboard are, however, housed together and the printer motor is used to restore the keyboard.

The printer has a 15 inch (381mm) carriage with a 10 pitch escapement and a pin feed platen with a 12.5 inch (317.5mm) writing line. The printer prints the system character set with the exception of minus zero. It operates in a closed loop mode; that is, when the attachment energizes the appropriate coils to perform a print, space, shift, or carrier return/index operation, an associated feedback contact closes signaling the attachment to end the coil pulses. When the feedback contact re-opens, it is a signal to the attachment that the printer is ready to perform the next operation. A print or space cycle requires 64.5 ms and the carrier returns at approximately 15 inches (381mm) per second.

The keyboard keys are mechanically interlocked in such a way that two keys cannot be pressed at the same time and yet rolling of keys is permitted. Using an array of nine reed switches (six data and one parity, one strobe, and the keyboard upper or lower case mode switch), the keyboard is capable of generating the system character set with the exception of minus zero. The keyboard also generates particular characters to signify shift key depression, shift key release, and return key depression. Automatic restore of the keyboard following a graphic, shift, or return key depression requires 64.5 ms. The keyboard also includes three noninterlocking keys (end, request, and cancel) which generate control signsl for the stored program and two indicators (proceed and request pending) which are controlled by the stored program.

See *IBM 5471 Printer Keyboard Attachment Field Engineering Maintenance Diagrams,* SY31-0259 for complete information concerning the operation of the printer keyboard attachment.

Section 1. Basic Unit

POWER SUPPLIES

Figure 5-1 lists System/3 power supplies. Figure 5-2 shows System/3 power distribution for machines with the early design power control box. See Figure 5-3 for power distribution if the redesigned power control box (printed circuit relay panel) is installed.

As features are added to the basic system, additional supplies must be added (Figure 5-1, Part B). See Section 2 for details about the feature power supplies.

Supply	Amperes	Where Used	Location
-4V	70A	Logic Voltage	CPU
+6V	15A	Logic Voltage	CPU
-30V	9.5A	Storage	CPU
+24V	25A	MFCU	MFCU
+60V	11A	Printer	Printer
+24V	5A	Control Voltage	CPU
+3V		Storage*	CPU
-14V		Storage**	CPU
7.25 Vac	25A	Indicator Lamps	CPU
41 Vac		Use Meter***	CPU

* +6V supply voltage dropped to +3V in storage module (B4C4 card).
** -30V supply voltage dropped to -14V in storage module (B4C4 card).
*** Applies to redesigned power control box only (printed circuit relay panel).

Ⓐ BASIC SYSTEM SUPPLIES

Supply	Where Used	Location
-12V	BSCA	CPU

Ⓑ FEATURE SUPPLIES

Figure 5-1. System/3 Power Supplies

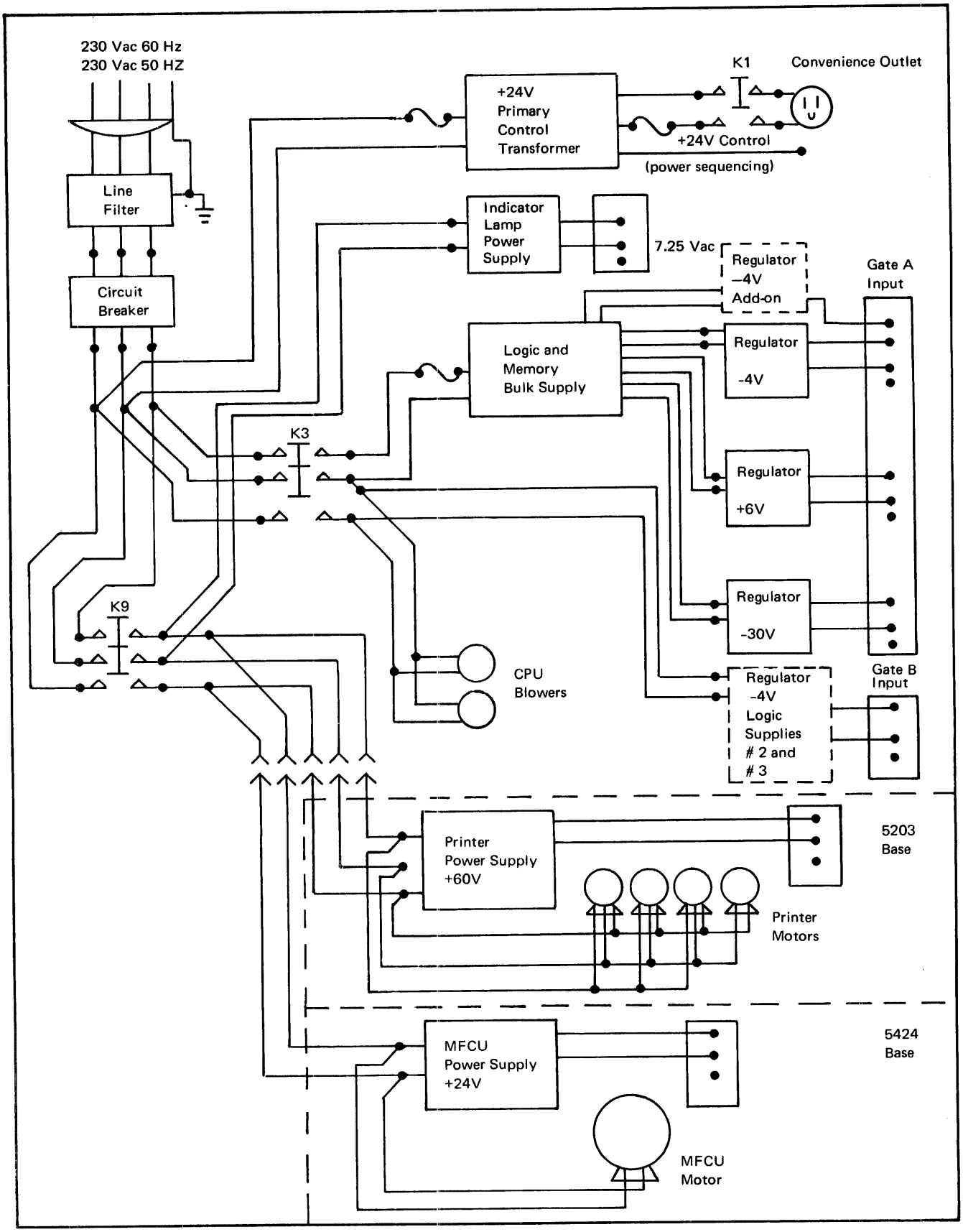


Figure 5-2. System/3 Power Distribution (Early Design)

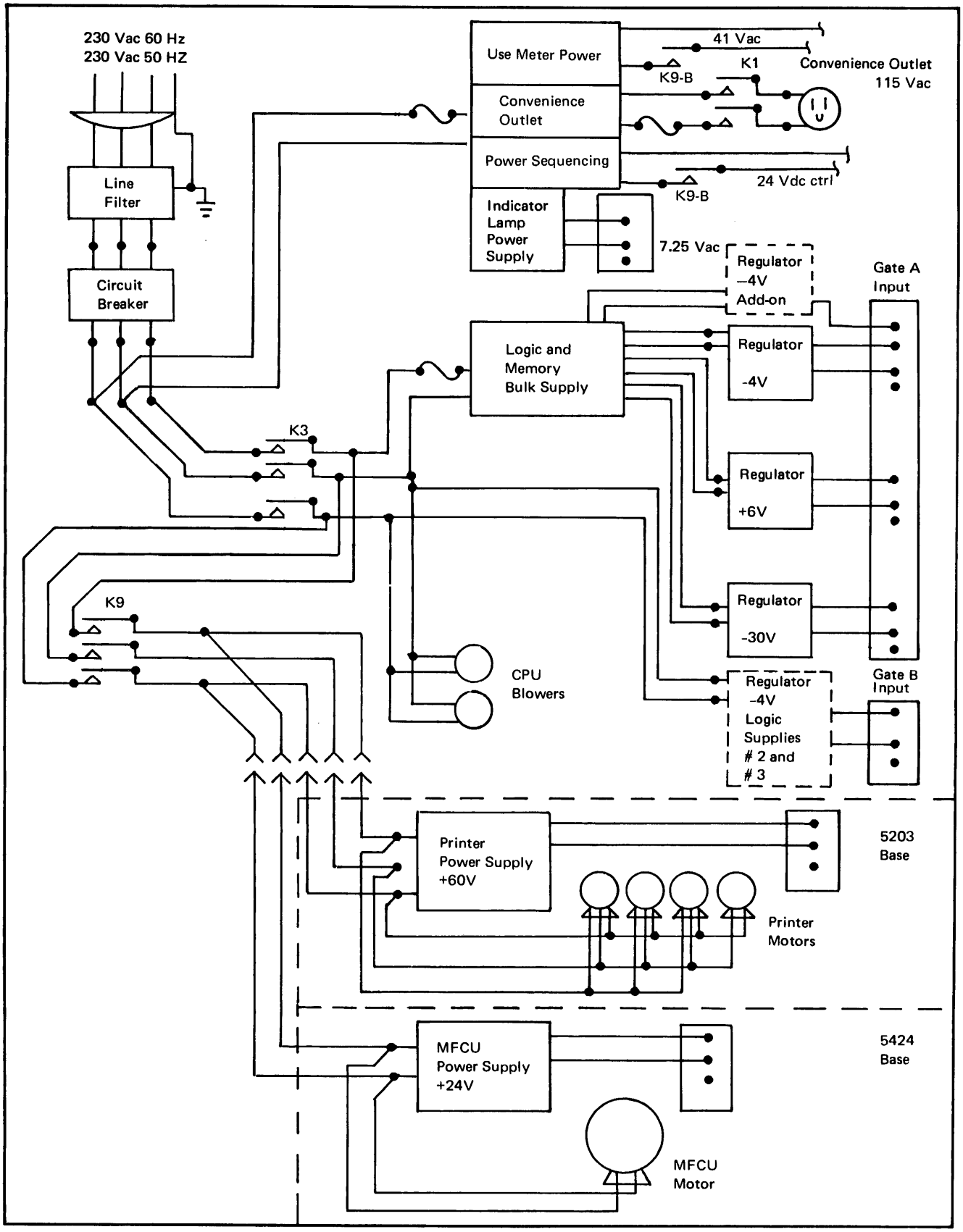


Figure 5-3. System/3 Power Distribution (Redesign)

POWER SUPPLY REGULATORS

The individual regulators receive the unregulated, filtered dc voltage from the logic and memory bulk power supply. The regulators provide the voltage regulation required to operate the system logic.

The -4V, +6V, and -30V regulators have identical terminals (E1-E14) that perform identical functions (Figure 5-4). In each regulator assembly, a regulator card provides regulated dc output and circuit protection.

Three inputs are required to raise a regulated output voltage:

1. Terminals E1 and E2—Bulk supply voltage.
2. Terminals E9 and E10—Bias supply voltage.
3. Terminal E12—Start signal (ground level) is applied to this terminal to turn on the regulator.

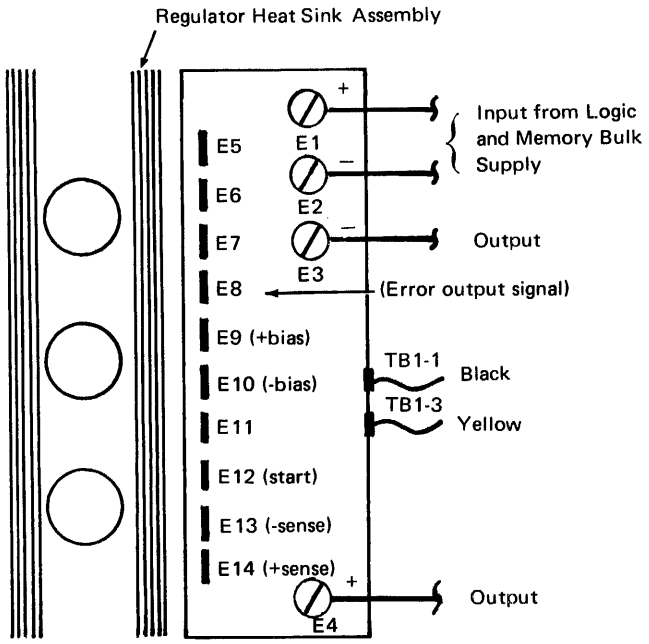


Figure 5-4. Power Supply Regulator

Voltage Regulation

Regulator output terminals E3 and E4 are connected to terminals E13 and E14 of the regulator card (Figure 5-5). A differential amplifier in the regulator card compares the output voltage at terminal E13 with the input bias voltage at terminal E9. The output of the differential amplifier is applied to the base of transistors Q1 and Q13 from terminal D11.

If an increase in output voltage is sensed, a negative voltage is applied to the base of transistors Q1 and Q13 causing the current through these transistors to decrease. A decrease in current results in a decrease in current through transistors Q2 and Q12, thereby causing a decrease in output voltage. For a decrease in output voltage, the current through transistors Q1 and Q13 increases causing an increase in current through transistors Q2 and Q12. Therefore, output voltage increases to the normal level.

Overvoltage Protection

An overvoltage protection circuit in the voltage regulator card monitors terminal E13 and E14 (Figure 5-5). Whenever the output voltage raises beyond the maximum normal limits, transistors Q1 and Q13 are turned off via the terminal D11 connection (Figure 5-5). Transistors Q1 and Q13 then turn off transistors Q2 and Q12 reducing the output voltage to zero. This action protects the logic circuits from an overvoltage condition.

If an overvoltage condition exists, system power must be turned off. When the overvoltage condition is sensed, the overvoltage circuit grounds terminal E8 via terminal D02 (Figure 5-5) to indicate a fault condition. Grounding terminal E8 causes the OV/OC relay to energize. Energizing this relay de-energizes contactor K3 which removes power from the primary of the bulk supply. See FEMD 6-010 for early design power control sequencing or FEMD 6-020 for redesigned power control sequencing.

Overcurrent Protection

The overcurrent protection circuit protects the regulator if load current exceeds the limits of the regulator. An overcurrent condition is sensed at terminal J02 of the regulator card (Figure 5-5).

The voltage drop across R4 and R24 (even-numbered resistors, Figure 5-5) is proportional to the load current.

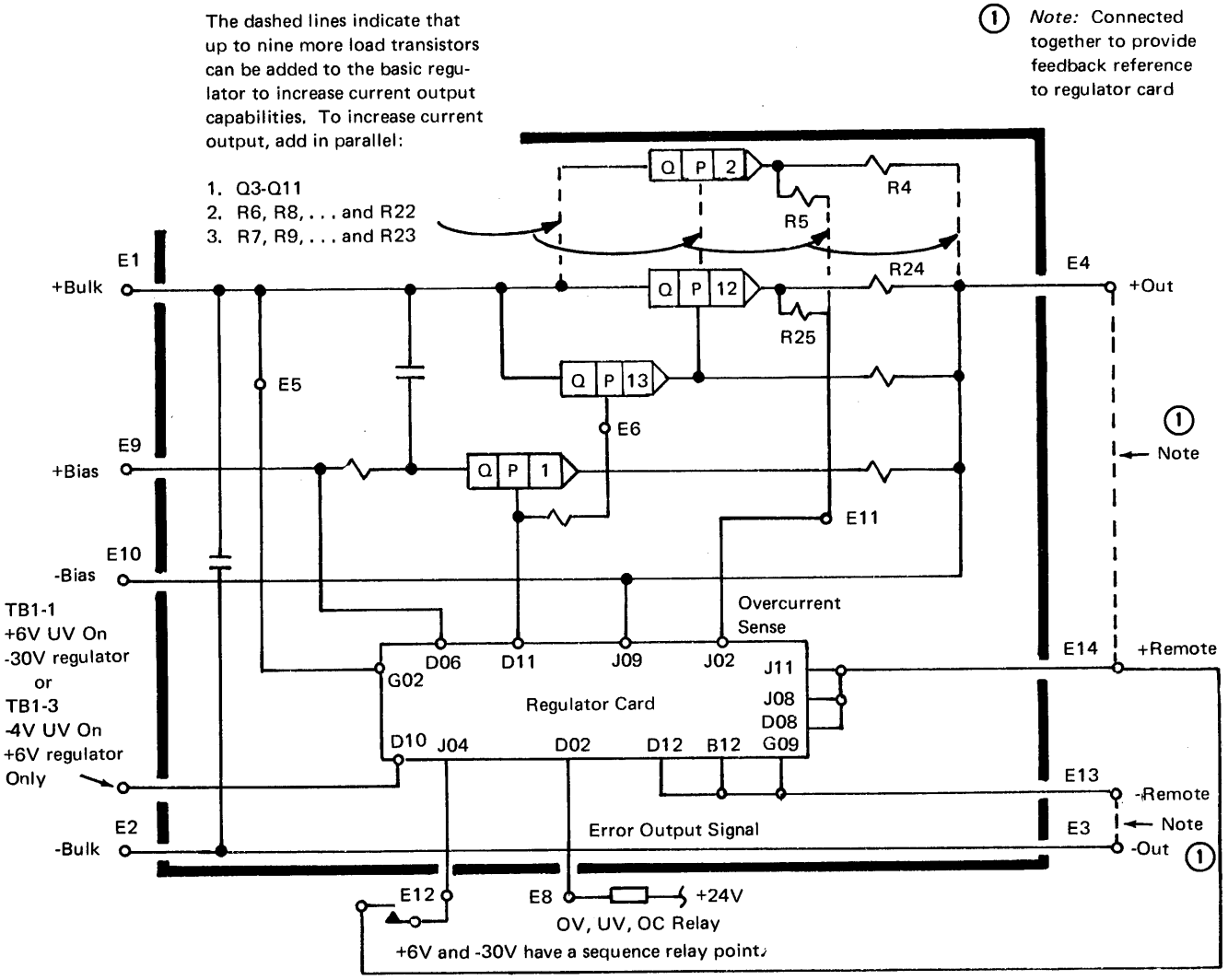


Figure 5-5. Power Supply Regulator Diagram

R5 and R25 (odd-numbered resistors, Figure 5-5) average these individual voltages. The average voltage feeds a differential amplifier in the regulator card via terminal J02. If the predetermined current limit for the average voltage is exceeded, a fault indication energizes the OV/OC relay by grounding terminal E8. Energizing the OV/OC relay K13, K14, or K15 de-energizes contactor K3 removing power from the primary of the bulk supply. See FEMD 6-010 for early design power control sequencing or FEMD 6-020 for redesigned power control sequencing.

Undervoltage Protection

An undervoltage protection circuit in the basic -4V supply and the +6V supply is necessary because:

1. Damage to the print magnets can occur if +6V is applied to the 5424 MFCU or the 5203 Printer when the -4V regulator output is low or decreases to 0. (The print magnet amplifiers continually drive the print magnets, which are not 100% duty cycle.)

2. Damage to the BSM logic cards may occur if -30V is applied to the BSM when the +6V regulator output is low or decreases to 0.

In order to protect the 5424 MFCU and the 5203 Printer, a silicon-controlled rectifier (SCR) is wired across the output of the +6V regulator. The SCR starts to conduct if:

1. The +6V regulator senses an overvoltage or an overcurrent condition.

2. The -4V undervoltage control circuit (ax card) senses an undervoltage condition.

When the SCR conducts, the +6V output is effectively shorted. This results in an overcurrent condition which, in turn, causes an immediate system power down.

The -30V regulator senses a +6V undervoltage condition (TB1-1 on -30V regulator). A +6V undervoltage condition results in dropping the -30V regulator and in energizing K15, the -30V regulator OV/UV/OC relay (-30V OV/OC or +6V UV). This results in de-energizing contactor K3, which causes an immediate system power down.

-4 Volt Undervoltage Circuit

The -4V undervoltage circuit monitors the output of the -4V regulator. If the undervoltage circuit detects an abnormal decrease in voltage, the -4V undervoltage control card ('ax drive', FEMD 6-010 for early design or 6-020 for redesign) turns on the +6V regulator SCR. This shorts out the +6V regulator output causing an overcurrent condition. The +6V regulator senses this simulated overcurrent condition and, as a result, picks K14. This, in turn, de-energizes K3, causing an immediate system power off. The action of shorting (axing) the +6V regulator when a -4V undervoltage condition is sensed protects the 5424 and the 5203 logic circuits.

+6 Volt Undervoltage Circuit

The +6V undervoltage circuit monitors the +6V regulator output. This circuit is located on the -30V regulator card.

The -30V regulator senses an undervoltage condition at the TB1-1 terminal. See FEMD 6-010 for early design power control sequencing or FEMD 6-020 for redesigned power control sequencing. Whenever the -30V regulator senses a +6V undervoltage condition, the -30V regulator error output signal energizes K15, the -30V OV/UV/OC relay (-30V OV/OC or +6V UV), causing an immediate system power down.

NORMAL POWER ON SEQUENCE (EARLY DESIGN POWER CONTROL)

A +24V control voltage controls power sequencing. Turning on the mainline circuit breaker activates this supply (Figure 5-2). Relays K1 and K2 energize to supply power to the convenience outlets and to allow power to be turned on.

Turning on the power on/off switch results in energizing contactor K3. Power is then applied to the logic and memory bulk power supply (Figure 5-2). Bulk power and bias voltage from the logic and memory bulk power supply is then applied to the inputs of the -4V, +6V, and -30V regulators. Note that the regulator output cannot be raised without the start signal applied to terminal E12 of each regulator. Because terminal E12 is connected to ground, the -4V regulator output starts sequencing up as soon as input voltage is applied to the regulator.

Note: If installed, the -4V logic power supplies #2 and/or #3 sequence up at this time.

The -4V sense relay (K5) energizes after -4V is available. Relay K5-1 contacts provide the start signal to the +6V regulator (FEMD 6-010).

After the +6V output is available, the +6V sense relay (K6) energizes relay K7 which provides the start signal to the -30V regulator. When -30V output is available, the -30V sense relay (K8) energizes which, in turn, energizes relay K9. Power is then applied to the primaries of the +24V MFCU, the +60V printer power supplies, and the indicator lamp power supply. When the +24V and the +60V supply outputs are available, sense relays K10 and K11 energize. Energizing K10 and K11 allow power sequence relay K12 to energize. Relay K12-1 contacts turn off the power check light indicating that the power on sequence is complete. See FEMDs 6-005 and 6-010 for the detailed description of the power on sequence.

NORMAL POWER OFF (EARLY DESIGN POWER CONTROL)

A normal power off condition causes the power supplies to turn off in the following order:

1. -30V storage supply voltage

2. +24V MFCU and +60V printer logic voltages

3. +6V logic voltage

4. -4V logic voltage

Turning off the power on/off switch de-energizes relays K7 and K9. Relay K7-1 contacts remove the start voltage from the -30V regulator pin E12 (FEMD 6-010). Contactor K9 contacts transfer shortly after the relay is de-energized. Contactor K9 then removes power from the MFCU and the printer power supplies after contactor K9 contacts transfer. Relays K10 and K11 de-energize after the MFCU (K10) and the printer (K11) power supply outputs are no longer available. De-energizing these relays causes contactor K3 to de-energize. This results in removing power from the logic and memory bulk supply. This removal, in turn, causes the -4V and +6V regulator outputs to drop at the same time. See FEMDs 6-005 and 6-010 for a detailed description of the normal power off sequence.

NORMAL POWER ON SEQUENCE (REDESIGNED POWER CONTROL–PRINTED CIRCUIT RELAY PANEL)

Turning on the mainline circuit breaker applies power to transformer T1 (Figure 5-3). The outputs of T1 supply 41Vac for use meters, 7.25Vac for indicator lamps (through K9B points), 110Vac to the convenience outlets, and 24Vac to the +24Vdc power supply (see FEMD 6-020). Relay K1 energizes to supply power to the convenience outlet and K2 energizes to allow power to be turned on. A +24Vdc control voltage controls power sequencing.

Turning on the power on/off switch results in energizing relay K3. Power is then applied to the logic and memory bulk power supply (Figure 5-3). Bulk power and bias voltage from the logic and memory bulk power supply is then applied to the inputs of the -4V, +6V and -30V regulators. Note that the regulator output cannot be raised without applying the start signal to terminal E12 of each regulator. Because terminal E12 is connected to ground, the -4V regulator output starts sequencing up as soon as input voltage is applied to the regulator.

Note: If installed, the -4V logic power supplies #2 feature and/or #3 feature sequence up at this time.

The -4V sense relay (K5) energizes after -4V is available. Relay K5-2 contacts provide the start signal to the +6V regulator (FEMD 6-020).

After the +6V output is available, the +6V sense relay (K6) starts the time delay circuit. When the time delay circuit times out, transistors T3 and T4 conduct, providing a ground level start signal to the -30V regulator. When -30V output is available, the -30V sense relay (K8) energizes which, in turn, energizes relay K9. When the K9B points are closed, the circuit for the 7.25Vac lamps and the 41Vac use meter power is complete. Power is then applied to the primaries of the +24V MFCU and the +60V printer power supplies. When the +24V and the +60V supply outputs are available, sense relays K10 and K11 energize. Energizing K10 and K11 allow power sequence relay K12 to energize. Relay K12-2 contacts turn off the power check light indicating that the power on sequence is complete. See FEMDs 6-015 and 6-020 for the detailed description of the power on sequence.

NORMAL POWER OFF (REDESIGNED POWER CONTROL)

A normal power off condition causes the power supplies to turn off in the following order:

1. -30V storage supply voltage
2. +24V MFCU and +60V printer logic voltages
3. +6V logic voltage
4. -4V logic voltage

Turning off the power on/off switch de-energizes relay K9 and turns off transistors T3 and T4 thereby removing the start voltage from the -30V regulator pin E12 (FEMDs 6-015 and 6-020). Contactor K9 contacts transfer shortly after the relay is de-energized. Contactor K9 then removes power from the MFCU and the printer power supplies after contactor K9 contacts transfer. Relays K10 and K11 de-energize after the MFCU (K10) and the printer (K11) power supply outputs are no longer available. De-energizing these relays causes K3 to de-energize. This results in removing power from the logic and memory bulk supply. This removal, in turn, causes the -4V and +6V regulator outputs to drop at the same time. See FEMDs 6-015 and 6-020 for a detailed description of the normal power off sequence.

Abnormal Power Off

The five causes for an abnormal power off sequence are:

1. Overvoltage (OV)
2. Overcurrent (OC)
3. Undervoltage (UV)
4. Thermal (overheating–normal power off sequence)
5. Emergency power off switch opened

Overvoltage and Overcurrent Power Off Sequence

Whenever an overvoltage or an overcurrent condition is sensed, one of the OV/OC relays (FEMDs 6-010 or 6-020) energizes the -4V supply (K13), the +6V supply (K14), and the -30V supply (K15). Energizing OV/OC relay results in de-energizing contactor K3. De-energizing contactor K3 removes power from the logic and memory bulk supply.

On an abnormal power off, the power check indicator turns on to indicate a failure. Test points indicate the power supply that failed. The energized OV/OC relay contacts hold the relay energized until the check reset switch is pressed with the on/off switch off.

After an overvoltage, overcurrent, or an undervoltage failure, the check reset key must be pressed with the power on/off switch in the off position to de-energize the OV/OC /UV relay and to allow a power on sequence. See FEMD 6-010 for early design power control sequencing or FEMD 6-020 for redesigned power control sequencing. Power sequencing failures do not require this action.

Undervoltage Power Off Sequence

Only the -4V and the +6V outputs sense for undervoltage conditions. If the -4V ax card senses an undervoltage condition, the -4V ax circuit (a separate card) immediately shorts the +6V regulator via the SCR across the +6V regulator output. This is a +6V simulated overcurrent condition and the OV/OC/UV relay K14 energizes. The K14-1 (K14-2 redesign) contacts remove +24V from contactor K3. Contactor K3, in turn, removes power to the logic and memory bulk supply. This results in an immediate system power off.

Because K14 OV/OC/UV relay energizes, +24V is present at test point 13 to indicate a +6V power failure. However, a +6V overvoltage, a +6V overcurrent, or a -4V undervoltage could cause the failure condition.

The -30V regulator senses a +6V undervoltage condition. When the -30V regulator senses a +6V undervoltage condition, the -30V OV/OC/UV relay (K15) energizes. See FEMD 6-010 for early design power control sequencing or FEMD 6-020 for redesigned power control sequencing.

Contactor K3 then de-energizes causing an immediate system power off. Test point 14 equals +24V indicating a -30V overvoltage, a -30V overcurrent, or a +6V undervoltage condition.

Thermal Power Off Sequence

A thermal condition causes relay K2 to be de-energized. The K2-1 (K2-3 redesign) contacts turn on the thermal light to indicate overheating. Power then sequences off the same as a normal power off sequence by opening the power on/off switch circuit.

The thermal light and the power check light are on when the system power off sequence ends. Turning the power on/off switch off turns off the power check light. The thermal light remains on until the over-temperature condition has been corrected and the power on/off switch has been turned off. Power can then be restored to the system by turning the power on/off switch on.

Emergency Power Off

When you pull the emergency power off switch, system power drops immediately. However, power is still applied to the power input terminals and to the input terminals of K1, K3, and K9. If the redesigned power control box is installed (printed circuit relay panel), power is not applied to the input terminals of K9 after an emergency power off.

Test Points (TPs)

Test points (TPs) are on the power control box. When a voltage failure occurs, check these TPs inmumerical sequence to determine the voltage that failed. TPs 2-9 indicate which regulator voltage failed during power on or off sequence.

*Note:*In a normal system power off state, TP2 will read +24

Note: In a normal system power off state, TP2 will read +24Vdc. Because of a system power failure (power check), +24Vdc measured on TP2 indicates the -4Vdc failed to sequence up.

For example, a +6V regulator sequencing failure is indicated if TPs 2-4 were zero volts and +24V appeared at TP5. TPs 10-12 indicate an overvoltage condition, or an overcurrent condition. TP13 or TP14 indicates an overvoltage condition, an overcurrent condition, or an undervoltage condition. For example, an overvoltage/overcurrent failure in the -4V regulator occurred if +24V appeared at TP12. If TP13 or TP14 indicated +24V (failure condition), you may have difficulty in determining if an overvoltage, an overcurrent, or an undervoltage condition caused the failure. Refer to the MAP charts to help isolate the failure.

Section 2. Features

As features are added to the basic system, additional power supplies must be added:

1. –4V feature add-on regulator
2. –4V logic supply #2
3. –4V logic supply #3
4. -12 BSCA supply

The –4V feature add-on regulator is a prerequisite for each of the following features:

1. Disk file
2. Serial Input/Output Channel (SIOC)
3. Printer keyboard
4. 24K and 32K storage
5. B gate features
6. Dual program feature
7. Feature LSRs

System configuration determines the power supply requirements. As features are installed on the B gate, the –4V logic #2 feature supply and the –4V logic #3 feature supply are added to meet the increase in power requirements.

Note that the #2 feature supply must be installed before the #3 feature supply can be installed. Each feature logic power supply has its own bulk transformer, regulator, and cooling system.

The -12V feature power supply is installed in System/3 when features (such as the BSCA) are installed. A secondary winding of the bulk supply (also used by the +6V basic regulator) supplies unregulated ac voltage to the -12V supply. See FEMD 6-010 for early design power control sequencing or FEMD 6-020 for redesigned power control sequencing.

TP9 senses that the -12V power supply output is available to the BSCA. The absence of the -12V output prevents the system from powering up.

Section 1. Console

SYSTEM CONTROL PANEL

The system control panel (Figures 6-1 through 6-5) contains the lights and switches required to operate and control System/3.

System controls are divided into three sections: operator controls, customer engineering (CE) controls, and console display. The operator section contains the controls required for normal operation. The CE controls serve as diagnostic aids in locating malfunctions. The console display provides the operator and the CE with a visual record of the contents of the various registers in the CPU and the status of the major CPU controls.

Operator Controls

Emergency Power-Off Pull Switch (EPO)

Pulling this switch (Figure 6-3) turns off the power beyond the power-input terminal on every unit that is part of the system. The switch latches in the out position.

When the emergency pull switch is in the out position, the power on/off switch is ineffective.

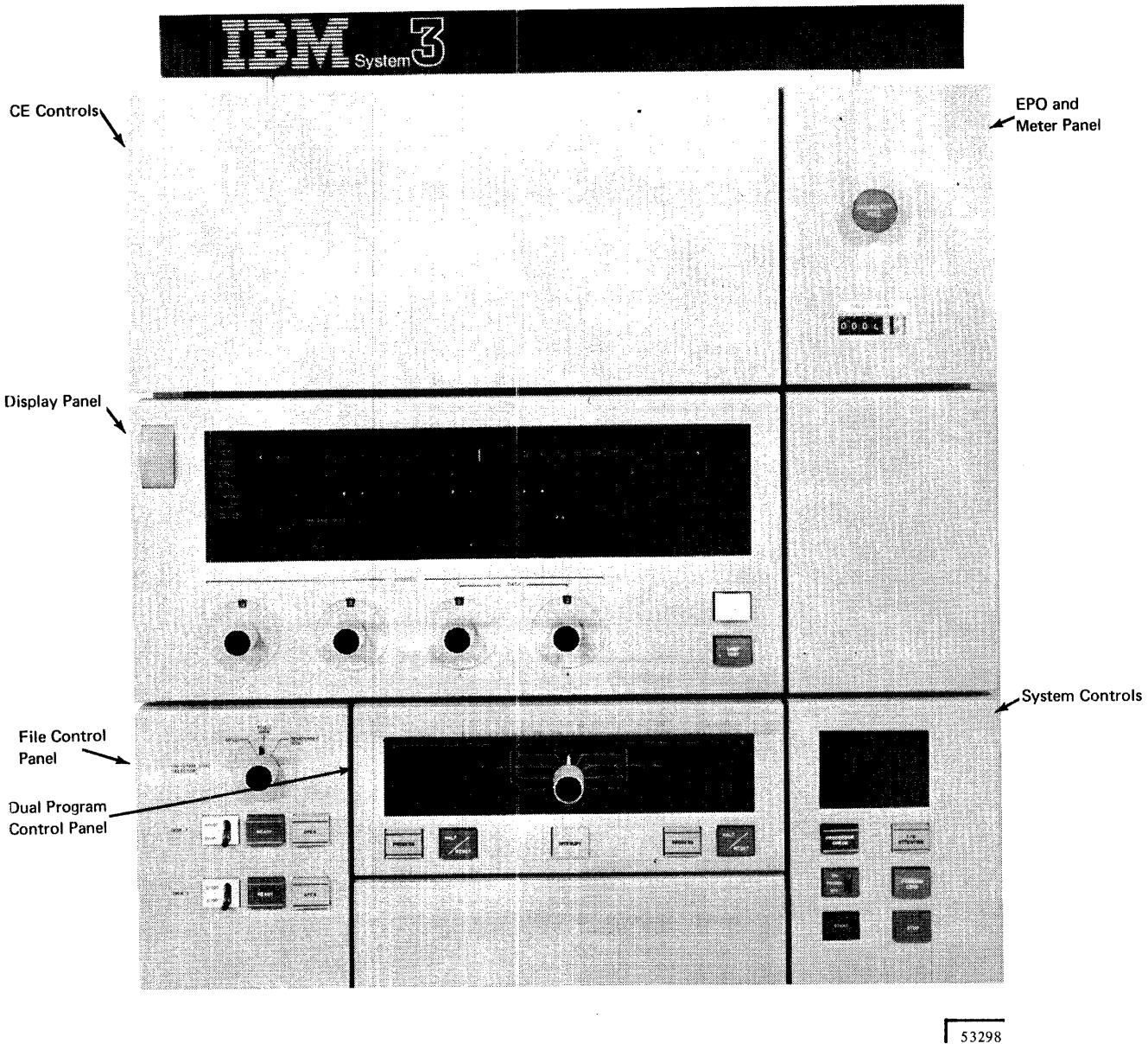


Figure 6-1. System Control Panel

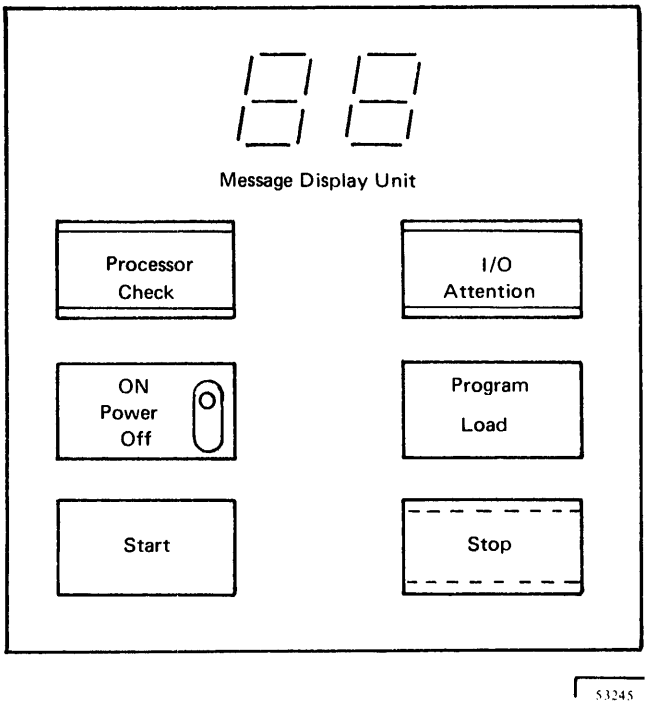


Figure 6-2. System Controls

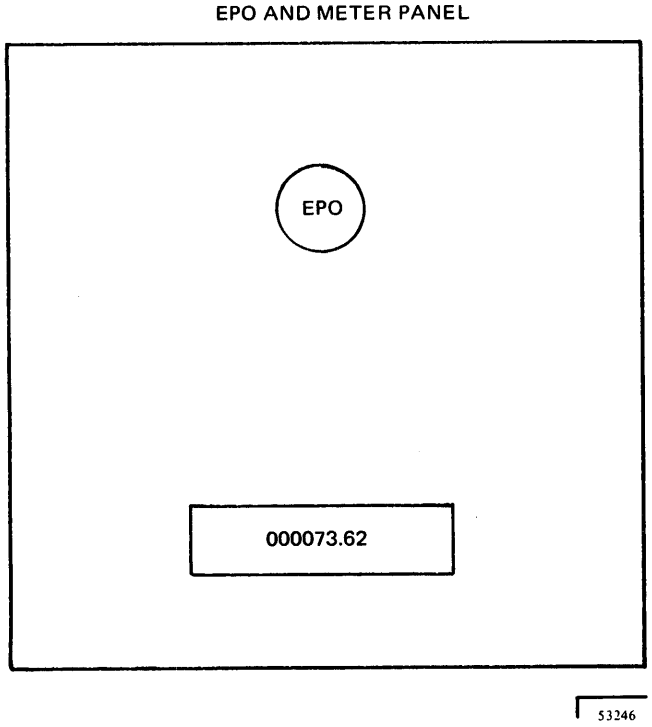


Figure 6-3. EPO and Meter Panel

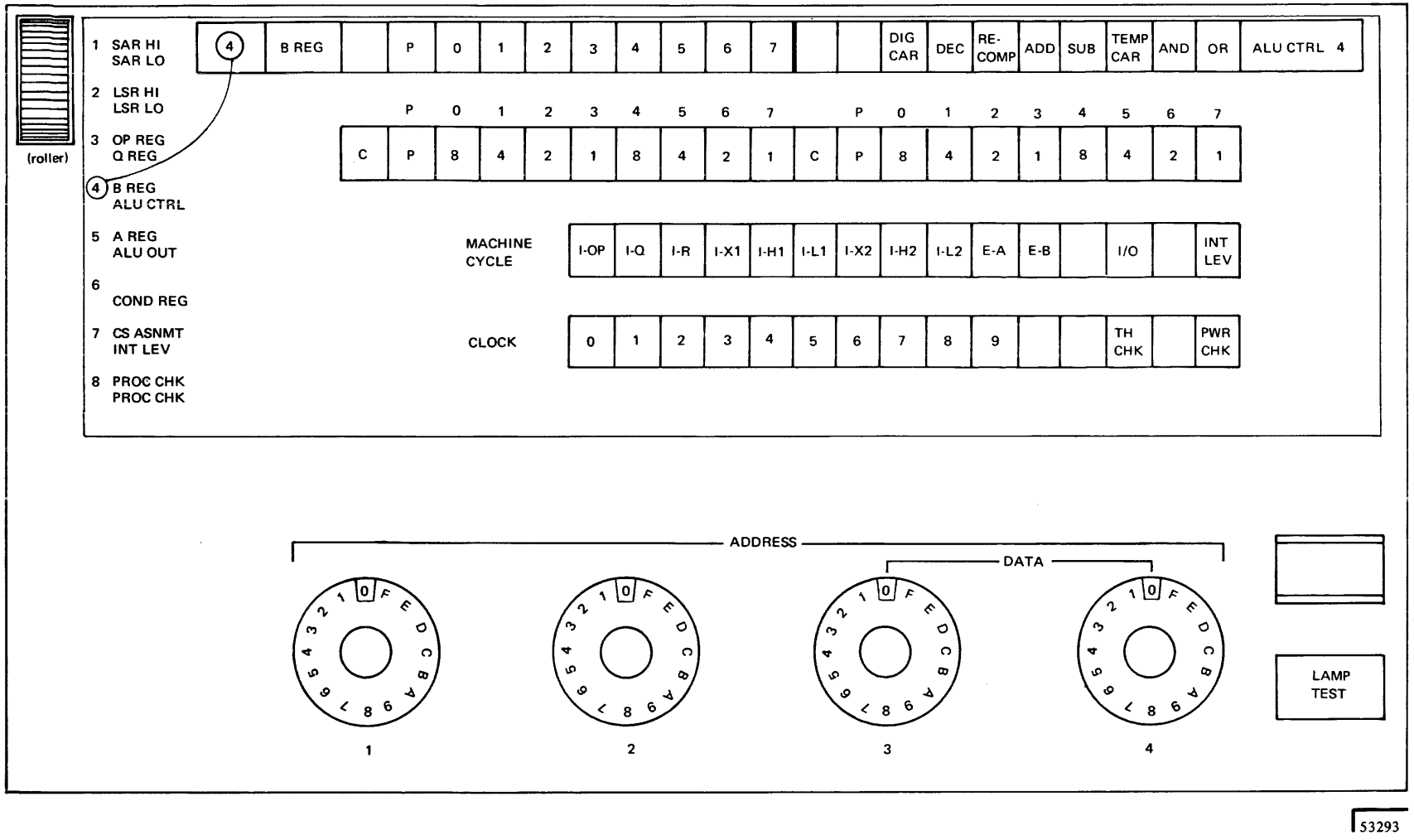


Figure 6-4. CPU Displays

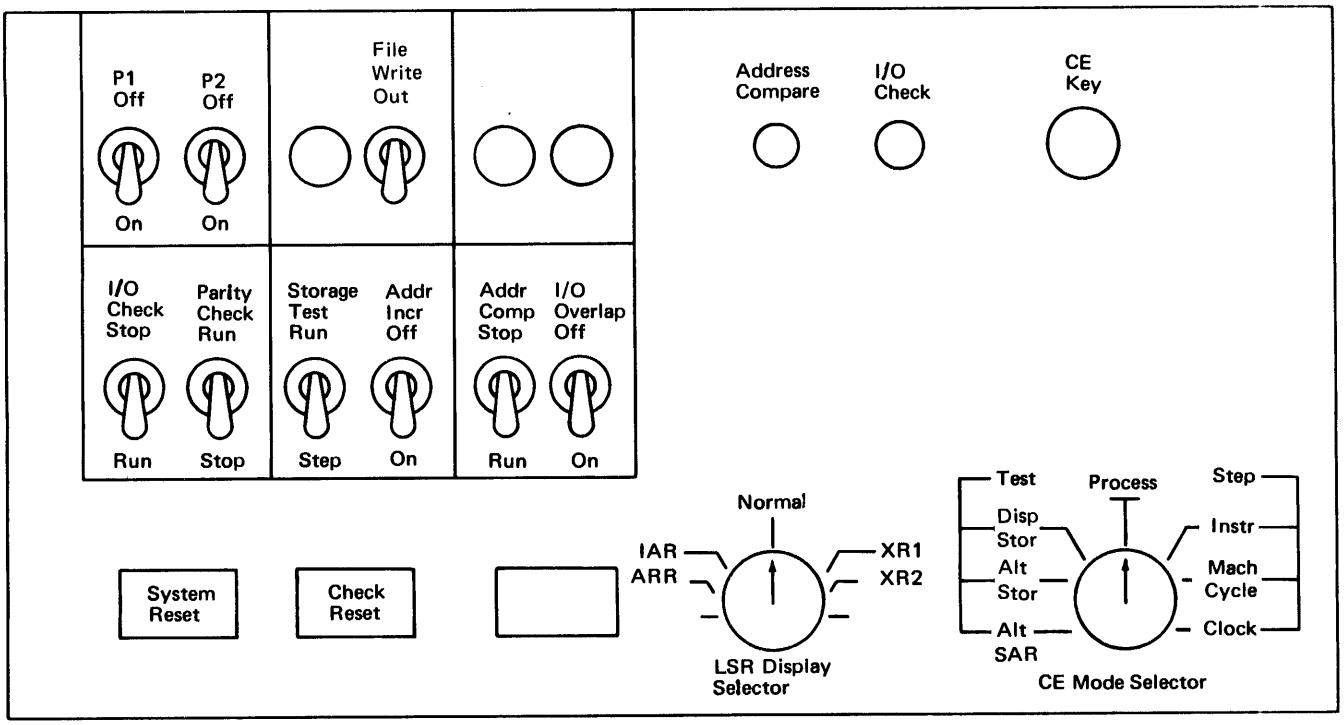


Figure 6-5. CE Control Panel

Message Display Unit

This two-position display unit (Figures 6-6 and 6-7) keeps a running display of the halt identifier portion of a halt instruction.

The display characters, generally the numerics 0 through 9, are selected by unique codes contained in the second and third bytes of a halt instruction.

Processor Check Light

This light is turned on when an invalid op code, an invalid address, or a parity error is detected in the CPU. It is also turned on whenever the device address (including the M field) and/or the N field of an I/O instruction is not recognized or whenever an I/O device recognizes a parity error on data bus out at the I/O attachment. It is turned off with system reset or CE check-reset.

The processor comes to an immediate stop on any of the above errors, and the input/output data may be lost. The specific error is displayed in the console display section.

I/O Attention Light

The I/O attention light is turned on when an addressed I/O device requires normal operator intervention. The light is turned off when the cause for I/O attention is removed and the device is returned to the ready state.

I/O attention does not stop normal CPU processing. However, start I/O or load I/O instructions will not be accepted. Normal operator intervention includes:

Printer - forms out, cover interlock
MFCU - hopper empty, stacker full, chip box full, cover interlock.

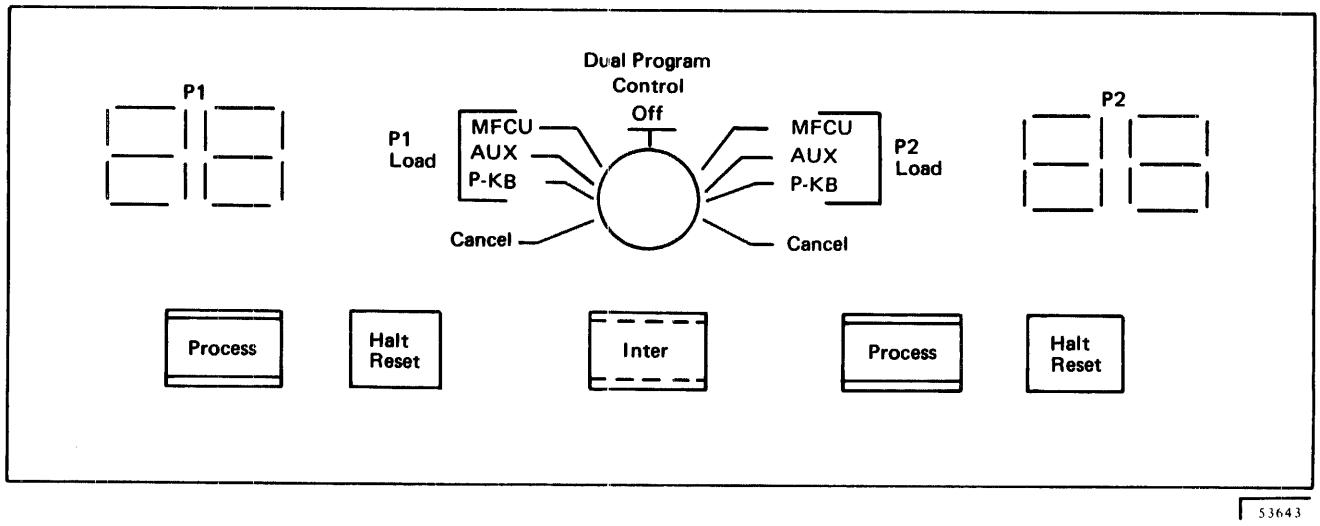
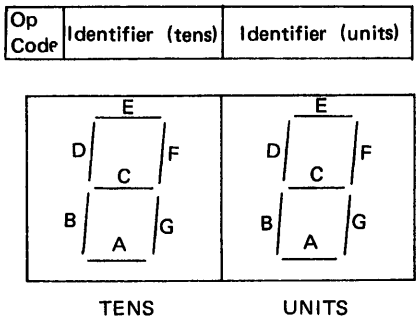


Figure 6-6. Dual Program Controls



Identifier	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Tens	Reserved	Ind A	Ind B	Ind C	Ind D	Ind E	Ind F	Ind G
Units	Reserved	Ind A	Ind B	Ind C	Ind D	Ind E	Ind F	Ind G

Figure 6-7. Message Display Unit

Power On/Off Switch

This switch initiates the power on/off sequence of the system. A system reset is performed as part of the power on/off sequence. Main storage data may be lost when power is dropped.

Program Load Key

This key is pressed to start initial program loading from the MFCU or Disk file. The I/O device is selected with the program load selector switch. A system reset is performed as part of the program load sequence.

Pressing the program load key allows the first card or record from the MFCU or Disk File to be read and stored in main storage, beginning at location 0000. When the key is released, the CPU proceeds to execute the instruction sequence starting at location 0000. Normal program load from the MFCU is executed through the primary hopper.

Should the I/O device selected be not ready, the console I/O attention light will come on when the program load key is pressed. Normally, to complete the program load function, it is only necessary to ready the device.

Stop Key/Light

This key is pressed to cause a processor stop.

The processor is stopped at the end of the operation in progress when the key was pressed. I/O data transfers are completed without loss of information. Processor stop turns on the stop light.

The processor may be restarted without loss of information by depressing the start key.

Start Key

This key is pressed to start or continue operation.

Pressing the start key turns off the stop light and allows the processor to continue its normal operation.

The start key is also used, in conjunction with the CE modes of operation, to start and/or advance the processor clock.

On systems without the dual program feature, pressing the start key clears the message display unit and allows the program to proceed after a halt operation.

Thermal Light

This light, along with the power check light, is turned on whenever an overtemperature condition is sensed in the CPU mainframe or electronic board in the printer. The light remains on until the condition is corrected and the power switch is turned to the off position. Overtemperature also results in power down. Normal power on can be performed after correcting the overtemperature condition. Figure 6-8 shows power check/thermal indications.

Lamp Test Switch

This switch turns on all system display lights.

Console Interrupt Key/Light (Dual Program Feature)

The console interrupt key is pressed to request immediate investigation of certain external conditions. The operation of the interrupt key is effective only when the interrupt light is on prior to pressing the key.

Pressing this key causes the normal operation to halt and to be replaced by an interrupt handling routine for interrupt level 0. Normal operation will be resumed after the interrupt routine signals completion of interrupt servicing with the SIO instruction to reset interrupt request zero.

This light is on only when the system is being used in the dual program mode and interrupt level zero is enabled (console interrupt will be recognized).

Selection of whether the system is to be used in the dedicated or the dual program mode is accomplished with SIO instruction. The SIO instruction is also used to enable or disable the use of interrupt level zero.

Dual Program Control Switch

This rotary switch is normally used in conjunction with the console interrupt key. The status of this switch is sampled with a TIO instruction.

POWER CHECK/THERMAL INDICATIONS

FAULT	POWER ON/ OFF SWITCH	INDICATORS		ACTION
		POWER CHECK	THERMAL	
Internal Power Supply Malfunction	On	On	Off	1. Turn power switch to OFF 2. Correct problem 3. Press Check Reset 4. Turn power ON
Thermal Condition	On	On	On	1. Turn power switch to OFF 2. Power check indicator goes off 3. Thermal light stays on until condition is removed
Customer Power Source Loss	On	On	On	1. Turn power switch to OFF 2. All indicators turn OFF 3. Turn power switch to ON and continue operation
Emergency Power Off (EPO) Activated	On	Off	Off	1. Turn power switch to OFF 2. Correct problem 3. Restore EPO interlock 4. Turn power switch to ON

Figure 6-8. Power Check/Thermal Indications

Halt Reset Key (P1, P2)

This key (one per program level) is pressed to take a specific program out of its programmed halt state.

Pressing the halt reset key clears the appropriate message display and allows the corresponding program to continue its normal operation.

Process Light (P1, P2)

These lights indicate the program level being executed or, in the case of an interrupt level servicing, the program level associated with the XR1, XR2, and PSR registers in use.

CE Controls

Address/Data Switches

These four switches are used to set up addresses or data. An address (16 bits) can be loaded into the storage address register (SAR). Data (8 or 16 bits) can be entered into main storage. Alter storage mode enters into storage, the data set up in the two rightmost switches. A sense I/O instruction with a Q code of hex 0/0 senses all four console switches. The data from the two rightmost switches is stored at the address specified by the control code and data from the two leftmost switches will be stored at the specified address minus one.

CE Key Switch

This switch when in the CE position, prevents the customer use meter from running.

Note: A processor check may occur if the switch position is changed while the clock is running.

CE Mode Selector

This rotary switch selects one of three processor operating modes: (1) normal process mode; (2) the step mode; or (3) test mode. Process is the mode for normal system operation.

Note: The CPU should be in a halt state before changing the position of the switch to prevent a processor check.

1. In the step mode, the rotary switch setting controls the manner in which the processor performs the stored program.

   a. Instruction step. Each time the start key is pressed and released, one complete instruction is performed. The I-phase is performed while the key is pressed, and the E-phase, if any, when it is released.

   b. Machine cycle step. Each time the start key is pressed and released, the instruction is advanced through one machine cycle. Pressing the key causes data in storage to be accessed, modified as required, and the result to be displayed in the arithmetic and logical unit (ALU) indicators of the console display. Upon release of the key, depending upon the operation being performed, either the old data or the new result is written back into storage.

   c. Clock step. Each time the start key is pressed, the clock advances through an odd-numbered clock, and each release, through an even-numbered clock.

2. The switch settings under the test mode permit the following:

   a. Alter SAR. The address, set up in the address/data switches, is transferred into SAR by the start key via the current IAR (P1, P2, or interrupt level).

   b. Alter storage. Pressing the start key allows transfer of the data set up in the rightmost two address/data switches, into the A register. Releasing the key causes this data to be placed in the storage location specified by SAR and into the Q register.

   c. Display storage. The contents of the storage location specified by SAR are transferred into the B register when the start key is pressed. These contents are rewritten into storage when the key is released, and are also transferred into the Q register.

LSR Display Selector

This rotary switch selects the local store register (LSR) to be displayed in position 2 of the display switch.

The LSRs that can be displayed with this switch are: IAR, ARR, XR1, and XR2. The selected LSR is displayed whenever the CPU is in a WAIT state.

When in the normal position, the LSR displayed is the one in use by the program.

Refer to FEMM for the procedure to display other LSRs.

System Reset Key

When the system reset key is pressed, the system enters an immediate 'idle' state. CPU registers, controls, and status indicators are reset and the processor clock is allowed to 'idle'.

Program Level 1 Instruction Address Register (P1-IAR) and Program Status Register (P1-PSR) are both reset to zero by a system reset.

The system must be in the PROCESS mode of operation for the pushbutton to the operative. After power on, the system reset key should be pressed prior to any CE operation. (Load key will also perform system reset.)

FEMD 5-220, 5-222, and 5-224 show the timing and circuitry of system reset.

Check Reset Key

This key is pressed to cause a reset of the Processor and/or Input/Output check conditions.

A check reset removes the current error conditions, thus allowing the processor to resume its operation after the Start key is pressed.

Storage Test Switch

This switch enables the altering or displaying of storage as follows:

1. In the step position, a storage location is accessed each time the start key is pressed.
2. In the run position, pressing the start key causes core storage to loop on either the same location repetitively or all of core sequentially. (See Address Increment Switch.)

Address Increment Switch

This switch enables address incrementing when in the CE test modes of alter or display storage. With the switch in the on position, the contents of SAR are incremented by 1 after each storage access. When the switch is in the off position, SAR is not incremented.

Address Compare Switch

This switch allows stopping the system when the setting of the address/data switches matches the register display. The register display must be positioned to SAR.

With the switch in the run position, comparison of address switches to SAR via the register display is performed, but no processor stop is initiated when a match occurs. The matched signal is provided as a scope sync point.

When the switch is in the stop position, a match of the address switches and the register display results in a processor stop at the completion of the storage read/write cycle. If, however, an SIO has been issued to some I/O device, that operation will be completed.

The processor is restarted by pressing the start key.

I/O Overlap Switch

This switch controls the system so that I/O operations may be executed in either an overlap or a non-overlap mode. With the switch in the normal on position, I/O operations are executed in an overlap mode. When the switch is in the off position, I/O operation is completed prior to execution of the next sequential instruction (non-overlap).

I/O Check Switch

This switch forces the processor to come to an immediate stop on an I/O error. The switch is normally set to run. With the switch set to stop, the processor stops on an I/O error and the console display indicates the processor status at the time the error stop occurred.

A check reset followed by the start key allows the program to continue.

Parity Switch

This switch allows processor parity errors to be ignored.

The switch is normally set to stop. This causes the processor to come to an immediate stop whenever a parity error is detected. A check reset followed by the start key allows the program to continue. With the parity switch in the run position, parity errors are detected and displayed, but the processor is not stopped.

Address Compare Light

This light is on whenever the register display is positioned to SAR and the address/data switches match the contents of SAR. The system will not stop unless the address compare switch is in the stop position.

I/O Check Light

This light is turned on when certain I/O errors (read check, punch check, etc.) are detected by an addressed I/O device. It is turned off when a system reset occurs, the check reset key is activated, or at the next SIO.

P1 and P2 Toggles

These two switches enable the CE to control selection of program level 1 or 2 to manually select the dual program mode of operation.

With P1 on and P2 off, the system operates in program level 1.

With P2 on and P1 off, the system operates in program level 2.

With both P1 and P2 off, the system is automatically enabled for the dual program mode of operation, with program level 1 being considered as the primary level.

For normal system operation, both P1 and P2 must be ON.

Note: An interrupt level 0 request is not accepted if either (but not both) P1 or P2 is turned off.

Console Display

The console displays are separated into two groups: (1) register display unit and (2) controls display section.

Register Display Unit

The register display unit (Figure 6-9) consists of a row of twenty lights and eight legend strips mounted on an eight position roller switch. At any one time, only one of the eight strips is visible through a cutout in the console above the row of lights. The legend strip and the corresponding register displayed by the row of lights are selected with the eight position switch.

Each legend strip by number is as follows:

1. SAR HI/SAR LO. Contents of storage address register high and low.

2. LSR HI/LSR LO. Contents of LSR selected by the LSR display selector.

3. OP REG. Contents of the op register.
   Q REG. Contents of the Q register.

4. B REG. Contents of the B register.
   ALU CTL. The state of the following ALU controls is displayed:
   DIG CAR (Digital Carry)
   DEC (Decimal Instruction)
   RE COMP (Recomplement)
   ADD (Addition)
   SUB (Subtraction)
   TEMP CAR (Temporary Carry)
   AND
   OR

5. A REG. Contents of the A register.
   ALU OUT. Contents of the output of the ALU.

6. COND REG. The contents of the condition register is displayed as follows:
   BIN OVF (Binary Overflow)
   TF (Test False)
   DEC OVF (Decimal Overflow)
   HI (High)
   LO (Low)
   EQ (Equal)

7. CS ASNMT. Cycle steal assignment is displayed as it is presented to the I/O devices on the I/O interface.
   INT LEV. Interrupt level indicates which I/O device is interrupting the program.

8. PROC CHK. The processor checks are displayed as follows:
   I/O LSR - indicates selection of an LSR by an I/O device was not performed correctly.
   LSR F1 - Parity is incorrect on the output of the LSR Feature 1.
   LSR F2 - Parity is incorrect on the output of the LSR Feature 2.
   LSR HI - Parity is incorrect on the output of an LSR high.
   LSR LO - Parity is incorrect on the output of an LSR low.
   SAR HI - Parity is incorrect in the storage address register high.
   SAR LO - Parity is incorrect in the storage address register low.
   INV ADDR - indicates that the SAR contains an invalid address.
   SDR - Parity is incorrect in the storage data register.
   CAR - indicates the carry out of the ALU is incorrect.
   DBI - Parity is incorrect on the CPU end of the data bus in coming from the I/O devices.
   A/B - Parity is incorrect in the A register or B register.
   ALU - Parity is incorrect at the output of the ALU.
   CPU DBO - Parity is incorrect on the CPU end of the data bus out to the I/O devices.
   OP/Q - Parity is incorrect in the op register or Q register.
   INV OP - indicates an invalid op code in the op register.
   CHAN DBO - Parity is incorrect on the I/O device end of the data bus out from the CPU.
   INV Q - indicates an invalid Q byte is present in an I/O instruction.

Controls Display Section

1. *Machine Cycles:* Twelve indicator lamps represent the twelve machine cycles. They identify the processor cycle just completed in all modes, except the CE clock step mode, in which case, they indicate the cycle in progress.

2. *Clock:* Ten indicator lamps represent clocks 0 through 9 which can be stepped through in the CE clock step mode. In the normal process mode, a machine cycle consists of clocks 0 through 8. Clock 9 is used with the CE step and test modes.

3. *Interrupt:* A single indicator lamp is used to monitor whether any interrupt level is being serviced.

1	SAR HI	P	0	1	2	3	4	5	6	7	P	0	1	2	3	4	5	6	7	SAR LO
2	LSR HI	P	0	1	2	3	4	5	6	7	P	0	1	2	3	4	5	6	7	LSR LO
3	OP REG	P	0	1	2	3	4	5	6	7	P	0	1	2	3	4	5	6	7	Q REG
4	B REG	P	0	1	2	3	4	5	6	7		DIG CAR	DEC	RE COMP	ADD	SUB	TEMP CAR	AND	OR	ALU CTL
5	A REG	P	0	1	2	3	4	5	6	7	P	0	1	2	3	4	5	6	7	ALU OUT
6											P			BIN OVF	TF	DEC OVF	HI	LO	EQ	COND REG
7	CSASNMT	P	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7	INT LEV
8	PROC CHK	I/O LSR	LSR F1	LSR F2	LSR HI	LSR LO	SAR HI	SAR LO	INV ADDR	SDR	CAR	DBI	A/B	ALU	CPU DBO	OP/Q	INV OP	CHAN DBO	INV Q	PROC CHK

Figure 6-9. Register Display Unit

Section 2. Maintenance Features

Refer to MAP charts for maintenance approach.

Appendix A. Unit Characteristics

For machine characteristics refer to the *IBM System/3 Installation Manual–Physical Planning,* GA21-9084.

READER'S COMMENT FORM

IBM System/3
5410 Processing Unit
Field Engineering
Theory of Operation Manual

SY31-0207-1

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. Comments and suggestions become the property of IBM.

	Yes	No
• Does this publication meet your needs?	☐	☐
• Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

- What is your occupation? _____
- How do you use this publication?

As an introduction to the subject?	☐	As an instructor in a class?	☐
For advanced knowledge of the subject?	☐	As a student in a class?	☐
For information about operating procedures?	☐	As a reference manual?	☐

Other _____

- Please give specific page and line references with your comments when appropriate. If you wish a reply, be sure to include your name and address.

COMMENTS:

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS, PLEASE. . .

Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold

Fold

FIRST CLASS
PERMIT NO. 387
ROCHESTER, MINN.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
General Systems Division
Development Laboratory
Rochester, Minnesota 55901

Attention: Product Publications, Dept. 245

Fold

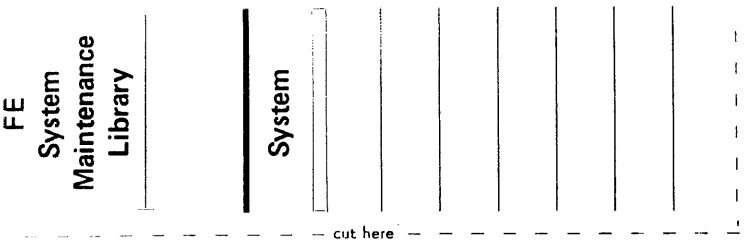
Fold

IBM

International Business Machines Corporation
Field Engineering Division
112 East Post Road, White Plains, N. Y. 10601

Cut Along Line

IBM System/3 Printed in USA SY31-0207-1



SY31-0207-1

IBM System/3 Printed in USA SY31-0207-1

IBM

International Business Machines Corporation
Field Engineering Division
112 East Post Road, White Plains, N.Y. 10601