IBM Field Engineering

Manual of Instruction

2075 Processing Unit -- Volume 1

Comprehensive Introduction

Functional Units

223-2872-1

Form 223-2872-1
FES S26-7033

PREFACE

This is one of six Field Engineering manuals for the 2075 Processing Unit. These six manuals contain the unit theory of operation, reference diagrams to be used when troubleshooting, and maintenance procedures.

A basic knowledge of the IBM System/360 contained in the IBM System/360 Principles of Operation, Form A22-6821 is considered a prerequisite for studying the unit theory of operation. The theory of operation is contained in a four volume manual identified as a Field Engineering Manual of Instruction (FEMI). Volume 1 is a prerequisite for the detailed information contained in volumes 2, 3, and 4. Volume 1 contains the introduction to the system and the processing unit and a description of the functional units (registers, adders, and decoders) of the processing unit. Volumes 2 and 3 contain detailed instruction analysis, and volume 4 contains detailed information on special features and power supplies and control.

The four volumes of theory of operation contain many references to the diagrams packaged in the associated Field Engineering Diagrams Manual (FEDM). All diagrams in the FEDM are identified by a four digit figure number and unless otherwise specified, all four digit figure references in the FEMI indicate that the figure is contained in the associated FEDM.

The complete titles and form numbers of the six 2075 Field Engineering Manuals are:

- 2075 Processing Unit--Volume 1, Comprehensive Introduction, Functional Units, Field Engineering Manual of Instruction, Form 223-2872
- 2075 Processing Unit--Volume 2, Theory of Operation: Storage Bus Control; Instruction Preparation; FLT, Logout, MCW; Interrupts, Field Engineering Manual of Instruction, Form 223-2873
- 2075 Processing Unit--Volume 3, Theory of Operation: Fixed Point, I Execute, Branch, Floating Point, Variable Field Length, Field Engineering Manual of Instruction, Form 223-2874
- 2075 Processing Unit--Volume 4, Special Features, Power Supply and Control, Appendix, Field Engineering Manual of Instruction, Form 223-2875
- 2075 Processing Unit, Field Engineering Diagrams Manual, Form 223-2876
- 2075 Processing Unit, Field Engineering Maintenance Manual, Form 223-2880

MAJOR REVISION (December 1965)

This edition, Form 223-2872-1, obsoletes Form 223-2872-0. The major change is the addition of Figures 10 and 63.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.
Address comments concerning the contents of this publication to:
IBM Systems Development Division, Product Publications, Dept. 520, CPO Box 120, Kingston, N. Y. 12401

ILLUSTRATIONS

SYSTEMS INTRODUCTION

- The processor is a number of independent units.
- CPU and core storage are integrated units.
- Overlapped cycles between four storage units increase the access speed.
- Each byte in processor storage is addressable.
- Data flow consists of 64 data bits and 8 parity bits in parallel.
- Operands are primarily based on the hexadecimal system.
- Instruction formats use both long and short data formats.
- Instruction cycles, execution cycles, and I/O operations may take place simultaneously.
- Cycle time is dependent upon the speed of the registers and logic.
- A high degree of reliability is obtained through fault location tests, parity checking, and logout.

The IBM System/360 Model 75 fills the need for a data processing system with high-speed operation, a versatile instruction set, large core storage capacity, and input/output capabilities for a wide range of applications. The Model 75 basic system (Figure 1) consists of a number of independent sections: the IBM 2365 Processor Storage, IBM 2075 Processing Unit and Bus Control Unit, IBM 2860 Selector Channel, and associated control units and input/output devices.

The functional organization of the central processing unit is equivalent to that of any other IBM System/360; however, its higher performance is attributed to the following:

1. The storage word consists of eight bytes (64 data bits and 8 parity bits).
2. It uses a three input addressing adder capable of handling 24 bits in parallel.
3. Double word floating-point operations as well as single word and fixed-point arithmetic instructions are performed in parallel.
4. Instruction requests from core storage are overlapped with the execution cycle of the current instruction.

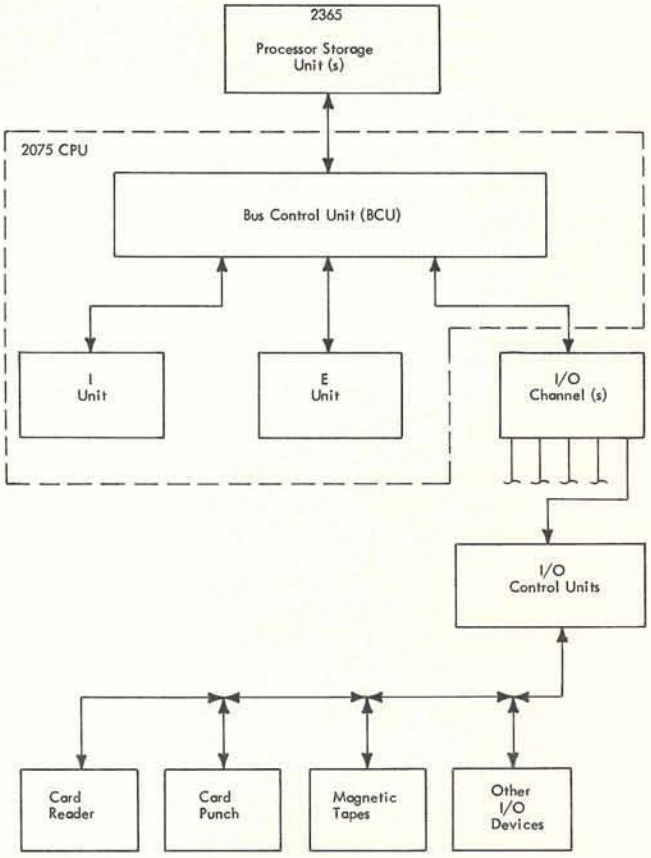


FIGURE 1. BLOCK DIAGRAM OF THE MODEL 75

The basic system consists of the Model 75 central processing unit and one 0.75-microsecond processor storage unit installed as a single integrated frame. In another configuration of processor storage, shown in Figure 2, the 2075 central

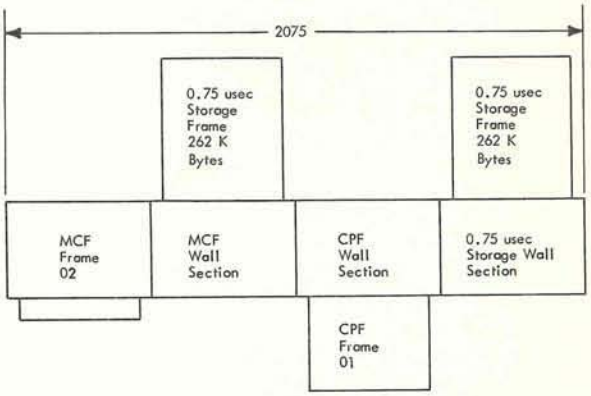


FIGURE 2. IBM SYSTEM/360, MODEL 75 CPU - STORAGE SYSTEMS

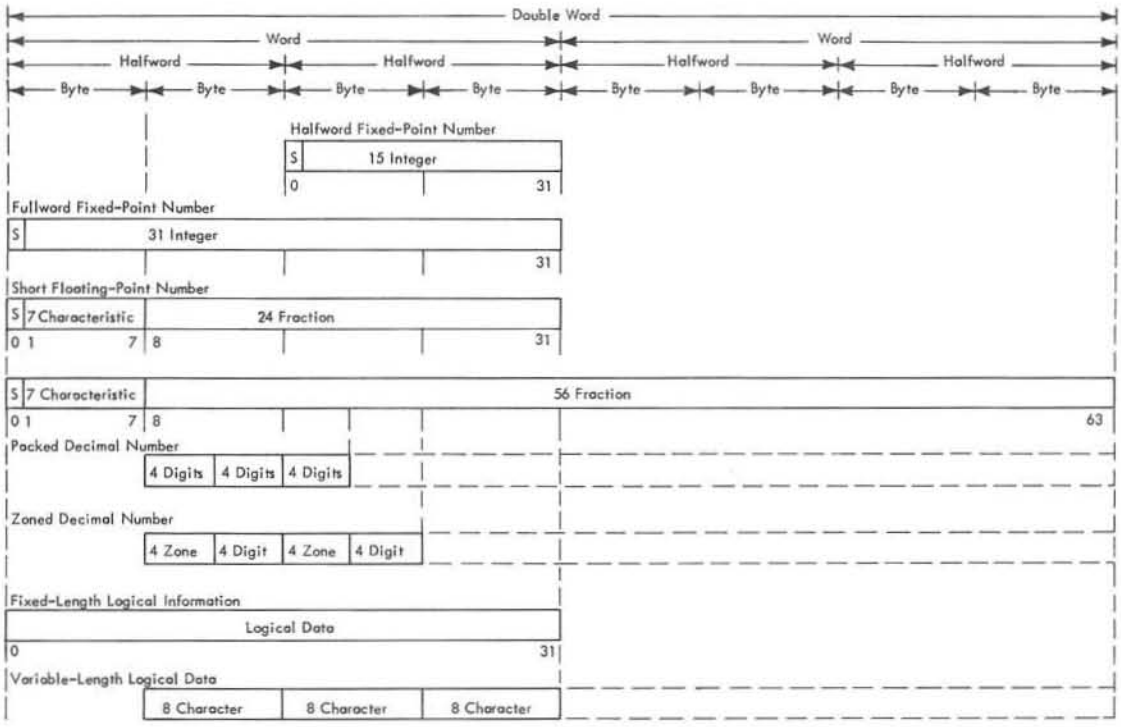


FIGURE 3. DATA FORMATS

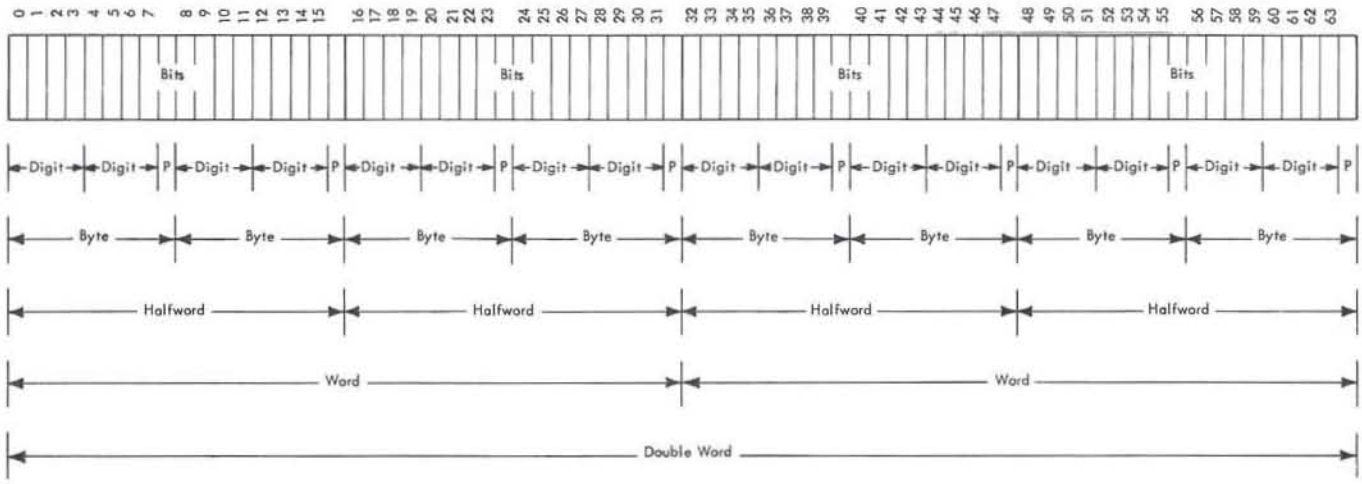


FIGURE 4. MODEL 75 INFORMATION RELATIONSHIPS

processing unit consists of a wall used to house the power conversion equipment and cables, the maintenance control frame (in line with the wall), the central processing frame (right angle to the wall), and the 2365 Processor Storage Units (right angle to the wall).

Overlapped cycles between two storage units increase the access speed. Addresses are staggered in the two units, and a series of requests for successive double words activates the two units alternately, thus doubling the maximum rate.

Each byte in the processor storage is individually addressable with a 24-bit binary address; however, instructions concerned with word boundaries must be addressed as such or error conditions result. The location of a stored field is specified by the address of the left-most byte of the field. Variable length fields may start on any byte location, but a fixed length field of two, four, or eight bytes must have an address that is a multiple of two, four, or eight, respectively. Some of the various alignment possibilities are shown in Figure 3.

The Model 75 data flow among the working registers, addressable registers, processor storage, AB registers, and main adder consists of 64 data bits and 8 parity bits in parallel. The working registers (J, K, L, and M) are not addressable but they play an important part in the overall operation of the Model 75 because they function as an intermediate storage for operands, partial products, factors, and multiples during instruction execution. At the completion of an instruction, the intermediate data in the working registers are transferred to storage, a floating-point register or a general register. If the data in the working registers are not stored by the end of the operation, the data are lost when the next instruction is executed.

The addressable registers are the 4 floating-point registers and the 16 general registers. Each register is addressable by a 4-bit binary address. In some operations an even-odd address pair of registers are used for addressing purposes. The floating-point registers are always addressed by their even address because they are double words. The general registers contain operands and indexes; the floating-point registers contain operands.

The AB registers hold prefetched instructions from the processor storage. These registers may obtain instructions directly from the storage bus out latch register or via the J register from the storage bus out latch register. From the AB register, the prefetched instructions are gated to the instruction decoders.

Operands are primarily based on the hexadecimal system. Four bits equal one digit; therefore, the value 0000 through 1111 in binary is possible. The binary, hexadecimal, decimal equivalents are shown in Figure 6. Some of the terms used in Model 75 work are as follows(Figure 4):

1. Bit: A bit is a single unit of information; its value is either a 0 or a 1.

2. Digit: A digit consists of four bits, in any combination of bits from binary 0000 through binary 1111.

3. Byte: A byte is two digits plus one parity bit. In most cases, when a byte is mentioned, the main concern is with the eight data bits. The parity bit is assumed to be transferred on the main data paths throughout the system, and it is used for valid byte detection throughout the system. Parity in the 2075 processing unit is odd.

4. Halfword: A halfword is 2 bytes, 4 digits, or 16 data bits plus 2 parity bits.

5. Word: A word, sometimes referred to as a full word, is two halfwords or 32 data bits plus 4 parity bits.

6. Double word: A double word, sometimes referred to as a core storage word, is two words or 64 data bits plus 8 parity bits.

The Model 75 uses several instruction and data formats. Included in the standard instruction formats are the RR (register-to-register), RS (register and storage), RX (register and indexed storage operation), SI (storage and immediate operand), and SS (storage-to-storage) formats shown in Figure 5. In each format, the first instruction halfword consists of two parts: the first byte contains the operation code, and the second byte is either two 4-bit fields or one 8-bit field.

The length and format of an instruction are indicated by the first two digits of the operation code as follows:

00	RR format
01	RX format
10	RS and SI formats
11	SS format

The second byte is specified from among the following:

1. Four-bit operand register designators (R1, R2, and R3).
2. Four-bit index register designator (X2).
3. Four-bit mask (M1).
4. Four-bit field length specification (L1 or L2).
5. Eight-bit field length specification (L).
6. Eight-bit byte of intermediate data (I2).

The second and third halfwords each specify a 4-bit base register designator (B), followed by a 12-bit displacement (D). An effective storage address (E) is a 24-bit binary integer given, in the typical case, by

$$E = B + X + D$$

where B and X are 24-bit integers from general registers identified by fields B and X, respectively,

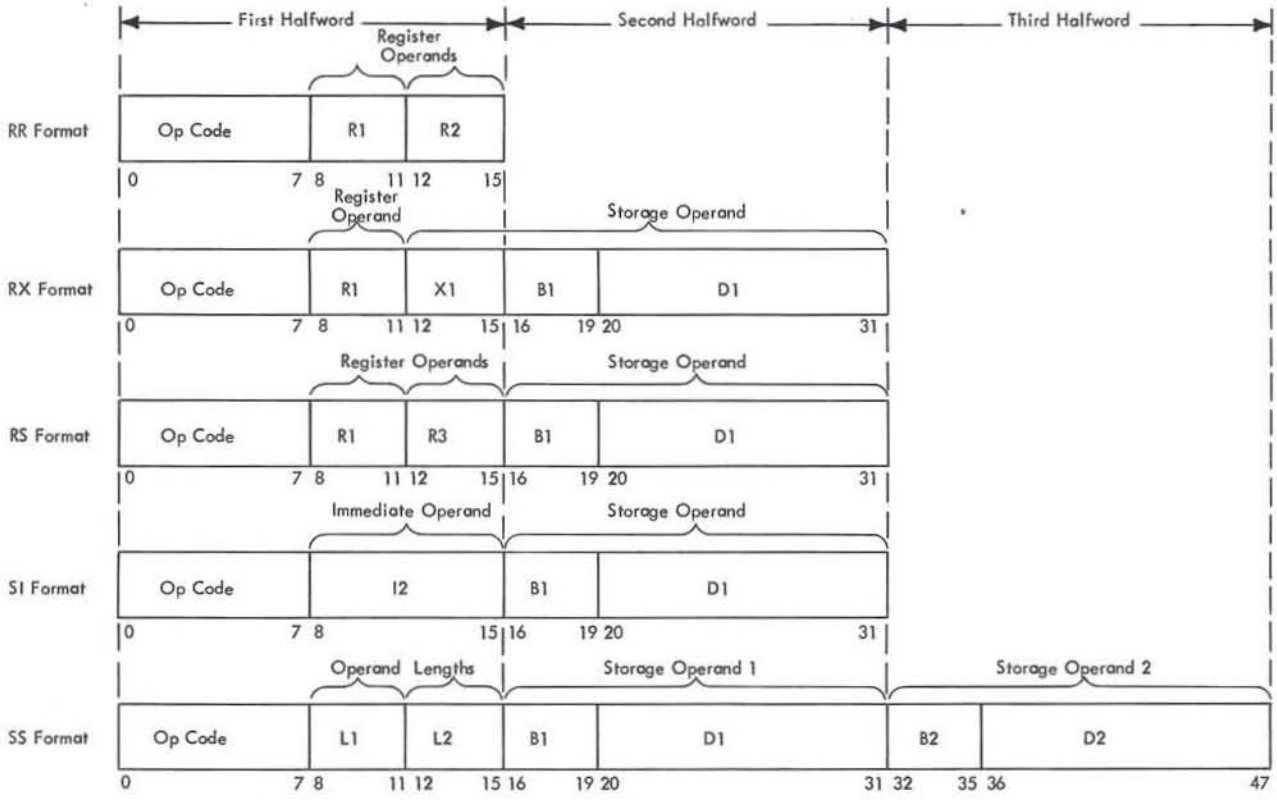


FIGURE 5. FIVE BASIC INSTRUCTION FORMATS

and the displacement (D) is a 12-bit integer contained in every instruction that references storage.

The base (B) can be used for static relocation of programs and data. In record processing, the base can identify a record; in array calculations, it can specify the location of an array. The index (X) can provide the relative address of an element within an array. Together, B and X permit double indexing in array processing.

The displacement provides for relative addressing of up to 4095 bytes beyond the element or base address. In array calculations, the displacement can identify one of many items associated with an element. Thus, multiple arrays whose indices move together are best stored in an interleaved manner. In the processing of records, the displacement can identify items within a record.

In forming an effective address, the base and index are treated as unsigned 24-bit positive binary integers and the displacement as a 12-bit positive binary integer. The three are added as a 24-bit binary number, ignoring overflow. Since every address is formed with the aid of a base address, programs are readily relocated by changing the contents of base registers.

A zero base or index designator implies that a zero quantity must be used in forming the address, regardless of the contents of general register 0. A displacement of zero has no special significance. Initialization, modification, and testing of bases and indices can be carried out by fixed-point instructions, by branch and link, branch on count, or branch on index instructions. Load effective address provides not only a convenient housekeeping operation, but alos, when the same register is specified for result

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

FIGURE 6. BINARY, HEXADECIMAL, DECIMAL EQUIVALENTS

and operand, an immediate register-incrementing operation.

Several data formats are shown in Figure 3. An 8-bit unit of information (the byte) is fundamental to most of the formats. A consecutive group of N such units constitutes a field of length N. Fixed-length fields of length one, two, four, and eight are termed bytes, halfwords, words, and double words, respectively. In many instructions, the operation code implies one of these four fields as the length of the operands. On the other hand, the length is explicit in an instruction that refers to operands of variable length.

Data are entered into the Model 75 system through input/output devices attached to the input/output selector channel via control units. Figure 7 shows a block diagram of data input from the card reader. The data sensed at the brushes in the card reader are assembled into bytes in the control unit. Once a byte is assembled, it is transferred to the B register in the selector channel. In the selector channel, the bytes are assembled into double words. When the B register contains an assembled word, it is transferred to the A register. A new double word is assembled in the B register while the double word in the A register is transferred to the memory data register in the core storage unit. Once the data are placed in the memory data register, they are written into the specified address in the core array.

An output operation (Figure 8) is the opposite of an input operation. The data are read from the specified address into the memory data register. From the memory data register, the data are transferred to the A register in the selector channel. When the B register has transferred the last byte of data of the previous core storage word to the byte register in the control unit, the transfer from the A register to the B register occurs. The data, in the B register, are disassembled into bytes for transmission to the byte register in the control unit. From the byte register in the control unit, the byte is disassembled into bits and punched into the card.

Although this is a general description of an input/output operation involving the selector channel, input/output control unit, and a card read/punch, all input/output operations are basically the same whether they are slow-speed devices such as telecommunication equipment or high-speed devices such as hypertape units; input/output operations, instruction cycles, and execution cycles may take place simultaneously for maximum processing speed.

The fundamental determinant of the central processing unit speed is the time required to take data from the internal registers, process the data through the adder or other logical unit, and return the result to a register. This cycle time is determined by the delay per logical circuit level and the number of levels in the register-to-adder path, the adder, and the adder-to-register return path.

The Model 75 processor has a 64-bit parallel data flow with parity checking on each 8-bit byte. The parity checked main adder operates on a basic cycle time of 200 nanoseconds (0.2 microseconds), and the nominal delay per logic level in the Model 75 is 6 nanoseconds. The basic time cycle of the Model 75 is 200 nanoseconds.

The speed of the central processing unit also depends on the speed of the general registers and the floating-point registers. The general and floating-point registers are implemented with 6-nanosecond logic circuits and communicate directly with the adder and other data paths.

The two principal measures of size in the central processing unit are the width of the data paths and the number of bytes of high-speed working registers. The Model 75 has a 64-bit (data) main adder, an 8-bit exponent adder, an 8-bit decimal adder, a 24-bit addressing adder, and several other data transfer paths, some of which have incrementing ability. In addition to the 16 general purpose registers, there are 4 floating-point registers, 6 working registers, and a 64-bit program status word register.

In the Model 75, many operations can take place at the same time. The central processing unit is divided into three units that operate somewhat independently. The instruction preparation unit (I unit) requests instructions from main storage, prepares them by computing their effective address, and initiates the request for the required data. The execution unit (E unit) performs the execution of the instruction prepared by the instruction unit. The third unit, the storage bus control unit (BCU), coordinates the various requests by the other units and by the channels for core storage cycles. All three units normally operate simultaneously, and together they provide a large degree of instruction overlap. Since each of the units contains a number of different data paths, several data transfers may occur during any one single cycle in a single unit.

Standard features of the Model 75 include single and double word floating-point arithmetic, fixed-point arithmetic, decimal arithmetic, variable field length (VFL) operations, logical and masking operations, storage protection, interval timer, direct control features for external control features for external control signal transmission, and automatic maintenance features such as fault locating tests (FLT), logout, and a comprehensive interrupt recovery system.

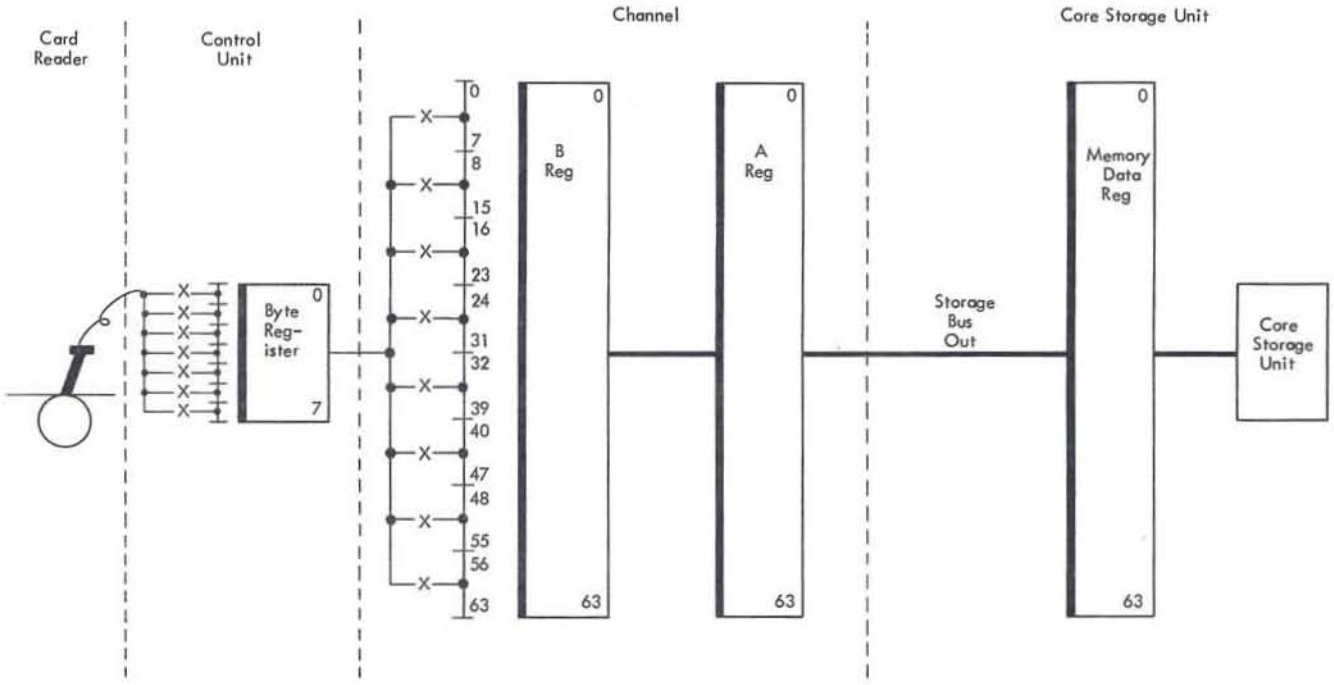


FIGURE 7. INPUT/OUTPUT CHANNEL (INPUT) OPERATION

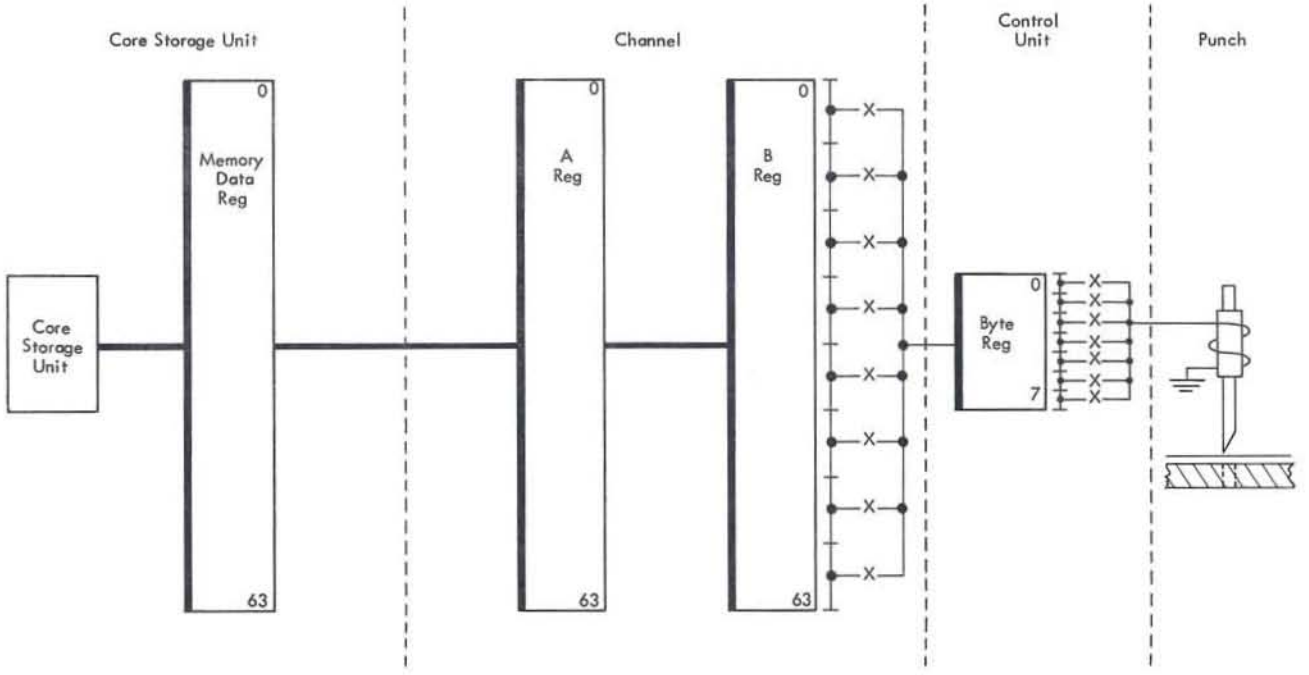


FIGURE 8. INPUT/OUTPUT CHANNEL (OUTPUT) OPERATION

2075 PROCESSOR UNIT

- Simultaneous I and E cycles are possible in CPU.
- I and E units perform interlocked overlapped operations.
- Instruction buffering is provided for greater speed.

The Model 75 has a 64-bit main adder, 8-bit exponent adder, 8-bit decimal adder, 24-bit addressing adder and several other data transfer paths. There are 16 general purpose registers, 4 floating-point registers, 6 working registers, and a program status word register.

The Model 75 has simultaneous I and E cycles. The central processing unit is divided into three units which operate somewhat independently. They are the instruction unit (I unit), the execution unit (E unit), and the bus control unit (BCU). As previously discussed, all three units normally operate simultaneously, and together they provide a large degree of instruction overlap. Since the instruction and execution units contain a number of different data paths, several data transfers may occur during any one cycle in a single unit.

The I and E units are interlocked to prevent one unit overrunning the other and to allow proper interrupt recovery. Some instructions, such as input/output instructions, are executed entirely by the I unit, while multiple load and store and set system mask instructions are executed by both the I and E units.

In addition to the overlapping of the I and E cycles, speed is also obtained by using instruction and operand buffering. Two double word registers (16 bytes) are used for prefetching instructions and one double word register (8 bytes) is used for prefetching operands. Prefetching allows most of the core storage access cycle to be overlapped by instruction execution.

I-Unit Controls and Data Flow

- Main functions are controlled by the clock.
- Instructions are placed temporarily in the AB registers.
- Effective Address (E) = X + B + D.

The I unit and the E unit functions are controlled by the clock. The main function of the clock is to generate, shape, and distribute clock pulses to the I unit, E unit and maintenance console. Two types of pulses are generated by the clock: running pulses and controlled pulses. Running pulses are stopped during system reset; controlled pulses are stopped during system reset, machine check, during all resets, and in single-cycle mode. In single-cycle mode, controlled pulses are emitted one cycle at a time by depressing the single-cycle key.

The relationship among the clock pulses is shown in Figure 9. The basic clock pulse (the A pulse, also called A clock) is wide enough to reliably set a control trigger; its leading edge also defines the beginning of a machine cycle.

B cycles are approximately 1/4 cycle long and are generated at three different times within a machine cycle: early B clock is generated for the second quarter of the machine cycle, B clock is generated for the third quarter of the machine cycle, and late B clock is generated for the fourth quarter. Each B clock pulse is timed for its individual use; therefore, these times are not exact.

The L clock pulse is slightly longer than the A clock pulse and brackets the A clock pulse on both sides. The leading edge of an L clock pulse rises in the preceding A clock cycle and its trailing edge falls after the A clock pulse. L clock pulse sets the latch to insure the state of the control is held in the latch for the duration of the L clock even though the A clock is changing the status of the control trigger.

The C clock cycle is one-half cycle long and spans the first half of the machine cycle. It is used for special purposes in the bus control unit and maintenance console. The A, B, C, and L pulses are generated as controlled and running pulses.

The heart of the clock is a crystal controlled oscillator which produces all clock pulses. A provision is also made to gate a fixed or variable frequency oscillator into the pulse shaping and timing circuitry. The switch "enable frequency check" determines the oscillator being used.

Following the oscillator gate is a circuit that allows every other pulse from the oscillator to enter the clock controls and pulse shaping circuits. The shaped oscillator output feeds a delay line and two AND circuits which are gated by the control trigger and running trigger respectively. The delay line insures that the first pulse emitted is a complete pulse. From here the pulses are sent to pulse stretchers and another delay line to form the L and C clock pulses as well as separating the leading edges of the A and L clocks.

The B clock is formed from a L pulse by means of a pulse shrinker and then fed into three different delay lines to form early B, B, and late B clocks. Each clock pulse leaving the large circuit boards has its own delay line, which is adjusted so that all

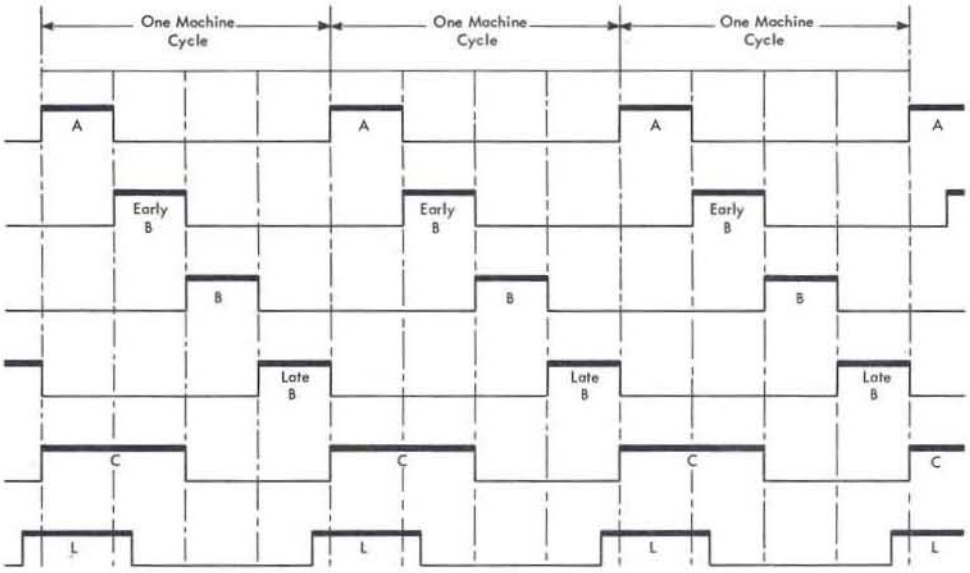


FIGURE 9. CLOCK PULSE RELATIONSHIPS

pulses of the same kind (A, B, C, or L) arrive at every portion of the machine at the same time.

The data flow in the I unit is shown in the IBM 2075 Processing Unit Field Engineering Diagrams Manual, Form 223-2876. See Figure 2000. The instructions are brought from core storage and placed temporarily in the AB registers. Upon the completion of the I time of the previous instruction, the next instruction is transferred into the I operation register. If the next instruction time is not blocked, the next operation is decoded. During I time the instruction placed in the instruction operation register is decoded to determine the operation to be performed. The op code and R1 fields are placed in the B operation (BOP) register. The B operation register is an operation register for those instructions executed by the I unit. The general registers specified by the X and B fields are decoded and the contents of these registers are added to the contents of the displacement (D) field in the addressing adder to calculate the effective address (E). The effective address is gated from the addressing adder to the storage address register (SAR) or the H register. If the effective address specifies a location in core storage, the contents of the storage address register is transferred to the memory address register (MAR) when the storage request is recognized by the bus control unit (BCU). The data from core storage (or the general purpose register or floating-point register specified by R1) is returned to the J register. By the end of instruction time the instruction is decoded, the operands are in the working registers, (if the instruction is an RR format) the instruction counter is updated, and the next instruction is waiting in the AB registers.

E-Unit Controls and Data Flow

- There are many main data paths in the execution unit.
- Binary arithmetic operations fall into four classes.
- Binary addition-subtraction is accomplished by parallel addition.
- Binary multiplication is performed at the rate of four bits per cycle.
- Binary division uses a non-restoring division algorithm.
- VFL and logical operations process fields of fixed and variable length.
- Instruction sequencing is normally in sequential order.
- Instruction counter controls control the advancing of the instruction counter.
- Branch requests are made in the same way as an operand request.

The data flow in the execution (E) unit of the Model 75 is shown in Figure 10. There are many main data paths in the E unit: a typical one is one in which the main adder is used. Data is gated from one or more of the working registers (J, K, L, and M register) to the main adder (AM); the main adder output bus (AMOB) latches are gated back to

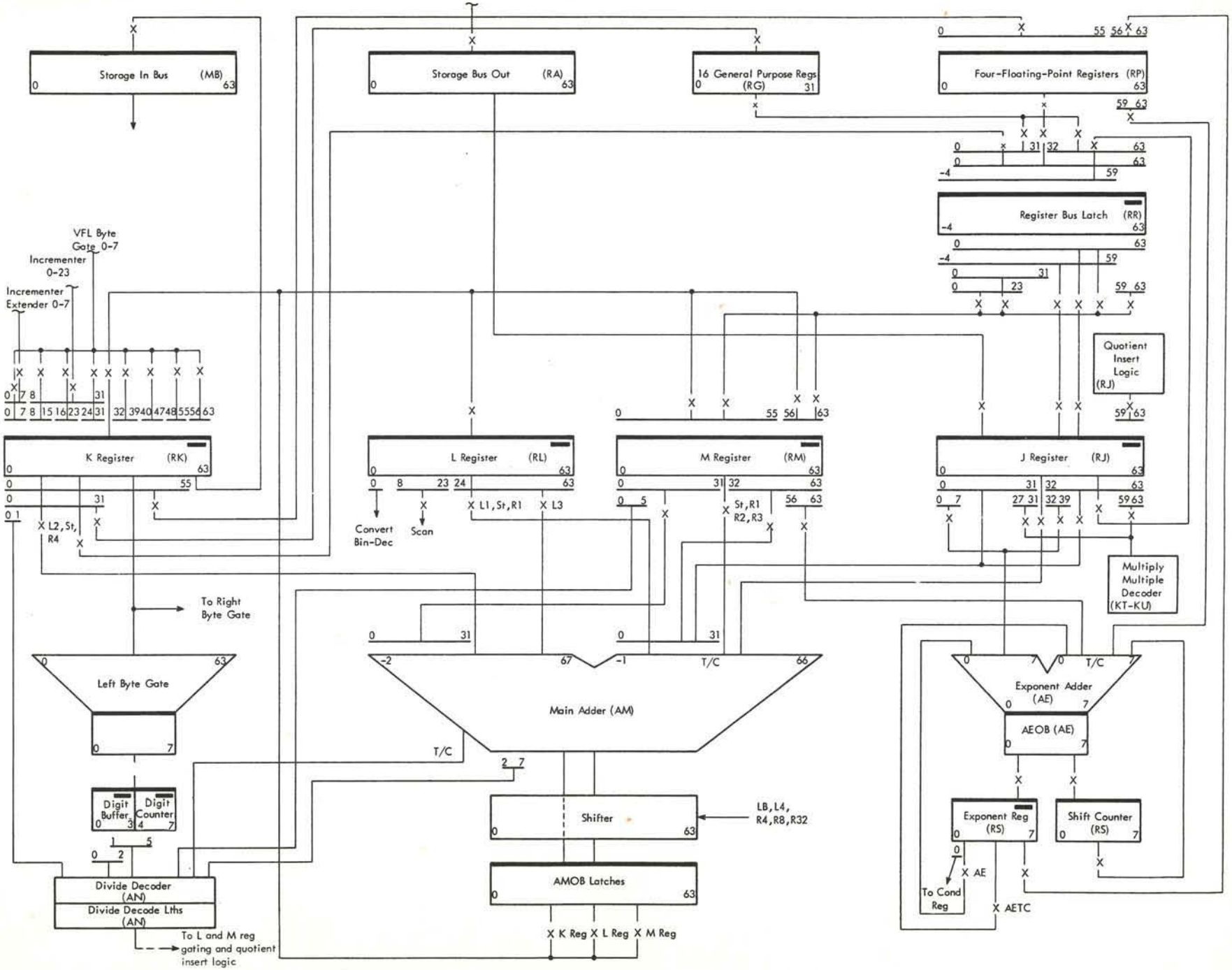


FIGURE 10. SIMPLIFIED E-UNIT DATA FLOW

one or more of the working registers. A typical floating-point data path is one in which the M register is gated to the true/complement (T/C) input of the main adder, and the K register is gated to the normal input. The sum on the main adder out bus is gated to the K and M registers.

Another data path is the one that is used in expanding a half-word from core storage into a full word. The data flow is from the J register through the main adder, the shifter, the main adder output bus latches, and to the K register. From the K register the data path is to the register bus latch (RBL) and then back to the J register. System/360 Model 75 binary arithmetic operations fall into four classes: fixed-point arithmetic, floating-point arithmetic, logical operations, and decimal arithmetic. The basic arithmetic operand is a 32-bit, fixed-point binary word. Some products and all dividends are 64 bits long, using an even-odd register pair.

Addition, subtraction, multiplication, divisions, and comparisons take one operand from a register and the other operand either from a register or core storage. Floating-point numbers may occur in either of two fixed-length formats -- short or long. These formats differ only in the length of the fractions, as indicated in Figure 3. The fraction of a floating-point number is expressed in 4-bit hexadecimal (base 16) digits. In the short format, the fraction has 6 hexadecimal digits, in the long format, the fraction has 14 hexadecimal digits. The radix point of the fraction is assumed to be immediately to the left of the high-order fraction digit. To provide the proper magnitude for the floating-point number, the fraction is multiplied by a power of 16. The characteristic (exponent), bits 1-7 of both formats, indicates this power. The characteristic is treated as an excess 64 number with a range from -64 through +63, and permits representation of decimal numbers with magnitudes in the range of 10^{-78} to 10^{75}. Bit position 0 in either format is the fraction sign. The fraction of negative numbers is carried in true form.

Floating-point operations are performed with one operand from a register and the other operand from either a register or core storage. The result, located in a floating-point register, is usually of the same length as the operands.

Binary Addition-Subtraction: In the Model 75, binary addition and subtraction is accomplished by parallel addition of the two operands. In RX format instructions, the storage operand is placed in the J register prior to the execution of the first good E cycle, and the R1 operand is brought either from the general register or floating-point register specified by R1 in the instruction and placed in the M register. If the instruction is a fixed-point instruction or a short floating-point instruction, the R1 operand is placed in the high-order half (bits 0-23, for floating-point or bits 0-31 for fixed-point) of the M register. The result is obtained in one pass through the main adder. The result is returned to the K register; from the K register the result is returned either to the floating-point or general register specified by R1 of the instruction.

Before the add cycle in floating-point add-subtract and compare instructions, the operand with the smaller exponent is shifted right until the exponents are equal. Addition of the fractions follows the shift cycles as in the fixed-point instructions. A guard digit is retained to increase the precision of the result following fraction addition. Subsequent cycles may be necessary to normalize the sum or recomplement the result in case the result is in complement form.

Tests are made during E time to detect exponent overflow, exponent underflow, addressing exceptions, specification, and lost significance for floating-point instructions, and addressing exceptions, specification, and overflow for fixed-point instructions.

Binary Multiplication: The Model 75 incorporates logic which allows binary multiplication at the rate of four-bits per cycle. All even multiples of the multiplicand are provided for addition to or subtraction from the partial product within one cycle, except for the X10 and X14 multiples, which require two cycles. Since only even multiples are used, an even multiplier digit yields a correct partial product; all odd multiplier digits yield a partial product which is the X1 multiple higher than the multiplicand. Correction for this over multiplication is provided for by decoding the unit bit of the next highest multiplier group. When an odd multiplier is anticipated, the X16 multiplicand is subtracted from the partial product. This results in the partial product being the X1 multiple low as the next multiplication cycle by an odd digit begins. The net result of this pair of cycles is a correct product if the units digit of the next multiplier group does not also contain a one bit. The first cycle of multiplication differs from the following cycles in that if the unit bit is odd, the multiplicand, located in the K register, is added to the multiple as though it is a partial product.

Fixed-point multiply is executed by placing the multiplicand in the M register and the multiplier in the J register. During the first cycle, the multiplicand is transferred from the M register to the main adder and back to the K register and the M register. The second cycle develops the X12 multiple by shifting and subtracting the X4 multiple

from the X16 multiple; the result (the X12 multiple) is placed in the L register. The shift counter is set to eight, and the multiply iterations are performed. The multiplier is decoded from the J register to select the proper multiple from the M register or L register. The partial product in the AMOB latches is returned to the K register, the shift counter is decremented, and the J register is shifted right four bits. When the shift counter is decremented to zero, multiplication is complete; the product is stored in the general register specified by R1 of the instruction.

Floating-point multiply is similar to fixed-point multiply; however, the multiplicand is prenormalized before the multiplication cycles and the sum of the exponents is determined. The multiple generation and iteration cycles are the same as for fixed-point multiply except that the shift counter is set to six for single-precision floating-point instructions or to 14 for double-precision floating-point instructions.

During each iteration cycle, the multiplier is tested for high order zeros; when detected, multiplication is concluded. The partial product is either hex-normalized or a high-order hex-zero digit exists. In the latter case, the hex-zero digit is shifted left four bits and one is subtracted from the exponent.

Floating-point multiplication is terminated by gating the final product from the K register into the floating-point register specified by R1 of the instruction.

Binary Division: In the Model 75, binary division uses a non-restoring division algorithm which incorporates a trial division by a multiple and produces two quotient bits for each iteration cycle. A non-restoring approach is used because by following a trial subtraction which overdraws, with a trial addition, restoration cycles are eliminated.

By execution time, the divisor is placed in the J register for RX format instructions or the M register for RR format instructions. The divisor is hex-normalized by gating it through the main adder and shifter. The result is placed in the K register and the L register. The X 3/2 divisor is generated by adding the contents of the K register to the contents of the L register shifted right one. The result is returned to the L register.

The dividend is placed in the M register, shifted the proper amount, and swapped with the K register. The required divisor multiples are located in the M register and L register. The X1 and X 3/2 divisors are obtained by a direct read out from the registers, and the X 1/2 and X 3/4 divisors are obtained by shifting right one.

The quotient is assembled in the J register; every second iteration causes the J register to be shifted left four bits by gating it to the RBL and back into the J register; thus space is provided for the next four quotient bits. The shift counter is set to the iteration count and division takes place until the shift counter is decremented to zero or three.

The first step in each iteration cycle is the selection of the divisor. If the dividend is true, the multiple is subtracted from the dividend; if the dividend is complement, the multiple is added to the dividend, and the quotient bits are entered into the J register. Most iteration cycles produce two quotient bits; however, if the X 3/4 divisor is used, three bits are generated. In this case, the third bit is retained and entered in place of the high order quotient bit developed during the next iteration cycle.

When the shift counter is reduced to one or three, the iteration cycles are terminated. If the last divisor used is the X 3/4, the quotient is complete; however, if the X 3/4 divisor is not the last divisor used, one more quotient bit is developed. The last quotient bit is generated by reducing the dividend by the X1 divisor.

In floating-point divide, the exponent is computed, tests are made for exponent overflow, exponent underflow, lost significance, and for division by zero. Also, the quotient is transferred to the floating-point register specified by R1. In fixed-point divide, the remainder is developed and transferred to the general register specified by R1 and the quotient is transferred to R1 + 1.

VFL and Logical Operations: Operands for comparing, translating, editing, bit testing, and bit setting are provided for processing logical fields of fixed and variable lengths. Fixed-length logical operands, which consist of one, four, or eight bytes, are processed from the general registers. Logical operations are also performed on fields up to 256 bytes in length, in which case the fields are processed from left to right, one byte at a time. Two scanning instructions permit byte-by-byte translation and testing via tables. An important special case of variable-length logical operations is the one-byte field, whose individual bits are tested, set, reset, and inverted as specified by an 8-bit mask in the instruction.

Decimal arithmetic imporoves the performance for processes requiring few computational steps per datum between the source input and the output; decimal arithmetic is provided with the operands and the result located in storage. Decimal arithmetic includes addition, subtraction, multiplication, division, and comparison.

Decimal digits are represented in four-bit binary-coded-decimal; they are packed two to a byte, appearing in fields of variable length (from 1 to 16 bytes), and are accompanied by a sign located in the rightmost four bits of the low-order byte. Operand fields are located on any byte boundary and have lengths up to 31 digits plus sign. Instructions are provided for packing and unpacking decimal numbers; packing of digits leads to efficient use of storage, increased arithmetic performance, and improved rates of data transmission.

The execution of VFL and decimal instructions is handled by a separate section of the E unit. The K register and the L register are data input sources; the K register doubles as the result register. During VFL and decimal operations, normal I unit operations are suspended and the addressing facilities of the I unit are placed at the disposal of the VFL controls for generation of storage operand addresses.

Generally, a VFL operation requests data from core storage containing operands one and two. These data are placed into the K register and the L registers. The operand bytes are gated to the VFL E unit and the required operation is performed; the result byte replaces the operand byte in the K register. When either the K register or the L register operand is depleted and the operation is not complete, a request is made to obtain the next sequential storage word.

The central processing unit normally takes instructions in sequential order. After an instruction is obtained from a core storage location specified by the instruction counter, the instruction counter is incremented by the number of bytes in the instruction.

Most branching is accomplished by inspecting the condition of two bits of the condition register (bits 34-35 of the PSW). Many of the arithmetic, logical, and input/output operations indicate their outcome by setting the condition register to one of its four possible states (00-11). Subsequently, a conditional branch operation selects one of its four possible states as a criterion for branching. For example, the condition code reflects conditions such as non-zero result, first operand high, operands equal, overflow, channel busy, zero, etc. Once the condition register is set, it remains unchanged until modified by an instruction that reflects a different condition code.

The program status word (PSW), a double word having the format shown in Figure 11, contains information required for proper execution of a given program. The program status word includes the instruction address, condition code, several mask bits, and several mode fields. The active or controlling program status word is called the current PSW; the status of an interrupted program is preserved by storage of the current program status word.

Five classes of interrupt conditions are distinguished:

1. Input/output
2. Program
3. Supervisor call
4. External, and
5. Machine check.

For each class, two program status words, called old and new, are maintained in the main storage locations shown in Figure 12. An interrupt in a given class stores the current program status word as an old program status word and then takes the corresponding new program status word as the current program status word. If, at the completion of the interrupt routine, the old and current program status words are interchanged, the system is restored to its prior state and the interrupted routine is continued.

The systems mask, program mask, and machine-check mask bits in the program status word are used to control certain interrupts. When masked off, some interrupts remain pending while others are ignored. The system mask keeps external and input/output interrupts pending; the program mask causes four of the 15 program interrupts to be ignored, and the machine-check mask causes machine-check interrupts to be ignored.

Response of the central processing unit to a special condition in the channel and input/output unit is facilitated by an input/output interruption. The address of the channel and input/output unit are recorded in the old program status word while related information is preserved in a channel status word (CSW) that is stored as a result of the interrupt.

Unusual conditions in a program create program interruptions. Eight of the fifteen possible conditions involve overflow, improper divide operations, exponent underflow, and lost significance. The remaining seven deal with attempted execution of privileged instructions, improper addresses, and similar conditions.

A supervisor-call interrupt results from executing the supervisor call instruction. Eight bits from the instruction format are placed in the interrupt code of the old program status word. Supervisor call permits a problem program to switch central processing unit control back to the supervisor.

The central processing unit responds to signals from the interrupt key on the systems control

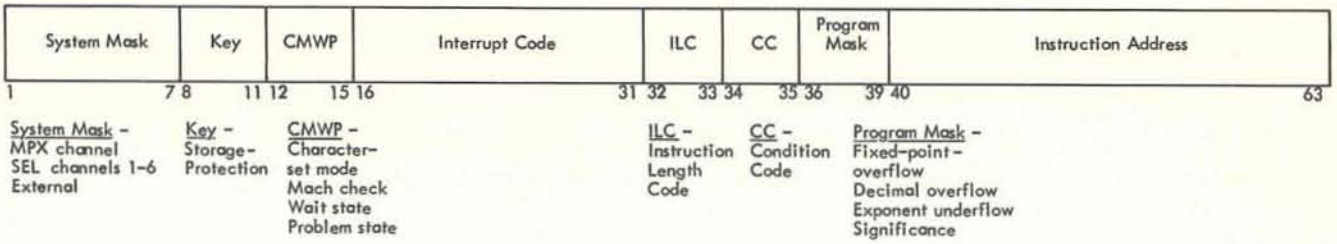


FIGURE 11. PSW FORMAT

Address	Byte Length	Purpose
0	8	Initial program loading PSW
8	8	Initial program loading CCW 1
16	8	Initial program loading CCW 2
24	8	External old PSW
32	8	Supervisor call old PSW
40	8	Program old PSW
48	8	Machine check old PSW
56	8	Input/output old PSW
64	8	Channel status word
72	4	Channel address word
76	4	Unused
80	4	Timer
84	4	Unused
88	8	External new PSW
96	8	Supervisor call new PSW
104	8	Program new PSW
112	8	Machine check new PSW
120	8	Input/output new PSW
128		Diagnostic scan-out area*

*The size of the diagnostic scan-out area is configuration dependent.

FIGURE 12. PERMANENT STORAGE ASSIGNMENTS

panel, the timer, special devices, or other central processing units through an external interrupt signal. The source of the interrupt is identified by an interrupt code in bits 24-31 of the program status word.

A machine check (if unmasked) is caused by a hardware malfunction; it cannot be caused by invalid data or instructions, and it terminates the current instruction, initiates a diagnostic procedure, and effects a machine-check interrupt. When several interrupt requests occur during execution of an instruction, they are honored in a predefined order.

Overall central processing unit status is determined by four alternatives:

1. Stopped versus operating state,
2. Running versus waiting state,
3. Masked versus interruptible state, and
4. Supervisor versus problem state.

In the stopped state (entered and left by manual procedure), instructions are not executed, interrupts are not acknowledged, and the timer is not updated. In the operating state, the central processing unit is capable of instruction execution and of being interrupted.

In the running state, instruction requests and execution take place in a normal manner. The wait state is entered by the program to wait for an interrupt. In the wait state, no instruction processing takes place, the timer is updated, and input/output and external interrupts are recognized if not masked off. Running versus waiting state is determined by the setting of a bit in the current program status word.

Central processing unit operations are interruptible or they are masked for system, program, and machine interrupts. When the central processing unit is interruptible for a class of instructions, interrupts are accepted. When the central processing unit is masked, system interrupts remain pending and program and machine-check interrupts are ignored. The interrupt states of the central processing unit are changed by altering mask bits in the current program status word.

In the problem state, processing instructions are valid, but all input/output instructions and a group of control instructions are invalid. In the supervisor state, all instructions are valid. The choice of problem or supervisor state is determined by a bit in the program status word.

All instructions are initially processed in the instruction unit. The basic I time consists of two sequencers, T1 and T2. Both sequencers may stay on for more than one clock cycle. During T1 and

T2 time, the instruction is decoded, addresses are computed, and most operand requests are made. At the end of the last T2 cycle, the operation code is transferred to the execution unit where the instruction is completed. T1 of the next instruction usually follows T2 of the previous instruction.

At the beginning of every T1 cycle, the two halfwords of the AB register addressed by the gate select latch (gate select addresses the left-most half word) are gated into the IOP register. During T1, instruction decoding is done from the IOP register and any registers required for an effective operand address are gated to the addressing adder. With the turn on of T2 (called TN T2), the addressing adder output is gated into SAR and/or the H register; and any required operand fetch requests are made; operands requested are generally returned to the E unit. During T2, GRP, or FPL register operands required by the E unit are gated to the RBL. At the end of T2, the I to E transfer occurs, providing an interrupt has not occurred. At this time, the instruction is transferred to the execution unit. During T2 time, decoding is done from both IOP and BOP. The turn on of T1, T2, and the I to E transfer are dependent on several conditions being met.

E unit decoding is done from the E-operation register (EOP) in the E unit which is set from IOP. A second E unit op register (LCOP) provides the op code during the last cycle of the E unit instruction execution. During LCOP, EOP is being set for the next instruction, thus assuring the E unit of one cycle of decoding before the I to E transfer. The instruction counter controls control the normal advancing of the instruction counter and the normal requesting of instructions. The instruction counter register (ICR) contains 24 bits. Bits 20-22 are advanced after the I time of each instruction; bits 0-19 are advanced independently of bits 20-22. Bit 23 of the instruction counter register is always zero because instructions start at halfword addresses. The instruction counter register is advanced by two adders: the gate select adder for advancing bits 20-22 and the incrementer for advancing bits 0-19. The gate select adder works in conjunction with the gate select register (GSR) to select gates from the AB registers to the IOP register.

The instruction counter (IC) controls also generate normal instruction counter requests and generate the instruction address. The addresses are generated by adding in the incrementer an amount equal to the length of the instruction plus the amount in the instruction counter register. The controls attempt an instruction counter request as soon as an empty instruction buffer is detected, but any instruction being executed may block an instruction counter request if it would interfere with the instruction being executed. Whenever an instruction counter block condition is generated, the instruction counter request which is not honored by the bus control unit is cancelled.

Special instruction counter request rules are implemented to insure that A and B registers are never both empty by forcing instruction counter requests into the instruction stream and by suppressing instruction requests in anticipation of branch requests.

Instruction counter request addresses are obtained by gating the instruction counter into the incrementer and adding the proper increment amount. The increment amount is determined from the value of the instruction counter register and the empty condition of the A or B register. Bit 20 of the instruction counter request address is zero if the request is for the A register, or a one if the request is for the B register.

Branch requests are made at TN T2 time similar to operand requests. Branch request return addresses are generated for the AB register and the J register. If bit 20 of the address is a zero, the request is returned to the A register, and if bit 20 of the address is a one, the request is returned to the B register.

Branch instructions initiate a request to fill the second buffer. The request is called the branch +1 (BR +1) request, and it is obtained from the storage location following the location of the branch-to instruction. The address for the branch +1 request is computed during T2 of the branch instruction, and it is normally made at the I to E transfer. The return address for the branch +1 request is opposite to that specified for the branch request.

During E time of the branch instruction, the success of the branch is determined. The tests complete (tests cmplt) trigger is turned on, and during this cycle the line branch successful tells the success of the branch instruction.

If the branch operand is returned before tests complete of any branch, the operand is not placed in the AB register; however, it is returned to the J register. If a successful branch is determined during tests complete and the branch operand is already in the J register, the branch operand is gated from the J register into the proper register of the AB register. If the branch operand is returned after tests complete is turned on and the branch is successful, the branch operand is gated into the proper register of the AB registers.

The branch +1 request is made late enough so that the operand is returned after tests complete is

turned on and the success of the branch is determined. If the branch is successful, the BR + 1 operand is gated into the proper register of the AB register; however, if the branch is unsuccessful, the BR + 1 operand is blocked upon its return. If the branch is successful, the branch address (contained in the H register) is gated to the gate select latch through the incrementer and to the instruction counter. If a branch is unsuccessful, normal processing of the next instruction starts at the same time tests complete is turned on.

The execute instruction is processed as a branch instruction, but the instruction counter is not replaced by the contents of the H register. After the instruction constructed by the execute instruction is performed, an instruction counter recovery cycle is taken providing the instruction is not a successful branch instruction. The instruction counter recovery cycle allows the central processing unit program to proceed with the instruction following the execute instruction. The program status word is incremented by the length of the execute instruction rather than the length of the instruction obtained by the execute instruction.

2365 PROCESSOR STORAGE

- 0.75 microsecond access time.
- Eight byte width on an interleaved access is possible.
- Interleaved and overlapping access for greater speed.
- Three main data paths are store data, fetch data, and address.

The IBM System/360 Model 75 is supplied with a minimum of 32 K, eight byte words of 0.75 microsecond processor storage (one 2365, Model 3) or a maximum of 128 K, eight byte words of 0.75-microsecond processor storage (2365 Model 5). Each 2365 processor storage has a word width of eight bytes (64 data bits and 8 parity bits) on an interleaved access; the Model 3 is a two-way interleaved access, and the Model 5 is a four-way interleave.

Simultaneity in core storage operation is obtained by overlapping the cycles of the two storage units; addresses are staggered in the two units so that a series of requests for successive words activate the two units alternately, thus doubling the maximum rate of the single unit. The 2365 processor storage unit functions the same as any core storage unit; its prime purpose is to provide a quick, high-speed access to data and instructions. Each 2365 Processor Storage houses two ferrite core arrays and their associated logic.

Figure 13 represents the three main data paths necessary to operate the 2365 Processor Storage. The address is an incoming data path carrying data that specifies the location in the processor storage that is affected. Addressing and the transfer from the central processing unit is done in binary. The address transfer from the central processing unit is momentary; therefore, the address must be retained in the processor storage unit. This address is retained in the storage address register (SAR), and is decoded to select the proper X and Y addresses; the output of the decoder selects one 72-bit word out of many possible locations.

The store data signal is an incoming data path and brings information from the central processing unit, systems control panel, or input/output channels to the processor storage unit. This data, since it is also momentary, is stored in the memory data register (MDR) instead of the information read out of the specified address. The write cycle places the data into the specified location.

The fetch data is an output data path from the processor storage unit memory data register to the central processing unit registers or to the input/output channels. The data in the memory data register is also used to rewrite the addressed location because the core read-out cycle is a destructive readout. Figure 14 represents a typical logic flow of any core storage cycle.

2361 LARGE CAPACITY STORAGE

- Prime purpose of LCS is to increase storage capacity.
- One 2361 LCS provides up to 2048 K bytes.
- LCS provides a means of intercommunication between systems.
- Interleaving is an option.

The 2361 large capacity storage (LCS) unit functions the same as any core storage unit; its prime purpose is to increase the amount of storage available, on an immediate access basis, to the system.

The maximum number of 72-bit words available in one large capacity storage (LCS) is 262,144. Figure 13 also represents the three main data flow paths that are necessary for operating the large capacity storage unit. The address is an incoming data path carrying data that specifies the location in storage is affected. The addressing is done in

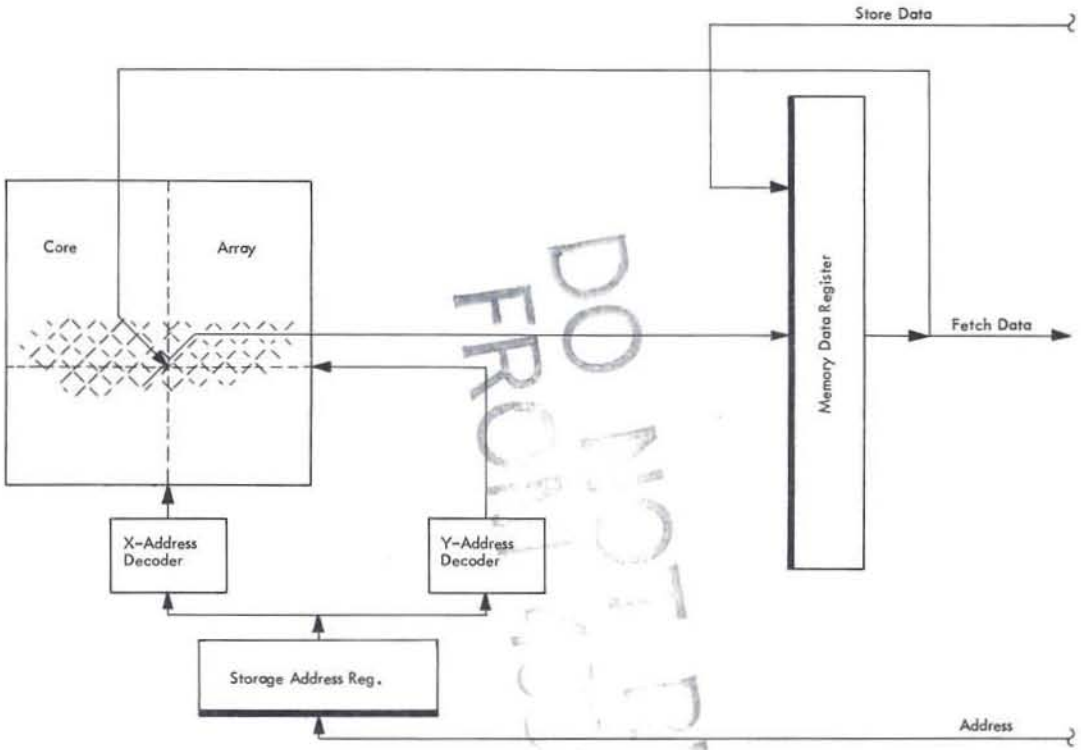


FIGURE 13. SIMPLIFIED CORE STORAGE OPERATION

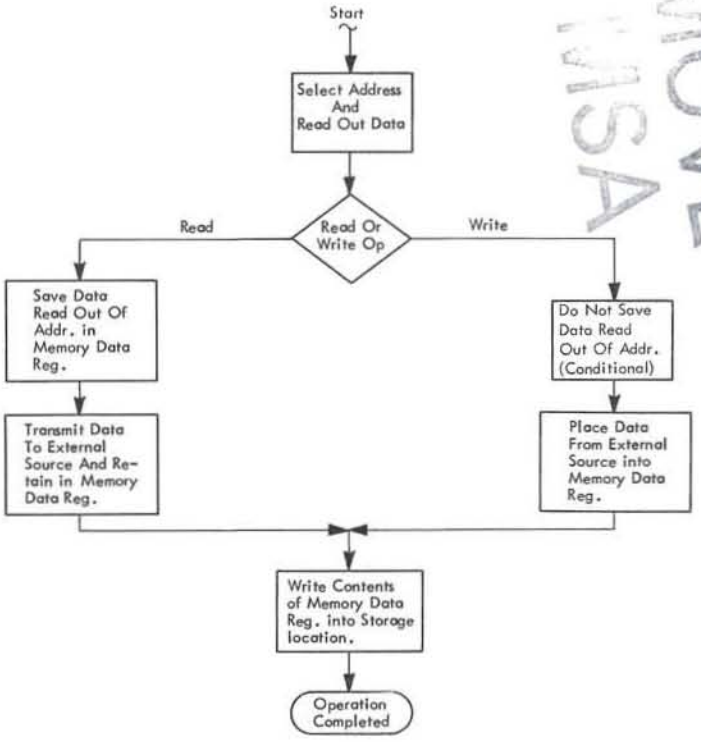


FIGURE 14. BASIC CORE STORAGE OPERATION

binary, and the address transfer from the central processing unit is momentary; therefore, the address must be retained in the large capacity storage unit. This address is contained in the storage address register, and is decoded in groups of one, three, and four bits each. The outputs of the decoders are combined until the selection is narrowed down to one 72-bit word out of the 262,144 possible word locations.

The store data is an incoming data path and brings information from the central processing unit, systems control panel, or input/output channel to the large capacity storage unit. This data, since it is momentary, is stored in the memory data register in place of the information read out of the specified address. The write cycle places this data into the specified location.

The fetch data is an output data path from the large capacity storage unit's memory data register to the central processing unit registers or to input/output channels. The data in the memory data register is also used to rewrite the addressed location because the read-out portion of the cycle is a destructive read-out.

The 2361 Core Storage shared storage feature provides a means of communication between two IBM System/360, Model 50, 60, 62, or 75. Two systems can have access to the data and instructions contained in the large capacity storage unit. Through this arrangement, one program is available to two systems or data for the programs in two systems is available to both systems for immediate access. Up to four 2361 large capacity storage units in any combination of Model 1 (1024 K bytes) and Model 2 (2048 K bytes), can be attached. Interleaved operation of large capacity storage is an option, with the restriction that two 2361 storages of the same model be paired for interleaving. Pairs of interleaved 2361's can be attached with 2361's not equipped for interleaving, in which case, the interleaved pairs are assigned the lower addresses.

INPUT/OUTPUT CHANNELS

- I/O operations are performed by selector channels, control units, and I/O devices under control of a supervisory program.
- Interlocks prevent one I/O program from interferring with another I/O program.
- The approximate maximum data rate is 156,000 eight-byte words per second.

The IBM System/360 Model 75 input/output operations are performed by selector channels, or multiplexor channels, control units, and input/output devices operating under the control of a supervisory program. The supervisory program allocates equipment to multiple programs and monitors the execution of each program. The supervisory program provides a means of assigning to each program the proper input/output device. This assignment consists of establishing a path for exchanging control and status information between the program and the input/output device and for transferring data between the input/output device and core storage. The design must anticipate erroneous problem programs and provide means whereby one program is protected from interference by another.

The channel hardware and the supervisory program must solve the following problems:

1. Input/output devices must be protected; a problem program should have access only to its assigned input/output device and must be prohibited from writing and reading on another device. On devices such as magnetic tape and disk storage, protection is necessary for units of storage smaller than the entire recording medium, so that a program is permitted access to some areas but not to others.

2. Core storage must be protected from input/output operations. An input/output operation may permit transferring information only to storage areas assigned to its associated program.

3. The program is often concerned with particular recording mediums, such as tape, disk packs, and cards, and not with the physical device on which the input/output operation is performed.

4. Since areas of main storage may be assigned to the problem program at execution time, the program must be able to specify data addresses symbolically.

5. When an input/output operation is terminated, status information must be channeled back to the program that initiated the operation. The input/output device must be identified, and the extent of the core storage area used must be communicated.

Other interlocks and supervisory functions are provided by the channel. For direct-access storage devices (disk and drums) data transfers may involve a sequence of operations, such as positioning the access mechanism and searching for the designated data block. Two programs could interfere with each other during this sequence by initiating input/output operations for the same channel, or the input/output device, if accessible from more than one channel. Therefore, facilities must be provided

to protect a chain of input/output operations specified by one program against interference by input/output operations from another program. The design of the system considers the channel as an independent unit which executes a program consisting of commands. The central processing unit program starts the channel operation by specifying the beginning of the channel program and the device to be used. The command specifies the direction of data flow (read or write), the data source or destination in main storage, and all functions associated with the data transfer. Upon completion of the channel program, the central processing unit program is interrupted and status information is made available.

Depending upon performance requirements, a system may contain up to six selector channels with an approximate maximum data rate on one channel of 156,000 eight-byte words per second. The selector channel utilizes the central processing unit data paths for initiation and termination of the input/output program, but does not use the central processing unit data paths for byte transfer, storage word transfer, or for chaining operations. Central processing unit and input/output overlap operations are possible.

External storage devices, as well as the equipment used to communicate with the external world, are referred to as input/output devices; the use of these devices by the central processing unit is referred to as an input/output operation. In addition to magnetic tape units and direct access storage devices such as disks and drums, input/output devices include card readers, card punches, printers, inquiry stations, visual displays, process control devices, and devices for receiving and transmitting information over communication lines.

A typical input/output device requires control equipment that is unique to its particular function. This equipment is referred to as the control unit, and it is considered part of the input/output device. Some devices, such as magnetic tape units, share a common control unit which is a separate unit.

The part of the system that connects the input/output device to the central processing unit and main storage is the selector channel. The selector channel contains the equipment necessary for attaching input/output devices and their control units to the system and for synchronizing input/output data cycles with core storage cycles. The input/output channel maintains control over the input/output operation at all times.

Execution of Channel Programs

- I/O operations are initiated by commands.
- Data transfer is under control of the I/O device and I/O channel.
- Termination of an I/O operation provides a status byte.
- Commands are coupled into a channel program.

Input/Output operations are initiated by commands, which constitute the channel program. A command is specified in a channel command word (CCW) and is decoded by the channel.

The basic input/output operations are: read, which causes data transfers from an input/output device to core storage; write, during which data from core storage are recorded at the device. In either case, the channel command word designates the storage area by addressing the initial data address and the number of bytes contained in the storage area. Data are taken from or placed into core storage in ascending order.

A variation of the basic read operation is the read backward command. This command initiates the transfer of data from the device (such as magnetic tape) to the channel in a reverse order. In such a case, the channel command word designates the highest address in the storage area, and the data are placed into storage in descending order.

The volume of data transferred during an input/output operation is under the control of the device as well as the channel. The device cannot transfer more data than specified by the channel program. When the allocated storage areas are filled, the channel terminates the operation and requests an "end" signal from the device. On the other hand, if the device receives or transmits the specified number of data bits or words associated with the operation, the device signals the end condition regardless of whether or not the allocated storage is filled.

The program specifies the input/output operation by the 8-bit command code in the channel command word. The channel transfers the entire code to the device; for the most part, the code contains all the information needed by the device to initiate the data transfer. A portion of the command code is common to all devices and indicates the data

flow direction; the remainder of the code depends on the type of device.

For disk files, the program must position the access mechanism before data are transferred. The use of magnetic tape units involves such operations as backspacing the tape, rewinding, or loading a tape cartridge. These functions are unrelated to data transfer and cannot be easily combined with reading and writing operations without tieing up the channel facilities.

All auxiliary control functions are specified by the programmer as orders. The orders are decoded by the input/output device, and the codes are transmitted to the device in a control operation. The orders are encoded in the command code of the control operation; if additional information is needed, it is obtained from the core storage area designated by the channel command word. A control operation is indistinguishable from a write operation in the channel.

When an input/output operation is terminated, a byte is provided to indicate the general condition during the operation. The condition is made available to the program in the sense operation. The sense command is a request to the input/output device for status information, such as the position of a magnetic tape, the condition of the card stacker and hopper, or the error conditions detected in the last operation. The status information is transferred to the channel as data during read and is placed in the core storage address specified by the channel command word. Sense data is indistinguishable from read data.

Commands are coupled into a channel program by chaining channel command words. Chaining is specified by two flags in the channel command word; the presence of either causes the channel to request a new channel command word. The channel command words are brought from sequential locations in core storage, unless the transfer-in-channel command is encountered, which causes the channel to branch to the location specified in the channel command word.

When command chaining is used, the channel uses the new channel command word to initiate a new operation at the device. Command chaining reduces the frequency of communications between the channel and central processing unit, and permits a single input/output instruction to start such sequences as printing multiple lines, or reading multiple tape blocks. Chaining also permits auxiliary functions, such as backspacing tape and data-transfer operations to be coupled, thus allowing multiple input/output operations.

A storage protect mechanism is needed for the execution of multiple programs in the central processing unit; protection is implemented in hardware and is extended to storage references made by the input/output channels.

Certain input/output operations executions are contingent upon the result of preceding input/output operations. For example, on direct access storage devices data transfer is initiated only when the designated data block is under the recording or reading head. Usually the data block is identified by a key field immediately preceding the data area. To establish the relative position of the recording medium and the head, the system must match the identifier specified by the program with that appearing on the recording medium. The match can be performed by the channel or the device. In the Model 75, the comparison is performed in the device, and the channel is programmed to execute a closed loop of commands. The channel remains in the loop until the device signals a successful match.

Communications Between CPU and Channels

- CPU controls channel activity by four I/O instructions: start I/O, test I/O, halt I/O, and test channel.
- A connection established between the control unit and CPU releases the CPU for further operations.
- The channel program interrupts the CPU when an I/O operation is terminated.

The central processing unit controls channel activity by means of four input/output operations: start I/O, test I/O, halt I/O, and test channel. Each channel and, when applicable, the device, such as a particular tape unit or communication line, is identified by an address. In direct-access storage units, such as magnetic disks and drums, each access mechanism is considered a separate device.

The execution of a channel program for an input/output device is initiated by the central processing until issuing a start input/output instruction. This instruction provides the channel with storage-protection information, the address of the first channel command word, and causes a logical connection to be established between the channel and the designated device. The central processing unit is involved with the operation until the device responds and both channel and device verify that the operation can be executed.

Once the device is started and the channel is set up to execute its program, the central processing unit is released. In input operations, the channel accepts data from the device, assembles, when necessary, the 8-bit plus parity bytes into units equal to the double-word length for core storage, and transfers the assembled word to the designated area in core storage. During output operations, the reverse process takes place, with the channel normally receiving double words from core storage and sending the word in 8-bit plus parity bytes to the input/output device. Once the initial contact is made, the central processing unit operates in a normal manner, unhindered by channel operations except for the delay caused by channel core storage requests.

The central processing unit program retains control over the channel program, and when input/output activity must be rescheduled in response to conditions occurring after the channel program is started, the central processing unit issues a halt input/output operation which terminates the data transfers and further channel command word requests by the channel.

When the channel program is terminated, the channel interrupts the central processing unit operations and makes a channel status word available. This word identifies the last channel command word used, the amount of data transferred, and provides storage-protection information associated with the chain of operations. The status word also contains a status byte received from the device and a set of status bits provided by the channel, both of which describe the conditions of termination.

In order that the central processing unit program has a means of establishing in advance when conditions in the channel or device should alert the program, a mask bit is associated with each physical channel. A masked channel cannot cause an input/output interrupt; consequently, the central processing unit can suppress input/output interrupts by masking the channels. The conditions in the channel and devices are preserved until accepted by the central processing unit. A test channel instruction is used to determine if an interrupt condition is pending in the channel.

Channels intended for high-speed operation contain one subchannel. These channels are referred to as selector channels and are used for attaching such devices as magnetic tapes, disk, and drum units. A selector channel can sustain only one data-transfer-operation at a time. Once a data-transfer is initiated, a logical connection is established between the addressed device and the channel. The connection is maintained for the duration of the operation, and is established under program control. Other devices cannot communicate with the channel while a data-transfer operation is in progress. Such operations as backspacing a tape file or positioning a disk access mechanism may take place simultaneously.

2860 SELECTOR CHANNEL

- Data rates up to 156,000 double words per second are possible.
- Each channel contains its own CE panel.
- Maximum of six channels per system is possible.
- Maximum of eight control units per channel.
- The channel has FLT circuits.
- I/O operations are overlapped with CPU operations.
- The channel operates in burst mode.

The 2860 Selector Channel is a high-performance data channel designed to operate at data rates up to approximately 156,000 double words per second. The rate is measured by the number of double words that pass the interface to or from an appropriate input or output control unit.

The channel uses 30-nanosecond SLT circuits packaged on a mixture of one high and two high six-pac cards. The frame houses three swinging gates, each capable of containing 20 large boards. Two of the large boards are occupied by a CE panel; two large board spaces are left for options. Each channel occupies one full gate; up to three channels may be contained in the three gate frame. Each channel not only contains its own CE panel, but also contains manual controls and CE controls except power and biasing, which are mounted on the front of the frame.

Up to eight control units may be attached to one channel; however, only one device may be operated at any one time. Operational lines provide the only control needed during normal program operation, and the controls for implementing the fault location tests are incorporated in the channel. The scan operation requires large volumes of data broken up into short tests. This date are on tape or disc packs, and it must be brought into memory without the benefit of central processing unit instructions or interruptions; thus, the channel must work in conjunction with the fault locating tests controls to supply the data, keep the fault location tests

controls aware of the progress, retry when data errors are discovered, and start and stop transmission when indicated to do so by the fault location test controls.

The 2860 Selector Channel directs the flow of information between the input/output device and main storage. The channel relieves the central processing unit of the responsibility of communicating directly with the input/output device. Control is accepted from the central processing unit in the program supplied format; it is expanded into a sequence of signals acceptable to a control unit. After an operation is initialized, the channel performs an assembly or disassembly operation on the data, and synchronizes its transmission over the interfaces with the main storage unit or the input/output control unit.

Since the 2860 Selector Channel contains all of the necessary facilities for controlling the input/output operation, input/output operations are completely overlapped with the central processing unit activity. The only main storage cycles required during input/output operations are those needed to obtain input/output control words and to transfer the data in 8-byte (double words) blocks to or from main storage. These cycles interfere with the central processing unit program only when both units require concurrent use of the same main storage.

The selector channel operates in the burst mode. This mode extends over the entire block of data or when command chaining is specified over a whole sequence of blocks. Other input/output devices cannot be involved in a data transmission at the same time but they can execute operations that do not involve communicating with the channel. The selector channel scans the input/output devices for interrupt conditions when it is not executing an operation or chain of operations.

Channel Operation

- I/O operations are initiated and controlled by instruction, command, or control formats.
- I/O instructions generate the start I/O signal to the channel.
- The channel address word is requested by the channel.
- The command control word is requested from main storage.
- The channel status word tells the result of the I/O operation.
- The I/O device controls the duration of the I/O operation.

Input/output operations are initiated and controlled by three types of formats: instruction, command, and control. Instructions are decoded and executed by the central processing unit and are a part of the central processing unit program. Commands are decoded and executed by the channel and initiated input/output operations such as reading and writing. Any functions, such as a disk file seek, that are peculiar to a device are specified by means of control orders. Control orders are transmitted to input/output devices as data and are decoded and executed by the input/output device.

A read, read backward, write, sense, or control input/output instruction is initiated by the central processing unit sending the start I/O line to the selector channel, placing the 8-bit unit address on the unit address bus, and signaling the proper channel on the select channel line. If the subchannel is busy, condition code 10 is sent to release the central processing unit, or if the subchannel is not available, condition code 11 is sent to release the central processing unit. When the channel is available, the unit address is gated to the unit address register and the channel address word (CAW) is requested from storage location 72.

The channel address word (Figure 15) contains the command address (CA) and the storage protection key which are gated to their respective registers. A storage request is initiated for the command control word (CCW) (Figure 16). When the bus control unit (BCU) response is received, the channel places the command address on the storage address bus (SAB) and waits for an advance pulse which sets the command code, data address, flags, and count into the proper registers. During this time, the command address is incremented by one.

The device is selected by placing the unit address (UA) on the bus out and sending the address out and the select out signals. The control unit, upon recognizing its address, responds with the operational in signal which causes address out to drop. When the control unit loses the address out signal, it places the address of the device selected on the bus in lines and raises the address in line. The channel compares the address from the control unit with the one it sent. If a proper compare results, the command control word valid trigger is on, and no errors occurred, the operation is continued by placing the command code on the bus out and raising the command out line. The control unit responds with condition code 00 and releases the central processing unit. The channel and device executes the command received in the command control word.

Key - Specifies the storage protection tag for all commands associated with start I/O.

Command Address - Specifies the location of the first Channel Command Word in Core storage.

FIGURE 15. CHANNEL ADDRESS WORD

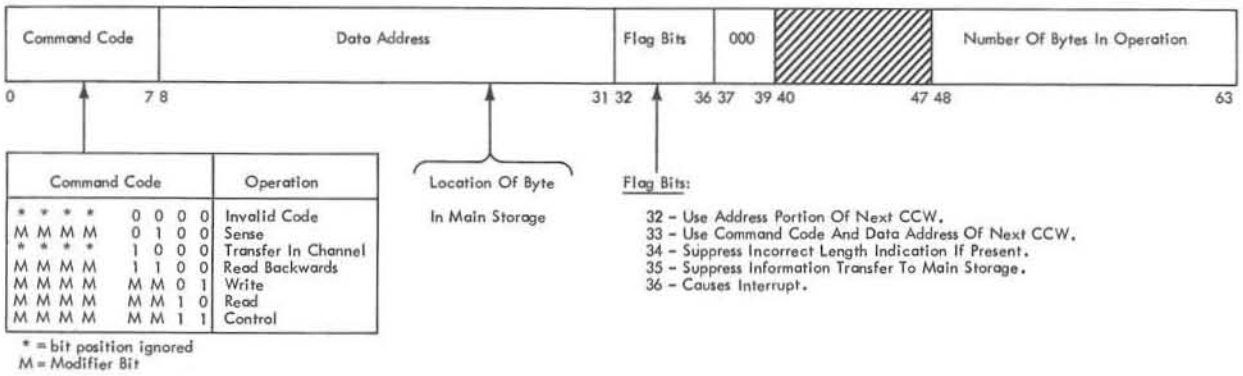


FIGURE 16. CHANNEL COMMAND WORD FORMAT

If an error occurs, a hardware generated test I/O code is placed on the bus out instead of the command code and the device is relieved of its status. The channel disconnects the device and requests a storage cycle. When the bus control unit response is received, the status information is placed in the channel status word and the central processing unit is released.

The channel address word specifies the storage protection key and the address of the first channel command word associated with the start I/O instruction. The channel address word is requested automatically from storage location 72. The channel address word (Figure 15) has the following format:

1. Bits 0-3 (key) specifies the storage protection tag for all commands associated with the start I/O instruction.

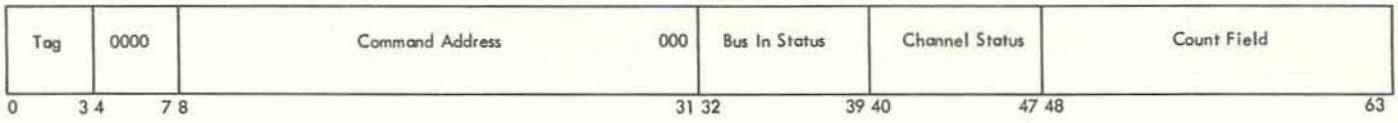
2. Bits 4-7 must contain zeros.

3. Bits 8-31 (command address) specifies the location of the first channel command word in core storage.

The channel command word (Figure 16) is requested from the location specified by bits 8-31 (command address) of the channel address word, and has the following format:

1. Bits 0-7 (command code) contains the operation code. It may be sense, transfer in channel, read backwards, read, write or control.

2. Bits 8-31 (data address) specifies the address of the first byte in storage. For all operations except read backwards, it specifies the high-order byte. For read backwards, it specifies the low-order byte.



Tag: Contains storage protect tag obtained during start I/O instruction. Obtained from Channel Address Word.

Command Address: Contains last command (+8 bytes) address used by the 2860 channel. Positions 29, 30, 31 are stored as zeros making the address a double-word boundary address.

Bus In Status: Status Bits received from the device.

Channel Status: Information generated by the channel to indicate the status of the preceeding command.

Count Field: Contains the residual count of the last CCW used by the 2860 Selector Channel.

FIGURE 17. CHANNEL STATUS WORD

3. Bits 32-36 (flag bits)
   a. Bit 32--use address portion of the next channel command word.
   b. Bit 33--use command code and data address of the next channel command word.
   c. Bit 34--suppress incorrect length indication if present.
   d. Bit 35--suppress information transfer to main storage.
   e. Bit 36--cause interrupt.
4. Bits 37-39 must be zeros
5. Bits 40-47 not used
6. Bits 48-63 specify the number of bytes of data in the operation.

When the sequence of operations initiated by the start I/O instruction is terminated, the channel or the device generates interrupt conditions. These interrupt conditions are brought to the attention of the program by the input/output interrupt mechanism, by the test I/O instruction, or in certain cases by the start I/O instruction. When an interrupt condition is cleared, the channel and the input/output device makes available to the program one or more status conditions that describe the result of the last operation. These conditions, an address, and a count indicating the extent of the operation sequence are presented to the program in the channel status word (CSW). The channel status word format is shown in Figure 17.

A start I/O instruction causes the input/output channel and device to perform certain tests during the initiation of the instruction. A command can be rejected by any of the following conditions:

Busy
Unit check
Exceptional condition
Program check
Channel control check
Interface control check
Unavailable device

When a channel rejects an instruction, it is indicated by the channel sending condition codes 01, 10, or 11 when it releases the central processing unit. Figure 18 lists the input/output instruction, the condition codes, and their meanings:

Instruction	Condition Codes			
	00	01	10	11
Start I/O	Available	CSW stored	Busy	Not available
Test I/O	Available	CSW stored	Working	Not available
Halt I/O	Not Working	CSW ready	Halted	Not available
Test Channel	Not Working	CSW ready	Working	Not available

FIGURE 18. I/O INSTRUCTIONS AND CONDITION CODES

When the input/output device accepts a command, the channel is set up for data transmission. This state remains until one of the following conditions terminates the operation at the channel:

1. A channel end signal is received from the input/output device.
2. A halt I/O instruction is issued.
3. A channel control check condition is detected.
4. An interface control check condition is detected.

During a normal execution of an operation, the channel signals the input/output device to terminate data transmission whenever any of the following conditions occur:

1. The storage area specified for the operation is exhausted.
2. A program check condition is detected.
3. A protection check condition is detected.
4. A chaining check condition is detected.

The termination is signaled in response to a service request from the device and causes data transmission to cease.

The input/output device controls the duration of an operation and the timing of the channel end signal by way of data blocking. When blocks are defined for the operation, the device always proceeds to the end of the block before providing the channel end signal.

Channel Interrupts

- Interrupts provide a means for central processing unit to change its state.
- Priority is determined by the position of a device on the channel.
- Interrupts occur when a channel is not masked.
- The channel status word is stored as a result of an interrupt.
- Bus in status bits
- Channel status bits

Input/output interrupt provide a means for the central processing unit to change its state in response to conditions which occur in input/output device or channel. These interrupt conditions are caused by the termination of an input/output operation or by operator intervention at the device.

Requests for input/output interrupts are initiated by the channel. The device initiates a request to the channel for an interruption whenever it detects one of the following conditions:

1. Channel end
2. Device end
3. Attention
4. Control unit end
5. Unit check
6. Unit exception

The channel end and device end conditions do not cause the channel to request an interrupt when the command chaining flag is on. Unit check and unit exception cause interruptions when the conditions are detected during the initiation of chained commands.

Priority among the devices on any channel for an interrupt is determined by their position on the channel. The priority is determined at the central processing unit; the first device to receive and be capable of responding to the select out tag has top priority. The priority among channels is also determined at the central processing unit; the lowest numbered channel with an outstanding interrupt condition has top priority.

An input/output interrupt occurs when a channel is not masked and after the execution of the current instruction in the central processing unit is terminated. If a channel establishes a priority among requested interrupts while it is masked, the interrupt occurs immediately after the termination of the instruction that removes the mask. If the priority among interrupts is not established in the channel by the time the mask is removed, the interrupt does not necessarily occur immediately after the end of the instruction that removes the mask.

The channel status word provides the program with the status of an input/output device or the condition under which an input/output operation is terminated. The channel status word is stored as a result of an input/output interrupt; the associated input/output device is identified by the unit address sent to the central processing unit when the interrupt response is released. The basic format (Figure 17) of the channel status word is as follows:

1. Bits 0-3 (protection tag) contain the storage protection tag that was fetched at the time the channel address word was brought into the channel during the last start I/O instruction.
2. Bits 4-7 are always stored as zeros.
3. Bits 8-31 (command address) contain an address which is 8 bytes higher than the last address used by the 2860 Selector Channel. Bits 29, 30 and 31 are stored as zeros, making the address a double word boundary address.
4. Bits 32-39 (bus in status) are the status bits received from the device over the input/output interface.
5. Bits 40-47 (channel status) contain the information generated by the channel to indicate the status of the preceding command.
6. Bits 48-63 (count field) contain the residual count of the last channel command word used by the 2860 selector channel.

The eight bus in status bits (32-39) are received with the status in tag. When the bits are stored in the channel status word, they are stored as they are received over the interface. The bits are interpreted as follows by the selector channel:

1. Bit 32(attention) signal is generated by the input/output device, and is interpreted as an attempt to interrupt the program. Program interpretation is required. An input/output (I/O) device waiting to present the attention condition to the channel appears busy to a command initiated by a start I/O instruction; however, commands sent to the channel by the chaining process do not give the busy appearance.
2. Bit 33 (status modifier) when received with the busy bit (bit 35) is interpreted by the channel as a control unit busy and is treated accordingly. When the bit is received with device end during a chain command operation, it causes the channel to skip the next sequential channel command word in the chaining process.

3. Bit 34 (control unit end) is received from the input/output device to indicate that the control unit is free. This bit comes from the device if the device sent a control unit busy in response to a previous command. When this bit is received by the channel an interrupt occurs.

4. Bit 35 (busy) when accompanied by bit 33 indicates that the control unit is busy as discussed previously. The bit, unaccompanied by the status modifier (bit 33), indicates that the input/output device cannot accept a new command because an interrupt condition is pending or it is executing a previously initiated operation.

5. Bit 36 (channel end) is caused by the completion of the portion of the input/output operation involving transmission of data or control information between an input/output device and channel. The bit indicates that the channel is free to accept the next operation.

6. Bit 37 (device end) is a signal from the input/output device that it has completed its portion of the input/output operation. This bit allows the channel to chain commands when the chain command flag is on. Like attention and control unit end, if the device end is received during a polling operation by the channel device end it may be rejected and stacked by the channels receiving another instruction.

7. Bit 38 (unit check) is sent by the device when it discovers an unusual condition. When this bit is accompanied by channel end or device end, the operation is automatically terminated even though a chain flag exists.

8. Bit 39 (unit exception) terminates the operation and indicates a condition in the input/output device that does not usually occur.

The channel status bits cause an interrupt condition to be set up in the channel and the channel to be busy until the interrupt condition is recognized by the central processing unit. These conditions are:

1. Bit 40 (program controlled interrupt) occurs when a channel fetches a channel command word and the program controlled interrupt (PCI) flag is on. The interrupt due to the program controlled interrupt flag occurs as soon as possible after the fetching of the channel command word, but may be delayed due to the masking of the channel or other activity in the system. An interrupt is requested by the selector channel but the input/output operation is continued.

2. Bit 41 (incorrect length indication) occurs in the channel any time the apparent record length on the device and the count received in the channel command word do not agree and the SLI flag is off. Incorrect length condition terminates command chaining and causes an interrupt.

3. Bit 42 (program check) is caused by programming errors that are detected in the channel:

   a. Invalid channel command word address specification.
   b. Invalid channel command word address.
   c. Invalid command code.
   d. Invalid count.
   e. Invalid data address.
   f. Invalid channel address word format.
   g. Invalid channel command word format.
   h. Invalid sequence. Detection of any program check condition during the initiation of a command causes the operation to be suppressed. If the condition is detected after the input/output device is started, the device terminates the operation.

4. Bit 43 (protection check) indicates that the channel attempted to store data into main storage and the protection keys do not match. The device terminates the operation and the command is suppressed.

5. Bit 44 (channel data check) is caused by parity errors in the channel or main storage. Input operations force correct parity while output operations do not change the parity sent to the device. Data chaining is suppressed but the present operation is not terminated by this check.

6. Bit 45 (channel control check) indicates a machine malfunction that affects channel controls. It includes channel command word fetch parity errors, data address parity, and channel command word contents parity.

7. Bit 46 (interface control check) is caused by an invalid signal on the input/output interface being detected by the channel. It usually indicates malfunctioning of an input/output device and may result from the following:

   a. Device address has invalid parity from the device.
   b. Status byte from input/output device has invalid parity.
   c. The input/output device responded with an address other than the one specified by the channel during command initiation.
   d. The addressed input/output device did not respond during command chaining.
   e. A signal from an input/output device occurred at an invalid time or had an invalid duration.
   f. A signal from an input/output device did not occur during a predetermined time-out interval. Any condition giving the interface control check causes an immediate termination of the operation.

8. Bit 47 (chaining check) caused by the selector channel during data address chaining on input operations. It occurs when the new data address does not fall on double or single-word boundaries and the input/output data address is such that the byte boundary of the first byte received is not determinable. The input/output device terminates the operation.

Initial Program Load

- A device is addressed by switches on the console.
- An IPL causes a complete reset.
- The channel selects a specified unit and reads 24 bytes of data.
- The channel fetches channel command word one and then operates in a normal manner.

The actual device used for initial program loading (IPL) is addressed by switches on the console. These switches feed the central processing unit channel decoder and raise and hold the proper select line until the channel release signal is received.

The handling of the initial program loading read within the channel, and the allocation of memory locations are:

1. Bytes 0-7 IPL PSW (initial program load program status word).
2. Bytes 8-15 IPL CCW1 (initial program load channel command word one).
3. Bytes 16-23 IPL CCW2 (initial program load channel command word two).

The channel uses the initial program load pulse to cause a complete reset. The operational out line to all control units is dropped, thus causing a reset. The selected channel retains the initial program load condition and at the completion of the reset, the CA is 1, DA is 0, the UA register is set, a read operation is forced with a count equal to 24 bytes, and a CC flag is forced in its flag register.

The channel selects the specified unit and reads 24 bytes of data into memory beginning at address zero. If an incorrect length indication (ILI) is generated, it is suppressed and the channel proceeds to act on the CC flag.

The channel fetches channel command word one, which should contain a read command. At this point, the channel is operating under normal rules until the list of commands is exhausted. If any error is detected during the initial program load operations, it does not send a release to the central processing unit.

Optional Input/Output Devices

- Consoles
- Magnetic tape units
- Serial file units
- Parallel file units
- Card readers, card punches, printers
- Displays
- Microfilm input/output units
- Communication and data acquisition devices

Provisions are made for connecting several input/output devices to the Model 75 via the IBM 2860 Selector Channel and the IBM 2871 Multiplexor Channel. Up to eight control units are attached to each selector channel and up to 256 input/output device addresses are available. The Model 75 configuration is shown in Form A22-6888. Included in the configuration are the input/output devices attached to the Model 75.

MULTISYSTEM OPERATION

- A single CPU must be able to perform a variety of tasks.
- Hazards of manual intervention are eliminated by program controlled switching and intercommunication.
- Fail-safe systems perform their entire functions when any single malfunction occurs.
- Fail-soft systems perform their work load in the presence of a malfunction.
- Simplex systems.
- Channel-to-channel adapter.
- Transmission control units.
- Shared control units.
- Shared devices.
- Shared storage.
- Centralized crossbar switch.
- Distributed crossbar switch.

A system consisting of two or more central processing units that can communicate without manual intervention is called a multisystem. Thus defined, the term, multisystem, encompasses a large variety of system configurations and should be distinguished from such terms as multicomputer or multiprocessing systems, which usually are given a more restricted definition.

A single central processing unit must be able to perform a variety of tasks, but it is not equally adept at each of the tasks. In particular, high performance is of no value if it is limited by its input/output capability. The logical design of System/360 permits a wide degree of specialization that makes it attractive for individual specialized jobs within a multisystem system.

Because the specialized computers join in a common job, data and programs must be communicated between them. The ability to communicate data and programs between systems may be a manual process; however, any manual intervention is, by nature, unreliable. The ability to switch tapes, core storage, etc. under program control eliminates the time hazard of manual intervention, and improves the performance of the equipment by pooling of storage and peripheral equipment among central processing units.

Equipment pooling, such as concurrent use of one storage array by two or more central processing units results in delays, called interference, during which one central processing unit waits because the storage array is performing a cycle for another central processing unit. Since the central processing unit is not time-dependent, the loss of time does not cause complications; however, when an input/output device competes with another input/output device for the use of a shared storage, the permissible rate of transmission is affected and an overrun may result. An overrun is any time data transmission is too fast to be used by the central processing unit, or the central processing unit cannot supply data fast enough for the input/output unit.

A system that can perform its entire work load in the presence of any single malfunction is said to be fail-safe, and a system that can perform the essentials of its work load in the presence of a malfunction is said to be fail-soft. Either of these can be accomplished through multisystem installations. A multisystem configuration of a fail-soft system consists of two equal or unequal central processing units with a complement of storage and input/output equipment. This complement of equipment is shared among two or more central processing units, but is considered to be logically independent if they are interconnected by well-defined interfaces so that they can, upon reconfiguration, operate without communicating with each other. Examples of logically independent system components are storage units, central processing units, input/output control units, and input/output devices.

Although logically independent, these system components may still be physically dependent because they share common equipment. Communication between the central processing units of a multisystem may be achieved by transmitting information from one central processing unit to another through a connecting link or by giving them access to a shared storage medium.

Figure 19 shows the major components of a single, or simplex, system, together with their functional dependencies, in simplified fashion. The interconnection between channel and control unit is independent of the particular control units connected or the particular implementation of the channel logic. This interconnection in System/360 is called the input/output interface. In contrast, the interconnection between control unit and device is specialized and differs for tape units, disk files, and communication equipment.

To obtain communication between central processing units in a multisystem system, the storage media and interface connections already available in a simplex system are used. Therefore, the additional equipment necessary to achieve a multisystem operation is minimized. This transmission of information is made possible by the channel-to-channel adapter and the transmission control unit.

A channel-to-channel adapter allows connecting the input/output interfaces of two channels, as shown in Figure 20. The main purpose of the channel-to-channel adapter is to make each channel appear as a control unit to the other channel. Transmission of data between the two channels is by byte at a rate established by the two channels. Because of the standardization of the input/output interface, the channel-to-channel adapter may connect any model of the System/360 to any other model, and may use any type of channel on a given model. Any number of channel-to-channel adapters may be used in a multisystem, but their main function is in a multisystem emphasizing medium reconfiguration time or equipment specialization.

A transmission control unit permits central processing unit communication by private line or common carrier. As indicated in Figure 21, communication is established by means of a specialized device interface rather than via the channel interface. The rate of data transmission is determined mainly by the line capacity. Any two models of System/360, as well as those of any other system, can be connected. The major application of

the transmission control unit is for geographically separated computers.

When two or more central processing units have access to a common storage, information placed in the common storage by one central processing unit can be read by another. In contrast to transmission, sending and receiving are not simultaneous, and a one-to-one relation between recording and retrieval is not necessary. The choice of the shared media is determined by access time, transmission rate, capacity, and cost per bit. Communication differing in application and implementation is achieved by the sharing of disk files, drums, data cells, and tape units.

Shared devices are useful for program restarting information for job recovery upon reconfiguration. Disk, drums, and tape units may be pooled for storage of system programs as well as a means of communication between specialized central processing units to achieve improved turnaround time. Because a control unit normally controls several disk files, drums, or tape units, a switch between channel and control unit allows efficient sharing of a control unit between two central processing units, as shown in Figure 22.

Tape units or other input/output devices are shared between control units, as shown in Figure 23, rather than control units being shared between channels. This choice permits pooling of tape units between control units and permits simultaneous operation of any combination of tape units. This logical ability increases the power of a simplex system, as well as a multisystem. As an example, the sharing of any pair of tape drives by two control units improves the sorting time significantly.

Large capacity storage can be shared, as shown in Figure 24, by two central processing units. When one program is executed by different central processing units, it is desirable to have the locations of instructions and data located in identical addresses in every processor unit. This addressing convention is adopted in System/360 for multisystem operation. The main application of shared storage is in multisystem configurations requiring short reconfiguration time.

The method used to connect the components of a multisystem should be general to permit freedom of choice in system components; expandable, to permit economic systems growth; and reliable, to enhance systems availability.

The System/360 Model 75 channel-to-channel adapter adheres to the input/output interface definition for all System/360's, while the transmission control units adhere to the industry standards for communication equipment. Interconnecting for the sharing of system components is also established between main storage and central processing unit, between channel and the control unit for disk and drums, and between control unit and devices such as tape units. Because of similarities in the logical approach to all three methods, the interconnection of main storage and central processing unit illustrates the discussion.

Figure 25 shows the crossbar switch interconnecting technique that allows connecting of each of M central processing units to any one of N storage units. Figure 26 shows the distributed implementation means of connection. Economic considerations favor the distributed crossbar switch because the need for separate frame and powering is eliminated. High availability is attained without duplication since failure of a switch element is counted as part of a storage unit failure because each storage unit contains its own section of the switch.

A distributed switch proves equally desirable for the connection of control units to channels; control units can be connected through multiple tails to different channels.

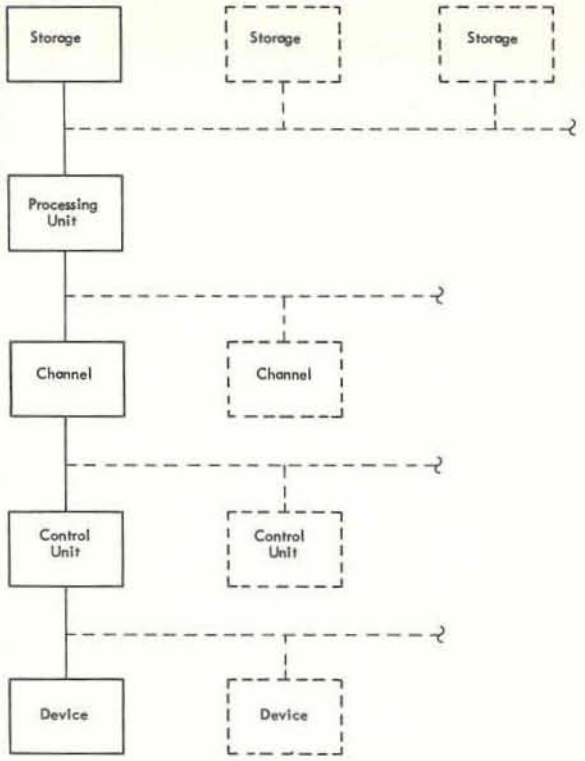


FIGURE 19. FUNCTIONAL STRUCTURE OF A BASIC SYSTEM

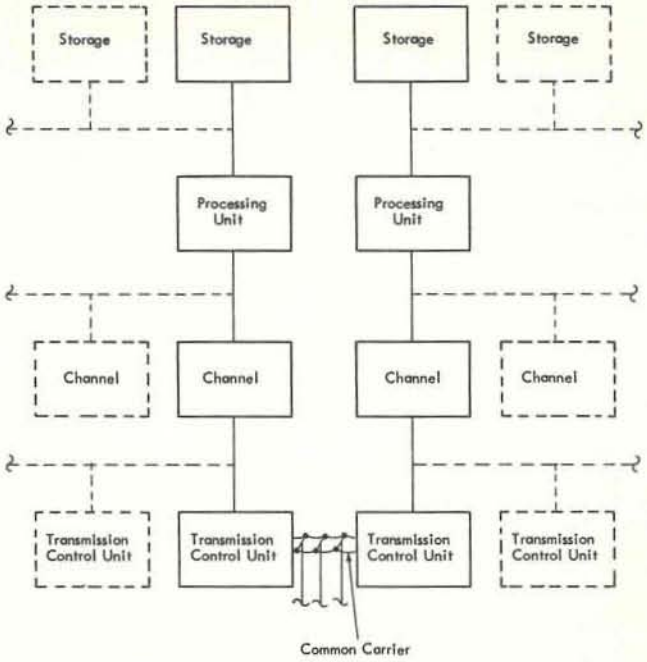


FIGURE 21. TRANSMISSION CONTROL UNITS AS MULTISYSTEM CONNECTORS

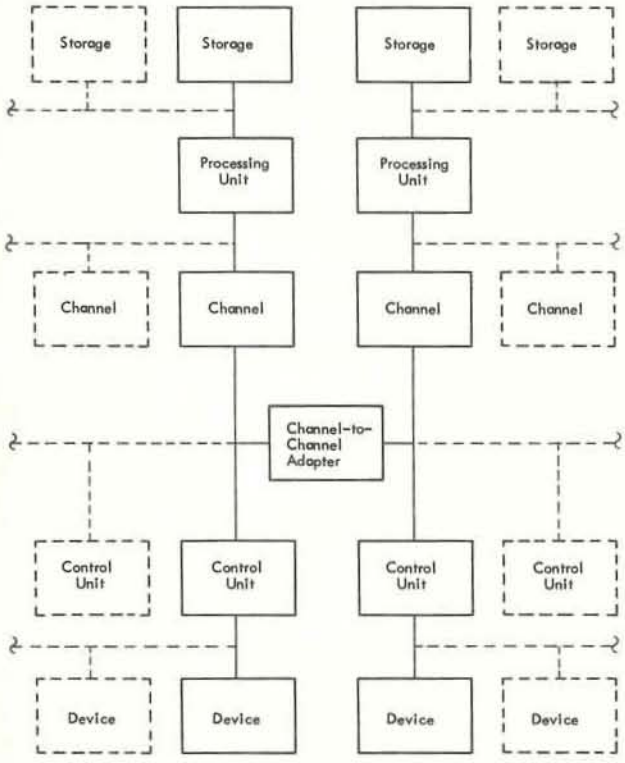


FIGURE 20. FUNCTIONAL STRUCTURE OF A CHANNEL-TO-CHANNEL MULTISYSTEM

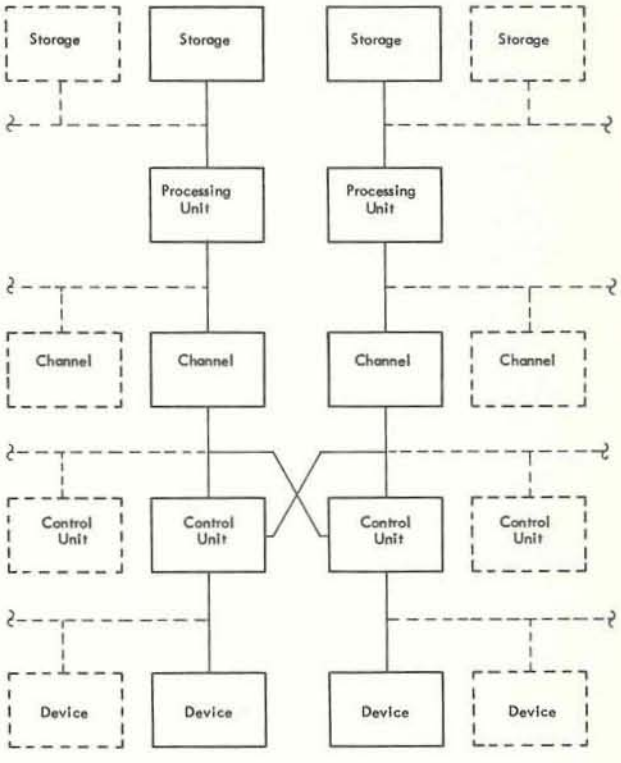


FIGURE 22. SHARED CONTROL UNITS AS MULTISYSTEM CONNECTORS

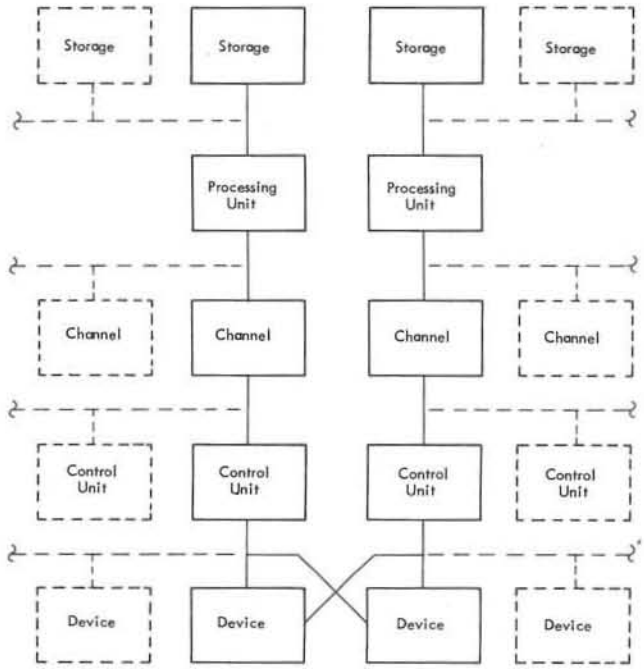


FIGURE 23. SHARED DEVICE AS MULTISYSTEM CONNECTOR

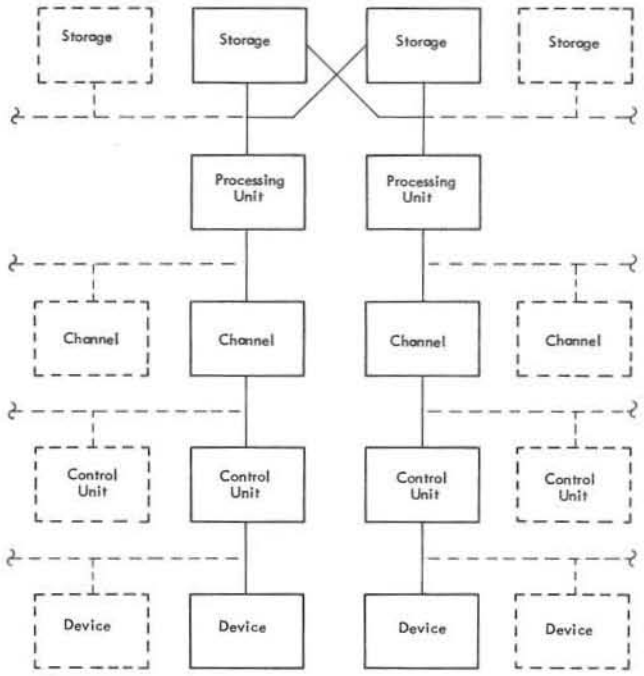


FIGURE 24. SHARED STORAGE AS MULTISYSTEM CONNECTOR

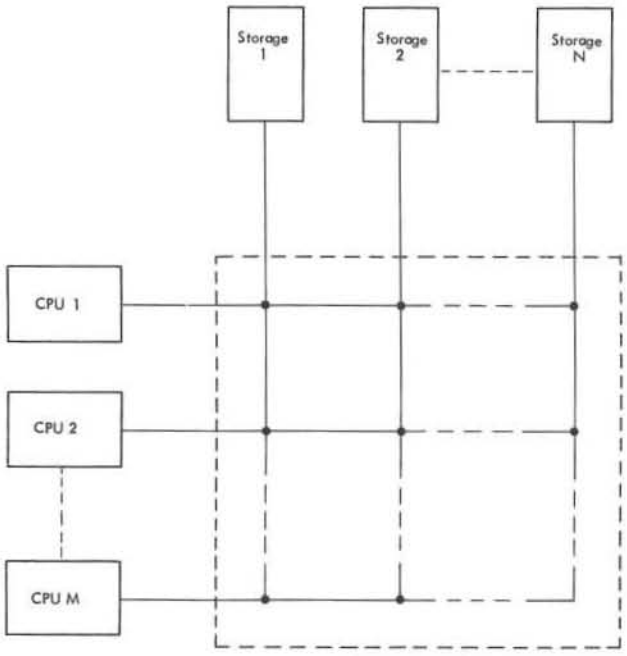


FIGURE 25. CENTRALIZED CROSSBAR SWITCH REPRESENTATION

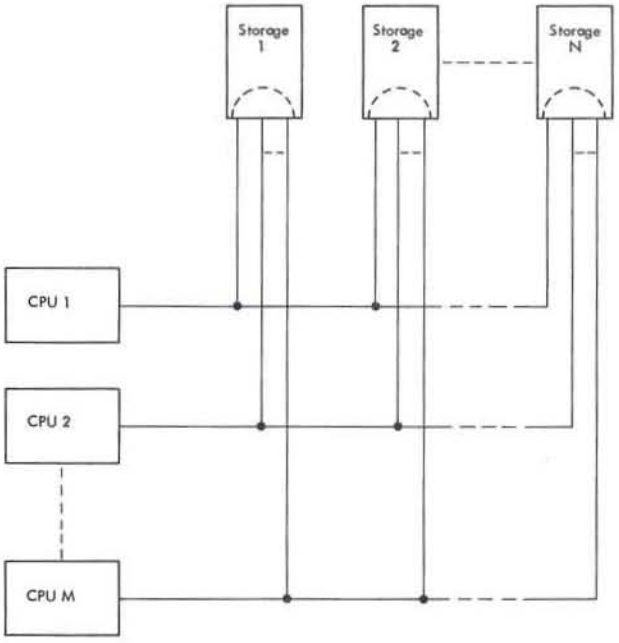


FIGURE 26. DISTRIBUTED CROSSBAR SWITCH REPRESENTATION

- Prerequisite for study of this section is study of Model 75 Systems Introduction.

SYSTEMS CONCEPTS

- The Model 75 System consists of a number of sections.
- Each section does a particular type of work.
- Insofar as possible, sections work independently of each other.
- Six large sections are:
  1. Main Storage
  2. Channels
  3. E Unit
  4. I Unit
  5. BCU
  6. System Control Panel

The Model 75 consists of a number of sections, each of which does its job more or less independent of the other sections. To illustrate, consider a core storage unit (Figure 27). It sits idle until given a start signal. Coincident with the start signal, certain raw material must be available to the unit. This raw material provides the core storage with the details of the job that is being requested, and at the minimum includes the address that is to be affected and whether the data at this address is to be brought out (fetched) or changed (stored). Once started, the core storage unit uses the raw material to do the requested job. It requires no further instructions or control, and it does not interfere with other sections of the system. At the end of its cycle, it gives the requested output or indicates that it has completed the requested job.

Other sections of the system which operate with much the same philosophy as just described for core storages are:

1. Channels
2. Execution (E) unit
3. Instruction preparation (I) unit
4. Bus control unit (BCU)
5. Main storage (one or more core storages)

The jobs done by each of these four sections can be likened to jobs of a manufacturing plant (Figure 28).

The receiving and shipping department handles the incoming orders and supplies and the outgoing products without interfering with other operations. Channels do a similar job for the Model 75. Once given a specific command and told to start, channels transfer data between an I/O device and main storage without interfering with other sections of the system.

The production line is the heart of the manufacturing plant. Here, products are made in accordance with sales orders. In our hypothetical plant, it is useful to think of a single production line which is capable of manufacturing a number of products, but can only work on one product at a time. The production line, then, becomes the limiting factor, or the bottleneck, of the factory.

The E unit resembles the production line because it does the actual job called for by an instruction,

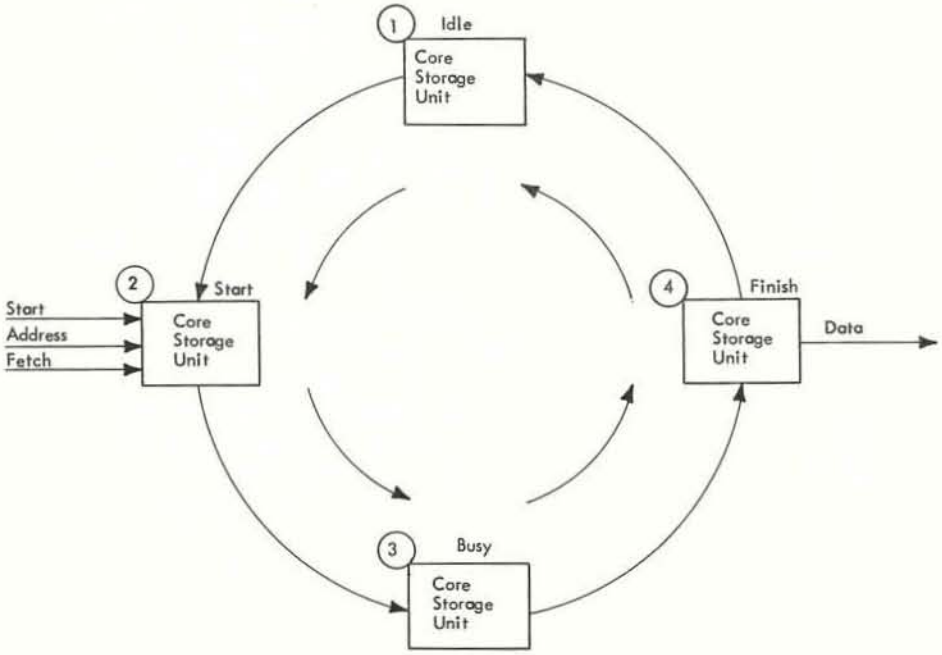


FIGURE 27. CORE STORAGE CYCLE

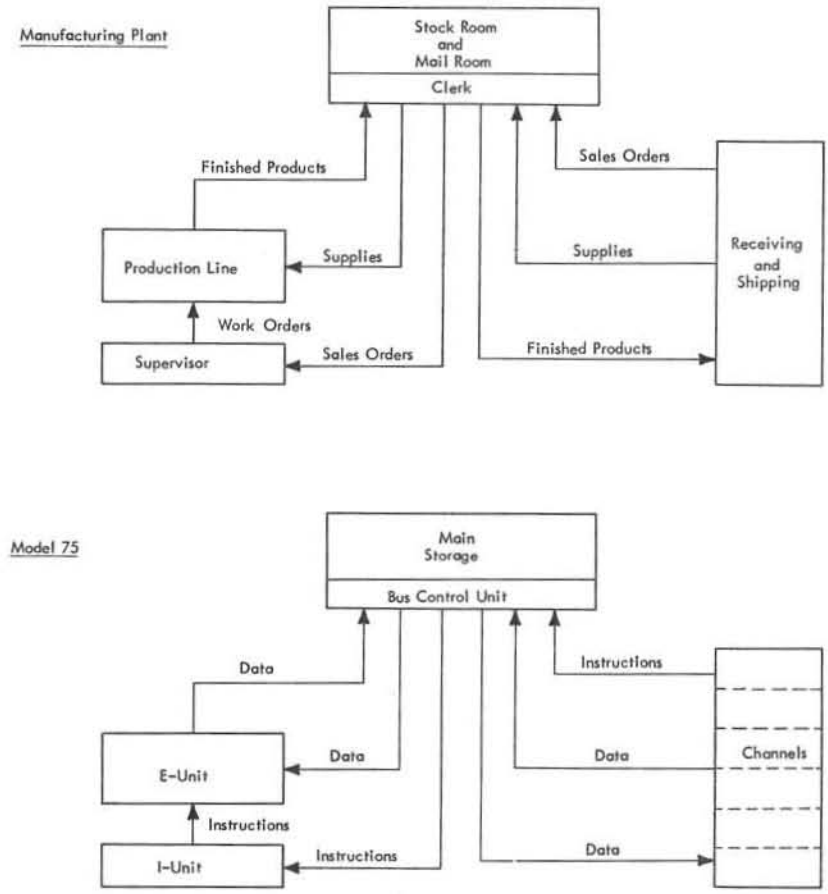


FIGURE 28. MODEL 75 WORKING AREAS

such as add or multiply. Once told the job it must do and given the data (operands) on which the job must be performed, it works independently of other system sections. However, the E unit, like the single production line, can handle only one job at a time, and therfore, tends to be a bottleneck in the total number of jobs that can be done in a given amount of time. This situation of a single production line (E unit) is necessary because the steps of a program (normally) are meaningful only when executed sequentially. For example, a partial result developed by one instruction is often used as an operand in the very next instruction. Execution of these instructions out of sequence would give a meaningless answer.

Because the E unit, or production line, is the apparent bottleneck, a supervisor, or instruction unit, is used to keep the E unit busy. The I unit eliminates delays between jobs by starting a new job coincident with the end of the last job.

As a supervisor interested only in getting the maximum work out of the production line, the I unit does the routine work of getting orders (fetching instructions) and ordering the materials (fetching operands) that the production line will need. Although the I unit works very closely with the E unit, I unit jobs are performed virtually independent of other system sections.

The channels, the E unit, and the I unit all require access to main storage. Because these sections are working independently of each other, requests come at random, much as stockroom requests might come from the various departments in a plant. Notice that a clerk handles stock requests (Figure 28). He takes care of requests one at a time. When several departments require service simultaneously, he takes care of the more urgent requests first. The Bus Control Unit within the Model 75 handles the clerk's job, and does its work independent of other system sections.

The Bus Control Unit has an extensive routing, or switchboard, job (Figure 29). Storage requests can be generated by all of the channels, by the I unit and by the E unit. Further, any request can be for any storage unit attached to the system. Although Figure 29 shows only two storage units, there may be four, eight, or even more attached to the system.

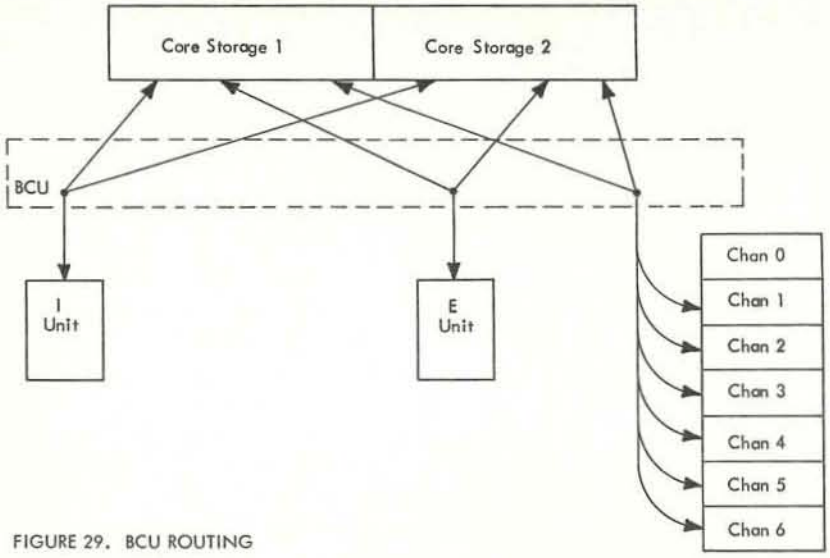


FIGURE 29. BCU ROUTING

Routing data is only part of the service performed by the BCU. The total job consists of taking storage orders and executing them. Consider the clerk in the factory (Figure 28). The production line section takes a finished product and gives it to the clerk. While the clerk is putting the item in its place, the production line is free to do other work. Similarly, the supervisor may call upon the clerk to deliver supplies to the production line. The clerk takes the order, fetches the requested supplies, and delivers them to the production line. Thus, the supervisor has only to place the order, then he is free to do other work. The BCU does exactly the same job as the clerk. It accepts an order to store or fetch a storage word, then frees the requester from further responsibility.

Another Model 75 section which initiates storage requests is the System Control Panel (Figure 30). The system control panel has all of the manual controls and indicators for operator and CE communication with the system. As shown in the figure, the system control panel is connected to the BCU as if it were another channel.

To summarize the discussion:

1. The Model 75 System consists of a number of sections.
2. Each section does a particular type of work.
3. Insofar as possible, sections work independently of each other.
4. Six large sections of the Model 75 are:
   a. Main Storage - two or more core storage units. Main storage stores instructions and data for the system.
   b. Channels - one to seven channels handle data transfers between main storage and I/O devices such as tape units and card machines.
   c. An Execution Unit - performs the actual job called for by an instruction.
   d. An Instruction Unit - keeps the E unit busy by readying new instructions and handling routine tasks.
   e. A Bus Control Unit - handles storage requests for the E unit, the I unit, and all channels. The BCU routes requests, addresses, and data, assigns priority, and frees the requesting area during the actual storage cycle.
   f. A System Control Panel - allows an operator or a CE to communicate with the system.

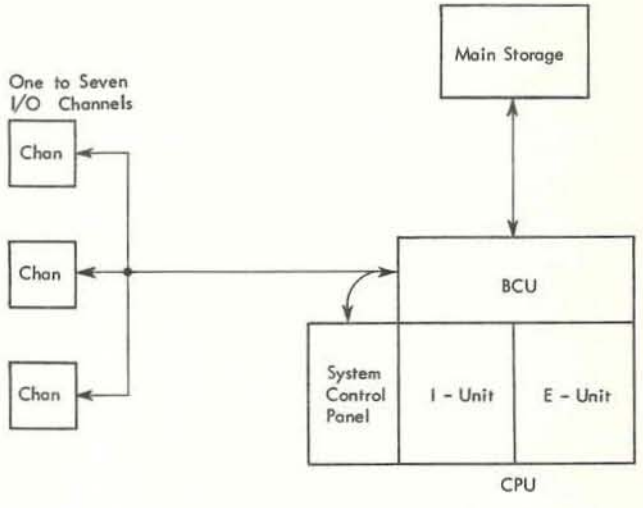


FIGURE 30. SYSTEM/360 MODEL 75

CENTRAL PROCESSING UNIT (CPU)

- CPU consists of an E unit to execute instructions, an I unit to prepare instructions for execution, a BCU to store and fetch instructions and operands from main storage, and a system control panel for operator communication with the CPU.

- E unit is further divided, later.

In general, the Model 75 CPU consists of the E unit, the I unit, the BCU, and the system control panel, although the BCU is often considered a part of the I unit because of physical packaging (Figure 30). A further breakdown of the E unit is necessary when the execution of particular instructions or groups of instructions is examined in more detail. First, however, consider the general way in which instructions are handled, remembering that the BCU, main storage, and each channel can be operating simultaneously with, and independent of, the I and E units.

Instruction Handling

- Simultaneous preparation and execution speeds running of program.

- Most I unit jobs are automatic and independent of the specific instruction to be performed.

- E unit jobs are directly controlled by instruction operation code.

- E unit cycles are controlled by triggers, called sequencers.

- There are many sets of sequencers, one set for each job-type.

- I unit keeps E unit busy.

- I unit is controlled by one set of two sequencers (T1 and T2).

- I unit fetches instructions, updates the instruction counter, and delivers operands to the E unit.

- I unit starts one of three execution units:
  1. E unit
  2. I-E unit
  3. Branch unit

- Only one execution unit operates at any one time, except when two are required to execute a single instruction.

The 2075 prepares instructions for execution in one unit and executes them in another unit, which greatly speeds up the running of any program. The use of different units enables simultaneous preparation and execution. While instruction one is being executed, instruction two is being prepared (Figure 31).

The instruction preparation unit (I unit) performs all functions that are not directly dependent upon the particular instruction being processed. For instance, preparation includes instruction fetching. No matter what the instruction, it must be fetched from storage. The execution unit (E unit) performs the specific operation called for by the instruction being executed. For instance, on a divide instruction the E unit divides, and on an add instruction, it adds.

The functions performed by the E unit are completely determined by the stored program. The I unit, however, automatically and without control from the stored program, performs all of those functions which are necessary for the running of any program. The functions of the E unit might be thought of as the many different problems that we must conscientiously attack and solve in the course of a day's work. The functions of the I unit are the work habits which enable us to do a great deal of our work without conscious effort.

Before being more specific about what I unit does, consider the E unit operation (Figure 32). Executions are done one at a time. The E unit cannot began a new execution until it is completely finished with the last execution. Before an execution starts, the operation code of the instruction to be executed is in the E unit operation register (EOP).

For most instructions, operands are also delivered before the execution starts. During the execution, the required operations are performed on the operands, and the results are delivered to a specified location from the K register.

On each machine cycle of an execution, the data flow and the operation performed are guided by a trigger called a sequencer. A series of sequencers is used for each execution. A different set of sequencers is available for each class or type of job to be performed. The sequencers to be used for any execution are determined by the instruction to be executed. An execution may be a simple move from one general register to another requiring two E cycles, or it may be as complex as a VFL divide in which both operands come from storage and the result is returned to storage. This latter execution may require hundreds of E cycles. In all cases, some selected sequencer designates the first cycle and another the last cycle of each execution. Because executions are done one at a time, the first cycle sequencer (E 1) for an execution may not come on

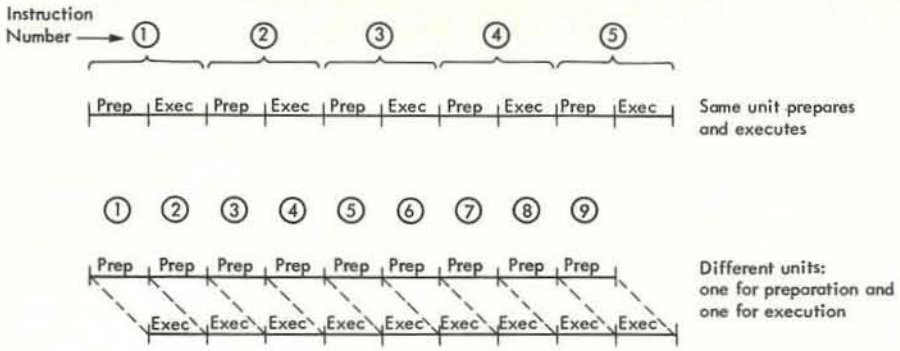


FIGURE 31. SIMULTANEOUS PREPARATION AND EXECUTION

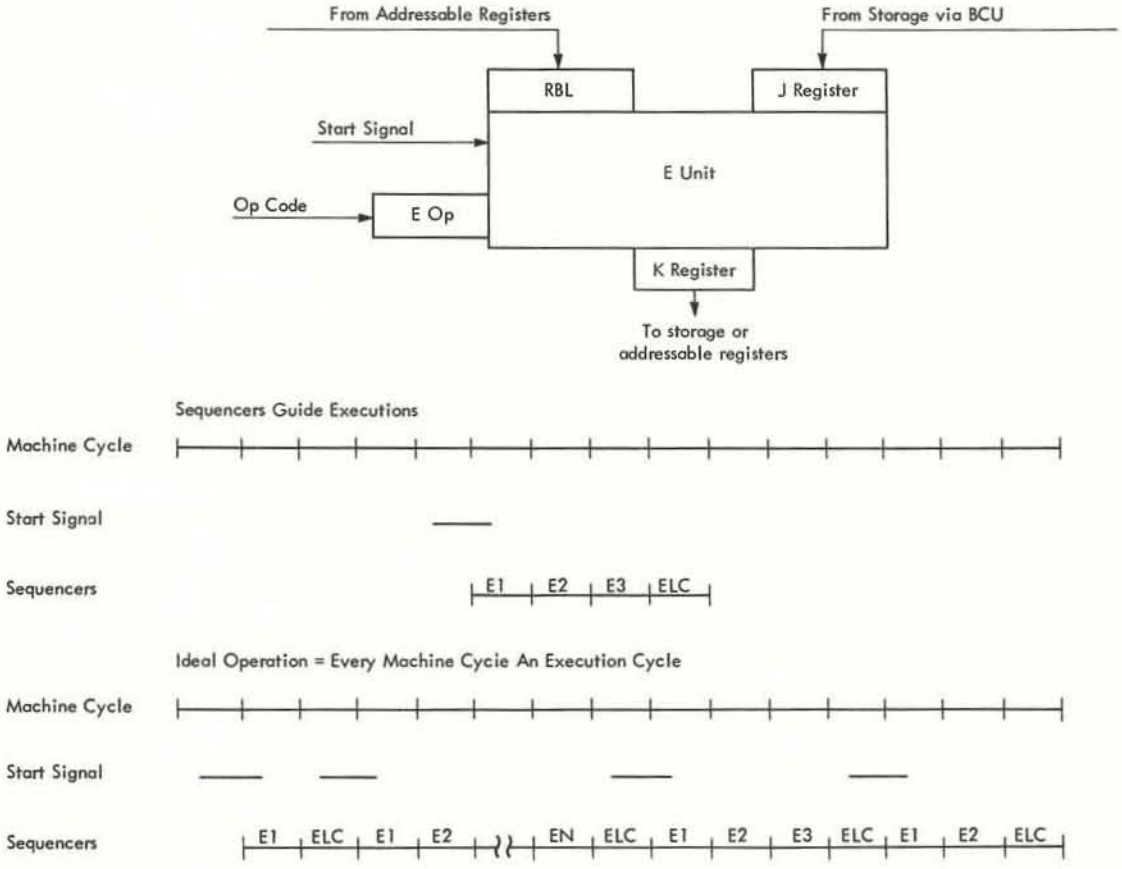


FIGURE 32. EXECUTION AND EXECUTION SEQUENCERS

until the last cycle sequencer (ELC) for the preceding execution has gone off.

For any program, the shortest running time is achieved when every machine cycle is an execution cycle. An I unit performs all preparatory functions in a way that enables continuous execution. This ideal performance is shown at the bottom of Figure 32.

Most preparation functions are sequencer-controlled, as are the execution functions. Unlike executions, however, one set of two sequencers is used for the preparation of all instructions. Before the preparation sequence can begin, instructions must be brought from storage to the processor. I unit contains instruction buffers and a set of mechanisms for keeping instructions available in the buffers. The mechanisms controlling instruction fetches monitor many conditions. Instruction fetches are not made when they will interfere with executions or other preparation functions. Generally, however, new instructions are fetched before all buffered instructions are used.

With instructions available in the buffers, the sequencer-controlled preparations are started. Under ideal conditions, sequencer-controlled preparations are completed in two machine cycles. The preparation sequencers T1 and T2 guide the delivery of operands to the execution unit, the setting of the operation register in the E unit (EOP), and the sending of a start signal to the E unit. Housekeeping functions such as updating the instruction counter and controlling the gates from the instruction buffers to the operation register are also guided by T1 and T2. The start signal is sent to the E unit as the T2 functions are completed.

Most execution sequences are longer than two machine cycles but some require only two cycles. Assuming two-cycle executions and otherwise ideal conditions, instruction fetches, sequencer-controlled preparations, and executions will all proceed simultaneously and without interfering with each other as shown at the bottom of Figure 33.

As previously mentioned, a further breakdown of the E unit is necessary to describe specific executions. The I unit sees three different sections that perform executions (Figure 34). The E execution unit performs on executions requiring arithmetic or logical manipulation of data. The instruction execution unit (IE unit) performs on executions that are closely associated with the I unit functions or functional units. The branch unit performs on executions that may result in a branch to a new instruction address. Each of these units executes certain instructions independently of the other execution units. Each receives its own start signal from the I unit and uses its own sequencers to control its operation much as has been described for the E unit.

The use of three independent execution units does not enable more than one instruction to be executed at any one time. The only simultaneous operation of execution units is when a single instruction requires the use of two units for its execution. On these few instructions, the E unit operates simultaneous with either the branch or the IE units. For instance, on branch on index-high (BXH) the E unit does arithmetic to determine the success of the branch and the branch unit fetches instructions from the branch address.

Bus Control Unit

- BCU is the clerk for main storage.
- BCU honors requests in a fixed-priority scheme.
- A request to BCU is an order to fetch or store a 72-bit word.
- Requester is released as soon as BCU honors request.
- BCU returns fetched words to the proper register.
- BCU overlaps storage requests.

The BCU is the Model 75 clerk for main storage. Whenever the E unit, the I unit, any channel, or the system control panel requires a storage reference (a store or fetch to main storage), a storage request is sent to the BCU. The BCU honors these requests one at a time according to a fixed priority scheme. For each request, BCU must start (select) the proper storage address, and route the data either to or from the selected storage unit (Figure 30).

A storage request to BCU is actually an order or command, to store or fetch a 72-bit word (64 data bit plus 8 parity bit) from main storage. As soon as the BCU begins to execute this order, the requesting area is free to do other work. For example, consider a fetch request from the I unit (Figure 35). BCU will attempt to honor this request every machine cycle. However, it may be several cycles before the request is actually honored. For example, all channels have a higher priority than the I unit. If a channel is ready to start storage, the I unit request is blocked. Also, storage may be busy, forcing I unit to wait.

When there are no conflicts, BCU will select the storage requested by the I unit and inform I unit by sending an accept pulse. The accept signal tells the I unit that its request is being serviced. The I unit is now free to drop its signals to the BCU and proceed with other work.

When the storage unit delivers the 72-bit word from the requested address, the BCU routes this word into the J register.

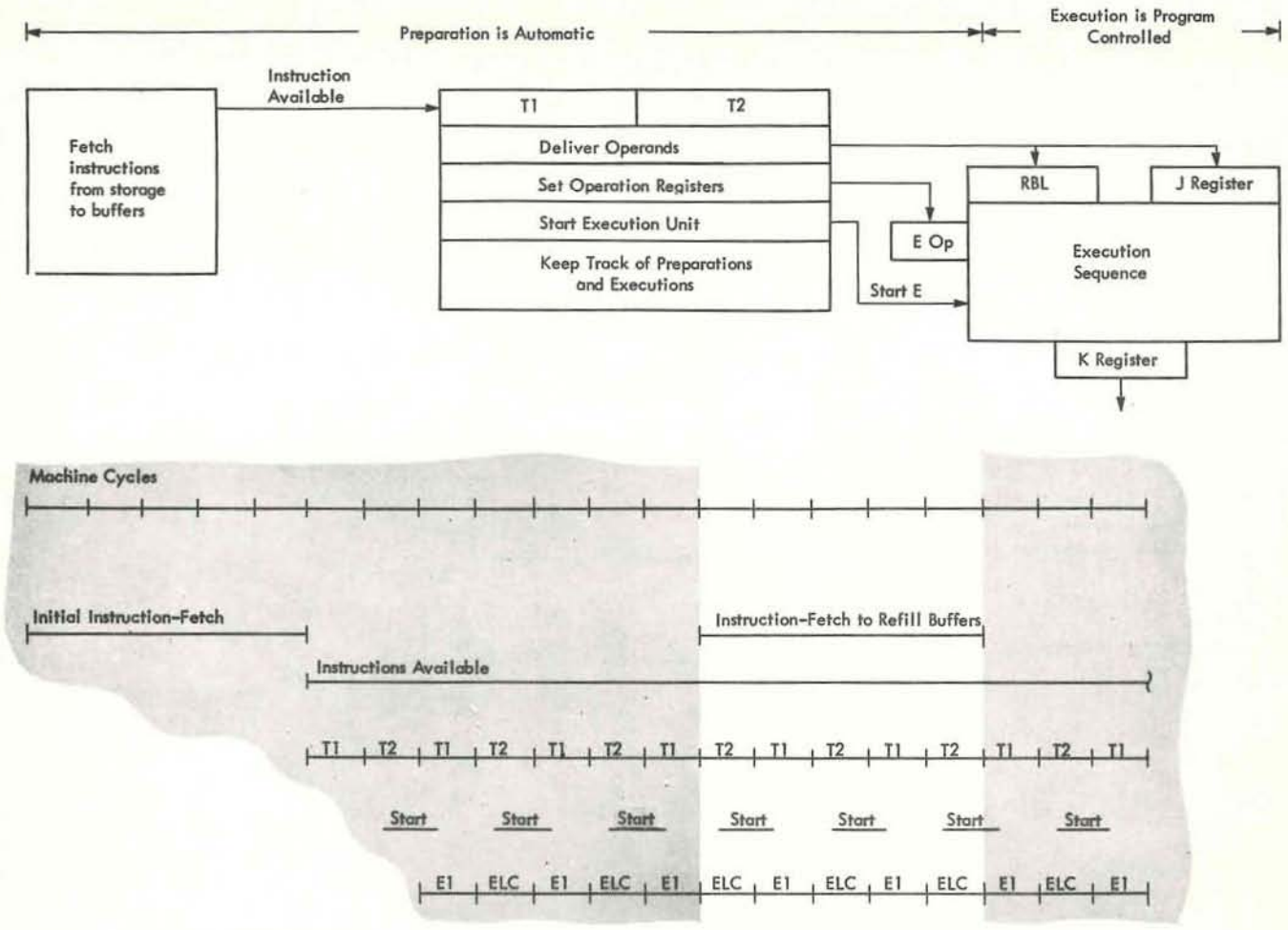


FIGURE 33. SIMULTANEOUS INSTRUCTION FETCHING, PREPARATION, AND EXECUTION

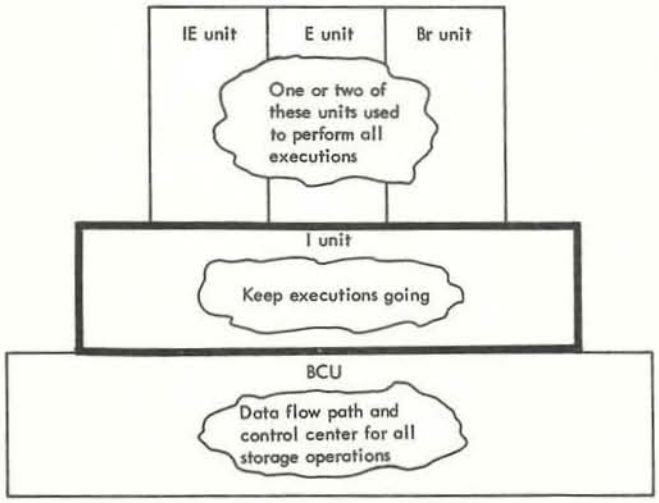


FIGURE 34. FUNCTIONAL SECTIONS OF THE 2075

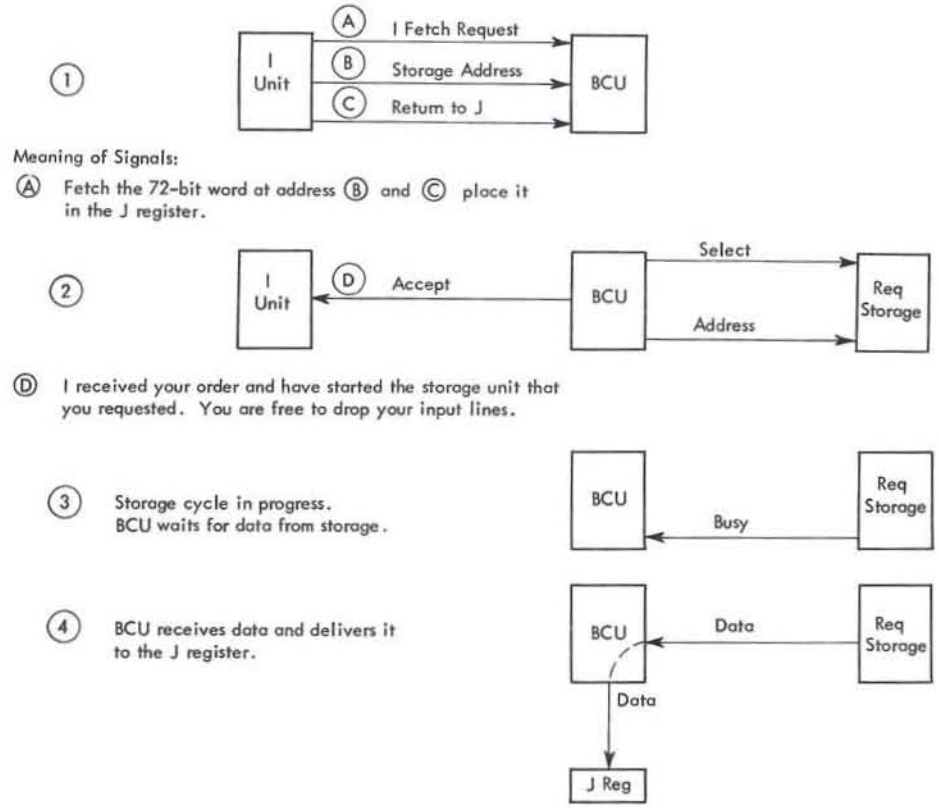


FIGURE 35. I-UNIT FETCH

The BCU overlaps the operation of any two core storage units within main storage (Figure 36). To overlap storages, the BCU must remember two return addresses and associate these addresses with the correct storage cycles (Figure 37).

The actual make-up of main storage is one, two or four 2365 Processor Storage Units (Figure 38). Each 2365 has two independently operating storages known as high-speed storages (HSS) or M-4's. The addressing scheme is such that one HSS contains even addresses and the other HSS contains odd addresses. This method of addressing is called "interleaving." An HSS storage cycle (Figure 39) requires 750 nanoseconds. Fetched data is delivered to the BCU from about 400 ns to 600 ns of the cycle.

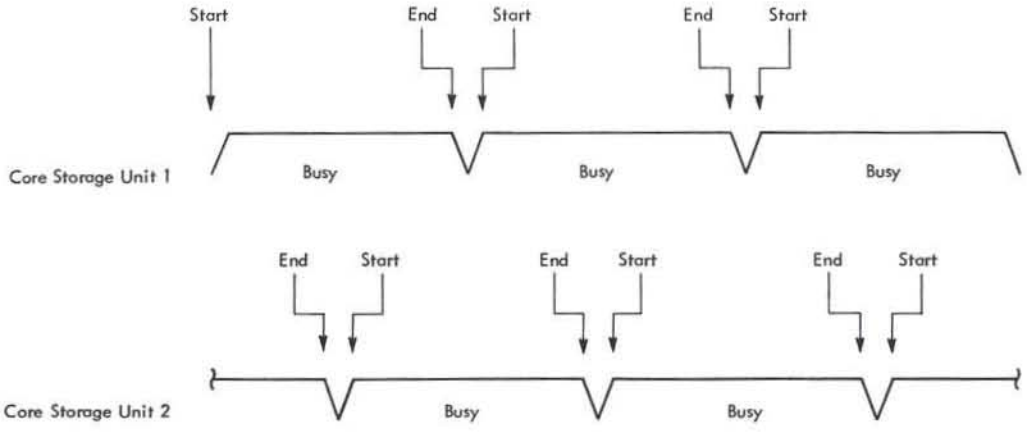


FIGURE 36. OVERLAPPED STORAGE CYCLES

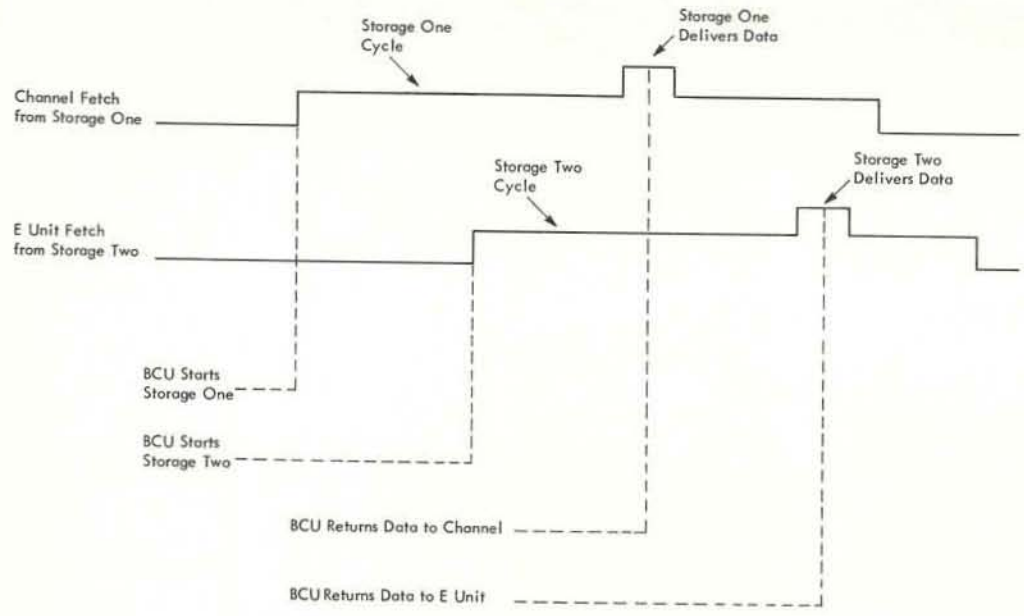


FIGURE 37. RETURNING DATA WITH OVERLAPPED STORAGES

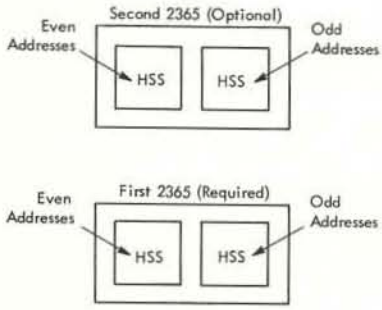
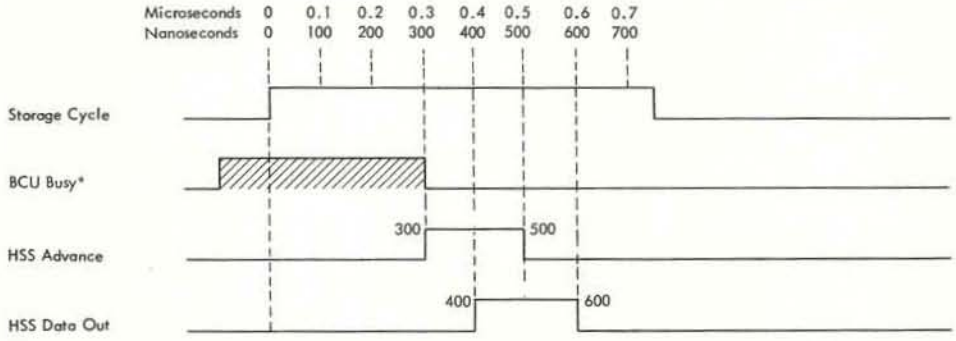


FIGURE 38. MODEL 75 MAIN STORAGE



* Approximate time BCU is busy with this selection.

FIGURE 39. HSS CYCLE

System Control Panel

- Panel contains lights and switches for communication with system.
- Lights are grouped into logical areas such as I controls, bus control, VFL, etc.
- Error indicators are grouped into one section.
- Four sections of controls:
  1. Operator control
  2. Operator intervention
  3. Customer engineer intervention
  4. Power control
- Operator control section allows:
  1. Power on/off
  2. Initial loading of programs and data
  3. Manual interrupt
- Operator intervention section allows:
  1. Start, stop, and reset CPU
  2. Manual store and display
  3. Set machine registers
  4. Single-step and single-cycle.
- Customer Engineer intervention section allows:
  1. Running of FLT's.
  2. Forcing of CPU logout.
  3. Disabling error-checking circuits.
  4. Forcing parity-errors.
  5. Changing main storage addressing scheme.
  6. Repeating an instruction or a group of instructions.
  7. Stopping the CPU or storage on an error.
- Power control section allows:
  1. Main system power-on
  2. Checking the status of power on units of the system.
  3. Marginal checking.

The systems control panel contains lights and controls for operator and CE communication with the system. The lights show the contents of most CPU registers and the status of most CPU control triggers. The controls consist of switches and keys to control power, to start, stop, and reset the CPU, to enable or disable various maintenance features, and to perform manual operations such as store, display, and load CPU registers.

Control trigger indicators are grouped, with the groups labeled according to logical areas of the machine such as I control, bus control, VFL, etc. A group of red indicators that show check conditions are generally turned on by parity errors.

The switches and keys on the control panel are divided into four sections:

1. Operator control.
2. Operator intervention.
3. Customer Engineer intervention.
4. Power control.

The operator control section contains the necessary controls for the operator to turn the machine on or off, to initially load programs and data into the machine, and to manually interrupt the CPU.

The operator intervention section of the panel contains the switches and keys that permit the operator to stop, start, or reset the machine; to manually store instructions or data into any storage location or into any of the addressable registers; to set up an entire new PSW in the PSW register or to set up the IC portion of the PSW register; to start processing over again, from the very beginning, by reloading the PSW from storage location 0; to display the contents of any storage location or of any addressable register; and to manually step through a program, either one instruction at a time or one machine-cycle at a time.

The customer engineer intervention section of the panel has controls to permit the CE to:

1. Start, stop, and control the running of fault locating tests (FLT) for maintenance purposes.
2. Force a CPU log-out operation to store the status of 1216 CPU triggers (both register positions and control triggers) into main storage.
3. Load the instruction buffer (AB) registers. The registers are set from the data keys which are in the operator intervention section of the panel.
4. Disable most errors.
5. Force certain parity errors.
6. Change the main storage addressing scheme.
7. Enable the system control panel data keys to be used in lieu of a main storage location.
8. Repeat one instruction or repeat a group of instructions.
9. Stop storage on an error condition to isolate a failing storage unit.
10. Stop the CPU on an error condition.

The power control section contains the controls concerned with system power, including marginal check controls, thermal controls, and emergency controls. Also included in this section is a key to test panel indicator lamps by turning all lamps on.

Machine Cycles

- Cycles are 195 nanoseconds in duration.

- Pulses are: A, early B, B, late B, L and C clock.
- B pulse is often timed for a specific job.
- L pulses overlap A pulses.
- L pulse holds data at origin while an A pulse samples the data into a receiving register.

A machine cycle is 195 nanoseconds in duration and begins with an A clock pulse (Figure 40). The timing of the B clock pulses is not always as shown in the figure. A delay line is often used to shift a B clock pulse for the most advantageous timing. The Significance of the L, or latch, clock is that it completely overlaps an A clock. Generally, data is held (latched) at an origin register by an L clock while it is sampled into a receiving register with an A clock.

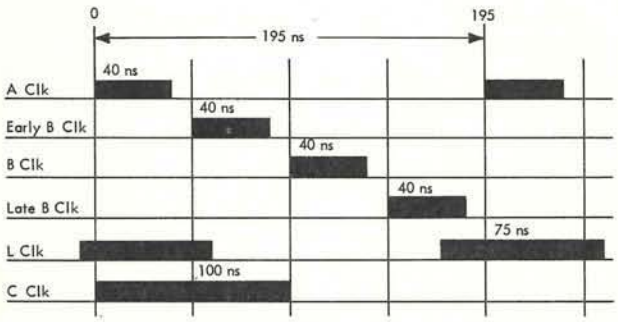


FIGURE 40. CPU MACHINE CYCLE

Fault Locating Tests

- FLT are an automated set of tests to check CPU logic circuits.
- About 100,000 tests are stored on an FLT tape.
- Special FLT circuits bring tests into CPU and execute them.
- A listing correlates failing test numbers with circuit cards that can cause the failure.

Fault Locating Tests (FLT) are an automated set of tests to check CPU logic circuits. There are about 100,000 fault-locating tests. The tests are stored on tape and brought into the CPU one at a time with special FLT circuits. Once started, the tests run one after the other until a failing circuit causes testing to stop. When testing stops, the failing test number is displayed in the test register. An FLT listing correlates the failing test number to the circuit card or cards that can cause this test to fail. The card or cards listed are replaced to repair the trouble.

The FLT's do not test the CPU functionally as does a diagnostic program. For example, a diagnostic program would probably test the multiplying circuits by actually multiplying two numbers and comparing the product to a predetermined result. The FLT's, however, test the multiplying circuits (and all other circuits) one element at a time. Each trigger is turned on and then checked to see that it did turn on; each trigger is turned off and then checked to see that it did turn off. Where practical, each AND and OR circuit is conditioned and then checked to see that the output is correct.

Each test on the tape tests a single element or a small group of elements for a single condition or output. For example, 72 separate tests from the tape checks the 72 triggers of the J register to see that they can turn on. Seventy-two other tests check the same triggers to see that they can turn off. After all of the triggers are tested (not only register positions but also control triggers), logical groups of circuits are tested, such as adders, decoders, etc.

To test a group of logic, the inputs to the logic must be properly conditioned so that a test of the output will be meaningful. Each test on tape, therefore, sets up the CPU to the status (certain triggers turn on) that will result in a particular output of the logic in question. This output is one of the indicators on the system control panel. Any one of the 1216 indicators on the panel can be selected for the test result indicator. After the CPU is set up (called scan-in) for the test, the proper indicator is selected and its status is compared with the expected result bit in a register that was also set as part of the scan-in operation. If the comparison shows equal, the test passed; if not, the test failed.

Many of the circuits tested cannot have their inputs conditioned directly by the scan-in or their output checked directly for the test. These circuits require that after scan-in, a number of machine cycles must occur before an indicator will reflect a true result of the circuit being tested. Therefore, after the scan-in, the controlled clock is started and

and allowed to run for a predetermined number of cycles. Then the comparison is made between the selected indicator and the expected result bit.

Interrupts

- Interrupt system switches programs for unusual or special conditions.
- An interrupt is accomplished by replacing the PSW.
- The five classes of interrupts are: external, program, machine check, supervisor call, and I/O.

Certain system conditions such as program errors and machine errors require immediate programming attention. These and many other unusual or special conditions are handled by the interrupt system. The interrupt system is a means of switching the CPU from executing one program to executing another program. Unlike branching, interrupt switching of programs is asynchronous with program execution. For example, an interrupt signal is generated whenever an I/O device completes an assignment. This interrupt is used to allow the CPU to keep an I/O device operating continuously. The general sequence of events is:

1. An I/O program is executed and results in the starting of some I/O device.
2. The CPU begins another program which is not dependent on the I/O operation just started.
3. After an indeterminate time, the I/O device finishes its operation and signals "interrupt."
4. The CPU leaves the program it is presently executing and goes back to the I/O program to start another I/O operation.
5. After restarting the I/O device on a new operation, the CPU branches back into the non-I/O program to continue from the point at which it was interrupted.

The interrupt scheme accomplishes steps 3 and 4 to switch to a program that corresponds to the type of interrupt signal which occurred. The interrupt scheme also records the necessary data to allow the programmer to switch back to the correct point of the interrupted program.

The interrupt process accomplishes the program transfer by replacing the current PSW register contents with a new PSW. The new PSW establishes the new status of the CPU, including a new instruction count (IC) which is the starting point of the new routine to be executed. Just before the current PSW is replaced, the interrupt code field of the PSW register is set according to the type of interrupt condition that was detected. Then, the entire current PSW is stored away and the new PSW is fetched from storage and set into the PSW register. Next, an IC recovery takes place, fetching a new stream of instructions from the transferred-to program.

Generally, interrupts transfer to program routines in the operating system program. The interrupt hardware simply loads a new PSW; from this point, the operating system program examines the stored-away PSW to determine what caused the interrupt and then takes appropriate action. This action might be to process a real-time job because of an external signal from another computer; or, to call in an analyzing program because of a machine check; or, to terminate the problem because of a program check; etc. After the necessary action by the operating-system program, the interrupted program may be resumed by the execution of a load PSW instruction, which loads the PSW register with the PSW that was stored away at the time of the interrupt.

There are five classes of interrupts:

1. External
2. Program
3. Machine check
4. Supervisor call
5. Input/output

All of the interrupts (with two exceptions) fall into one of these classes. Each class is distinguished by the fixed storage locations in which the current PSW is stored and from which the new PSW is fetched. The two exceptions are initial program load (IPL) and timer advance request. These two procedures use the interrupt circuits to accomplish their objectives, but do not result in the exchange of PSW's.

External: This class includes (1) timer word overflow, which occurs when the timer word goes from a positive to a negative value, (2) interrupt key on the system control panel, and (3) any of six external signals. When any of these three types of interrupt conditions occurs, the CPU takes an external interrupt, which stores the current PSW in the external old storage location (24) and fetches a new PSW from the external new location (88).

Program: This class consists of 19 separate interrupt conditions, all of them being associated with

either the interpretation or the execution of instructions. Any program interrupt results in the exchange of program old and program new PSW's. The fixed storage locations for the program PSW's are: old, 40; new, 104.

Machine Check: This interrupt constitutes a class all by itself. It is always a result of a machine malfunction and results in the exchange of machine check old and new PSW's (old, 48; new, 112).

When a machine check occurs, the CPU clock is stopped and a logout operation takes place. The logout consists of storing the status of most CPU triggers (both register triggers and control triggers) in a fixed area of storage. When the logout operation is complete, the machine check interrupt is initiated.

Supervisor Call: This class includes only the supervisor call interrupt, which occurs when the supervisor call instruction is decoded. The current PSW is stored in location 32 (old), and a PSW is fetched from location 96 (new).

Input/Output: The I/O class includes the interrupts caused by signals from any of the seven channels. An I/O interrupt causes the current PSW to be stored in location 56, the channel status word to be stored in location 64, and the new PSW to be fetched from location 120.

In addition to the slots reserved for old and new PSW's, the first three double word locations are reserved for IPL use, location 64 is reserved for the channel status word, location 72 for the channel address word (single word), and location 80 for the timer word (single word). The 19 double words starting in location 128 are for logout, which follows any machine check error.

Interruptible Status

Certain positions in the PSW, called masks, determine the interruptible status of the CPU. If a mask position is a one, the corresponding interrupt source is allowed to interrupt the CPU; if a zero, the interrupt is said to be masked off and the interrupt does not occur. Some interrupt sources, although masked off, remain pending and will be taken should the mask position in question be changed by the introduction of a new PSW. Each new PSW introduced may therefore change the interruptible status of the CPU. Two of the mask fields, system mask and program mask, may be changed independently of the rest of the PSW by the instructions "set system mask" and "set program mask." The mask fields and their effect on interrupts are:

System Mask, PSW 0-6: Each position either allows or masks off the I/O interrupt signal from its respective channel. If mask off, the I/O interrupt request remains pending.

PSW 7: This single position is also a part of the system mask field and either allows or masks off, as a group, all of the external interrupt class. If masked off, the external interrupt remains pending.

PSW 13: This position either allows or masks off machine check interrupts. If masked off, the machine check condition is ignored.

Program Mask, PSW 36-39: These four positions either allow or mask off respectively: fixed-point overflow, decimal overflow, exponent underflow, and significance. If masked off, the interrupt conditions are ignored. Note that of the 19 conditions that cause a program interrupt, only four are controlled by the PSW.

Triggers and Latches

- The AND-OR-invert is the logic building block of the CPU.
- Retention devices in the 2075 are called triggers and latches.
- Triggers and latches are, physically, flip latches (FL), polarity holds (PH), and flip-flops (FF).

The logic building-block of the 2075 is the AND-OR-invert (Figure 41). This unit consists of either of two or three non-inverting plus AND's feeding an inverting plus OR. This AND-OR-invert (AOI) unit is used for most AND/OR logic decisions within the machine and is the heart of most of the retention devices.

The retention devices in the 2075 are called triggers and latches; this nomenclature, however, does not reflect the physical circuits, but rather the set-reset timing of these devices. Devices changed at "A" time are usually called triggers; devices changed during "not L" time are usually called latches. This trigger/latch nomenclature is further explained later in this section.

Physically, most of the retention devices used are either flip-latches (FL) or polarity holds (PH). The basic retention device, shown in Figure 41, is a flip-latch. This device is more readily recognized as a cross-coupled AND/OR when redrawn (Figure 42). Most instruction diagrams (positive logic) represent the FL circuit with a single block, as shown in the figure.

A polarity hold is an FL that meets an additional requirement; set and reset inputs are gated by complementary timing pulses (Figure 43). This configuration has several unique characteristics:

1. A separate reset input is unnecessary. The clock timing pulse turns the PH on or off as dictated by a single input. An additional reset line, however, can be used on a PH.
2. The output follows the input during "release" time.
3. On the transition from "release" to "not release" the latch-back is activated to "lock" the output in its present status.
4. The input has no effect on the output except during "release" time. The device is said to be "locked during not release" (not A clock, in the example).

The PH is used extensively for both data register bit-positions and for control triggers and latches. An example use of the PH circuit is shown in Figure 44. When a PH is released, it assumes the new status of the data line; the PH changes state only if the data is now in the opposite status from that in which it was the last time the PH was released.

Some register positions consist of two PH's (Figure 45). The first, or input PH, is called a trigger and is released by an A clock pulse. The second, or output PH, is called a latch and is released by a "not L" clock. The significance of the latch is that it is locked during L time so that it cannot be changed during the A clock pulses that change the triggers. This scheme allows new data to be set into a register with the same A clock pulse that moves the old data to another register.

Like the double PH register positions, the retention devices that control data movements (control triggers) are usually trigger/latch combinations. These control devices are more fully discussed in the "Sequencers and Sequencer Cycles" section.

Special Retention Devices

- Most triggers operate with A clock timing, but there are exceptions.
- Most latches operate with L clock timing, but there are exceptions.
- Many retention devices require examination of individual circuits and the latch-back arrangement.
- A flip-flop changes to its opposite state upon the application of a single signal.

As previously explained, a trigger in the 2075 CPU is a retention device released (or set) by an A clock; a latch is a device whose output is good (or locked) during an L clock. Although the majority of the retention devices fit one of these definitions, some operate with other timing. For example, there are PH's released at early B (EB), B, and late B (LB). Some of these devices are called triggers; others are called latches. There is no useful distinction between "trigger" and "latch" when a device is timed at other than "A" or "L" clock time.

Physically, there are many variations of the basic FL and PH circuits. Many times the AND/OR circuits and the latch-back arrangement of these retention devices must be examined in detail in order to understand exactly how the device behaves. In ALD's, any AND, OR, or invert block that is a part of a retention device is tagged with the letters "PH" following the A, OR, or N symbol. The PH tag means that the block is a part of a retention device; not that all of these circuits are PH's. Some of the PH and FL variations are explained in the following paragraphs.

Often, a PH is set by more than one data (or control) source (Figure 46). Also, some PH's have a separate reset.

An FL is sometimes composed of two cross-coupled AOI Units (Figure 46).

Retention devices are often used to alter the timing of a signal (Figure 47). The duration may be shortened or lengthed or the signal may be delayed.

Occasionally, there is need for a device which will change to the opposite state upon the application of a single signal: if it is on, turn it off; if it is off, turn it on. This device is logically known as a flip-flop (FF). Reversing the state of an FF is said to be "complementing the FF," or "binary operation."

The 2075 FF's consist of two FL's or two PH's with interconnecting gating (Figure 48). In SMS technology, this circuit was sometimes called a binary latch configuration. The device works on the following principle. One retention device provides the output and is changed to the opposite state each time an active input signal is applied. To accomplish the binary operation of the first device, a second retention device is set to the same status as the first, shortly after the first has been changed. This second device now gates the input signal to the first device so that the next input signal will change the present status of the first device.

Sequencers and Sequencer Cycles

- Sequencers control 2075 operations.
- One or more sequencers are generally on.

AOI -- The Building Block

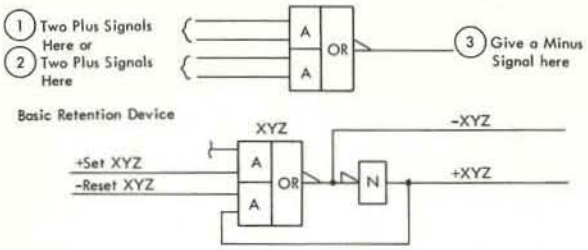
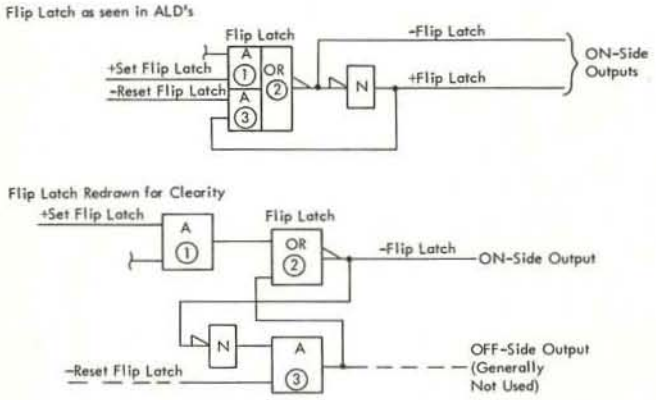


FIGURE 41. AND-OR-INVERT



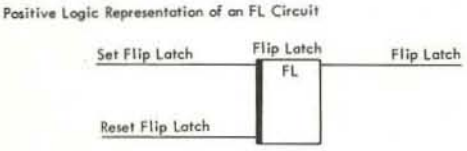
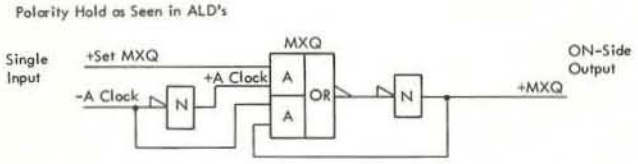


FIGURE 42. FLIP LATCH



(1) During every A clock, the status of the +Set MXQ line is tested.

(2) During an A clock, the +Set MXQ line is directly coupled to the device output; the device is said to be released.

(3) At the end of an A clock, the device output is in the last status of +Set MXQ and is "locked" in this status until the next A clock.

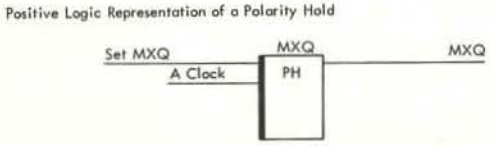
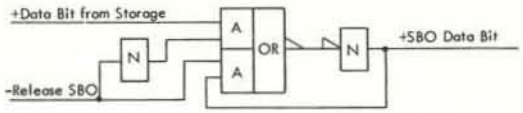


FIGURE 43. POLARITY HOLD



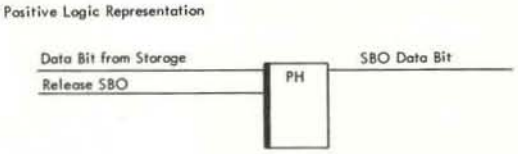


FIGURE 44. EXAMPLE OF POLARITY HOLD USE

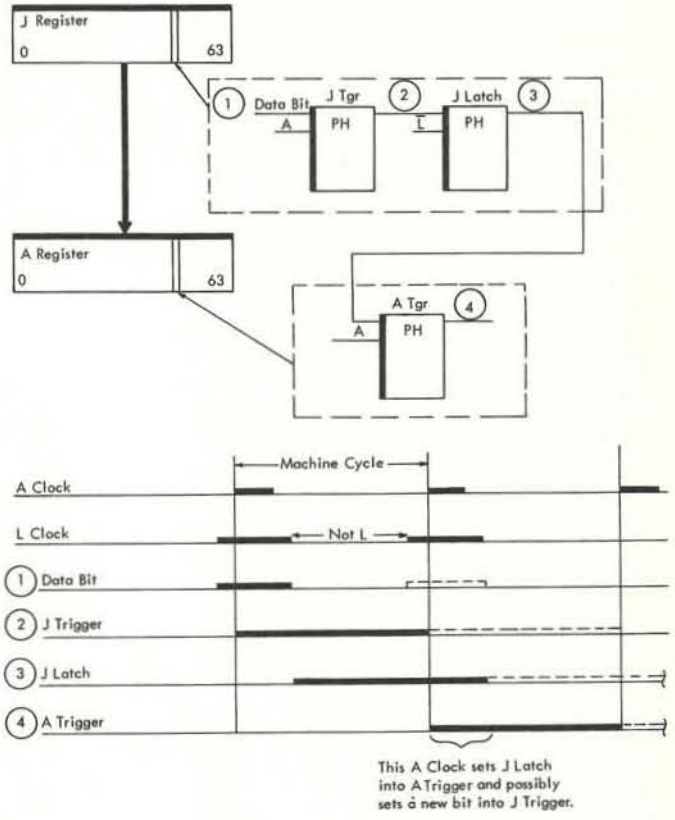


FIGURE 45. POLARITY HOLD REGISTER POSITIONS

- Most sequencers consist of a trigger-latch combination.

- Sequencers are generally arranged in chains.

Operations performed by the 2075 are controlled by sequencers. If no sequencers are on during a particular machine cycle, the CPU does practically nothing during that cycle. Generally, one or more sequencers are on during a given machine cycle. The particular sequencer that is on and the operation code, together, determine the data movements and other functions that are accomplished by the clock pulses of that machine cycle.

Most sequencers consist of a trigger-latch combination quite similar to the trigger-latch combination used for individual bit-positions of some of the data registers (Figure 45). The sequencer trigger is set by an "A" clock; the sequencer latch by "not L."

The sequencers are generally arranged in chains, so that the latch of sequencer one gates the trigger turn-on of sequencer two. An example of a sequencer chain is IE 1, IE 2, IE 3, and IEL. These sequencers control instructions executed by the I execution unit. Another example is a chain of 12 sequencers used for VFL instructions. This VFL chain (and most of the other chains) is not always used in the same way; for example, sequencers one through nine perform for a VFL add/subtract set-up sequence. In this application, the sequencers are referred to as SU 1 through SU 9. VFL sequencers two through seven perform VFL add/subtract store-fetch sequences. In this application, the sequencers are referred to as SF 1 through SF 6.

Each constructive machine cycle is named for the sequencer that exercises control over that cycle. For example, machine cycles occurring while the IE 1 sequencer is on are called IE 1 cycles. If the First FXP (fixed-point) sequencer is on, then the machine cycle is a First FXP cycle. Often, more than one sequencer is on during a particular cycle. For example, the two I unit sequencers, T1 and T2, generally overlap E unit sequencers; that is, a T1 cycle usually coincides with some E unit cycle such as First FXP.

As previously discussed, a machine cycle starts with an "A" clock pulse and ends with the next "A" clock. This corresponds to the timing of a sequencer trigger (Figure 49). This period of time (from A to A) is generally considered to constitute the sequencer cycle, although the sequencer latch timing extends into the next machine cycle. As shown in the figure, a sequencer cycle can be repeated any number of machine cycles. This is often the case when some condition prevents progressing to the next cycle, such as when awaiting requested data to return from storage. A cycle is repeated by leaving the sequencer trigger on and failing to turn on the sequencer latch.

When a sequencer cycle is repeated, the functions of the sequencer are split into three groups: those that occur only on the first sequencer cycle, those that occur on every sequencer cycle, and those that occur only on the last, or "good," sequencer cycle. If the blocking condition for a sequencer latch is not present, all functions of the sequencer (first, every, last) occur on the same machine cycle.

MAJOR UNITS AND DATA FLOW PATHS

- Local storage is transistor registers: 16 general-purpose, and 4 floating-point.

- Three-input addressing adder.

- 72-bit wide data paths.

- VFL section for byte processing.

- Two 72-bit instruction-buffer registers.

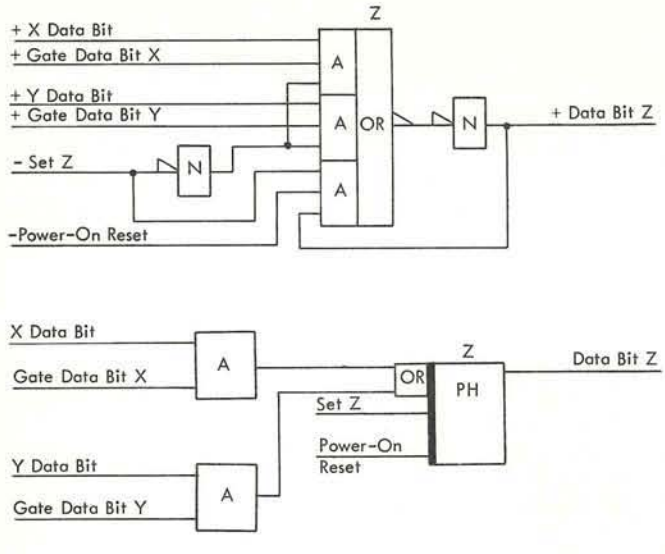
The major functional units of the 2070 CPU are shown in Figure 50. Some highlights are:

1. The local storage (sixteen 32 data-bit plus 4 parity-bit general registers and four 64 data-bit plus 8 parity-bit floating-point registers) are implemented in diode-transistor logic.
2. A three-input addressing adder calculates operand effective addresses in one machine cycle. Effective addresses consist of a base register, an index register, and a 12-bit displacement field.
3. Major data paths are 72 bits wide (64 data-bits plus 8 parity-bits). There are ten 72-bit registers and a main adder that adds two 72-bit operands in one machine cycle.
4. A variable field length (VFL) section to process instructions that require manipulation of single-byte operands.
5. A separate exponent adder to speed floating-point operations.
6. Two instruction buffer registers to hold a backlog of new instructions and thus avoid waiting for instruction fetches.

Instruction Preparation

Preparation consists of fetching instructions, op register loading, instruction counter updating, and local and main storage operand fetches.

Two-Way PH With Separate Reset



Cross-Coupled AOI Flip Latch

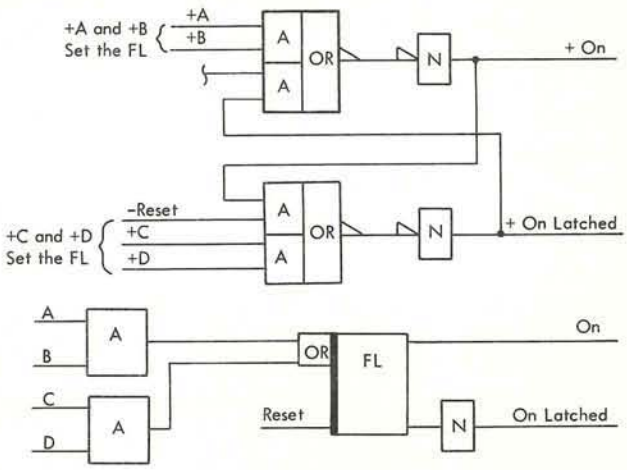


FIGURE 46. RETENTION DEVICE VARIATIONS

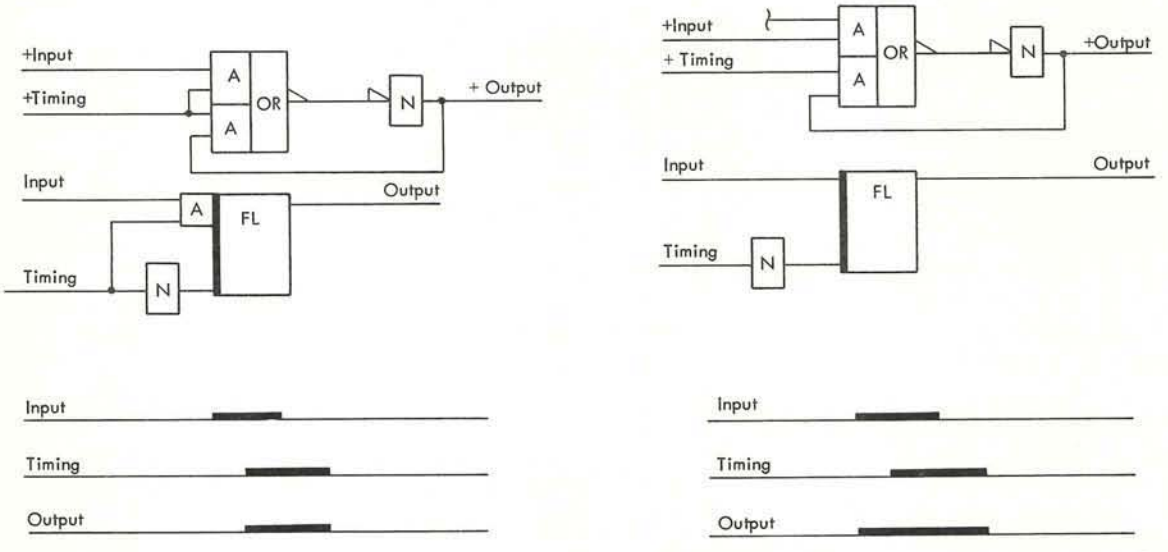


FIGURE 47. FLIP LATCHES USED TO ALTER SIGNAL TIMING

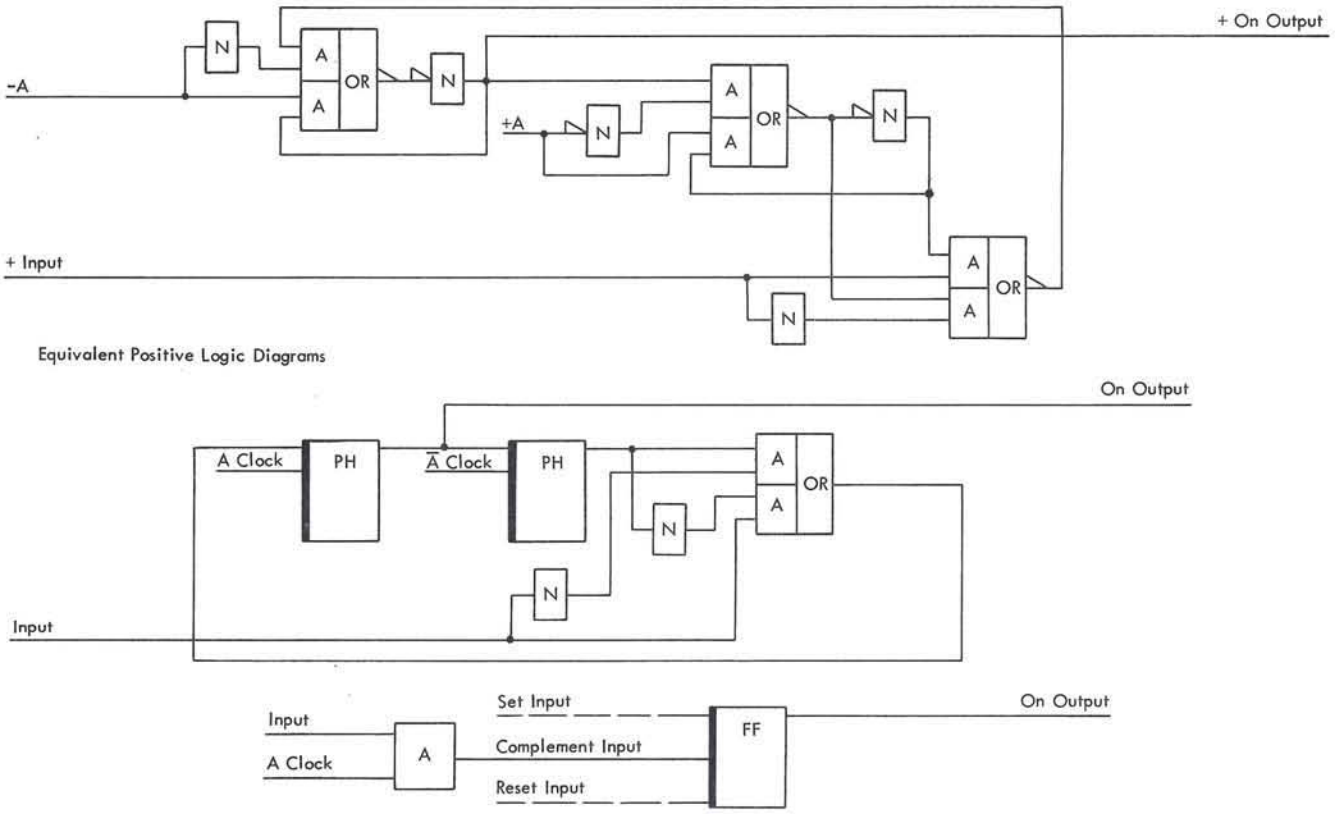


FIGURE 48. 2075 FLIP-FLOP

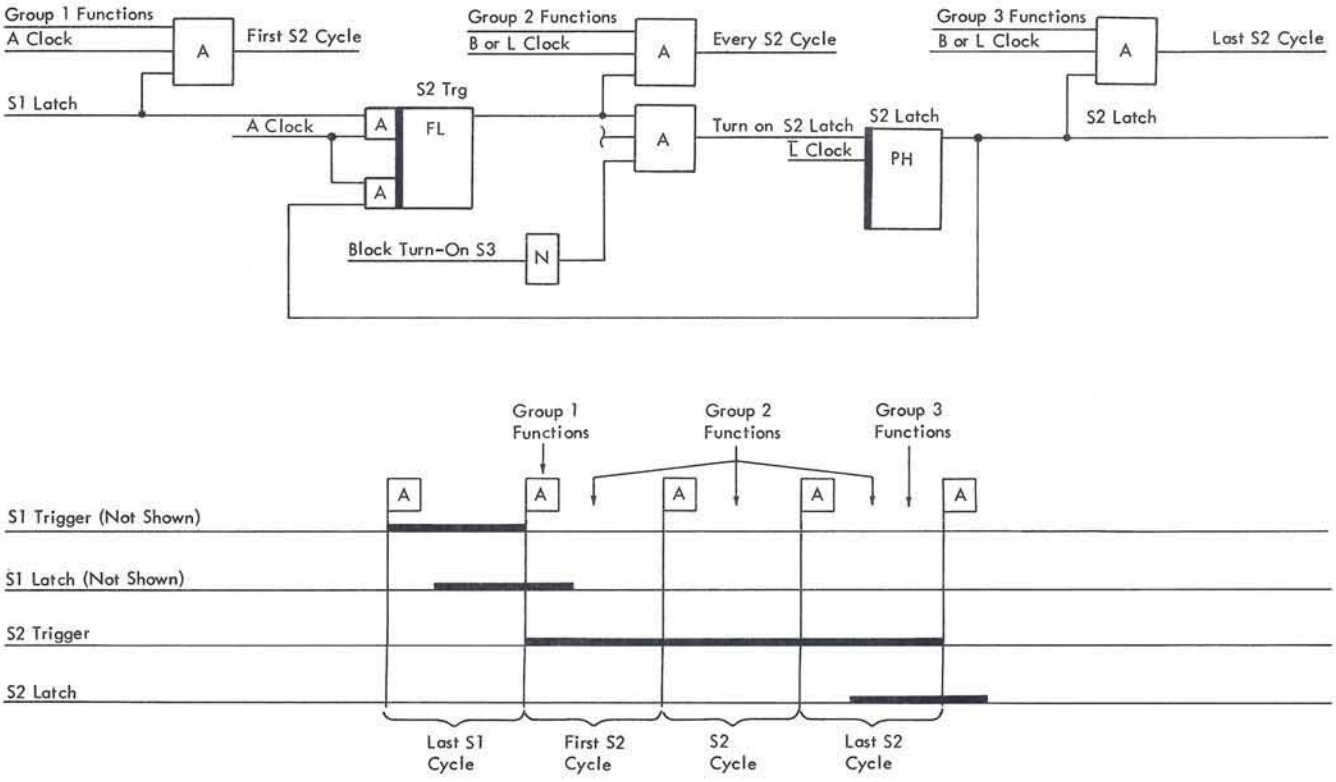


FIGURE 49. SEQUENCERS AND SEQUENCER CYCLES

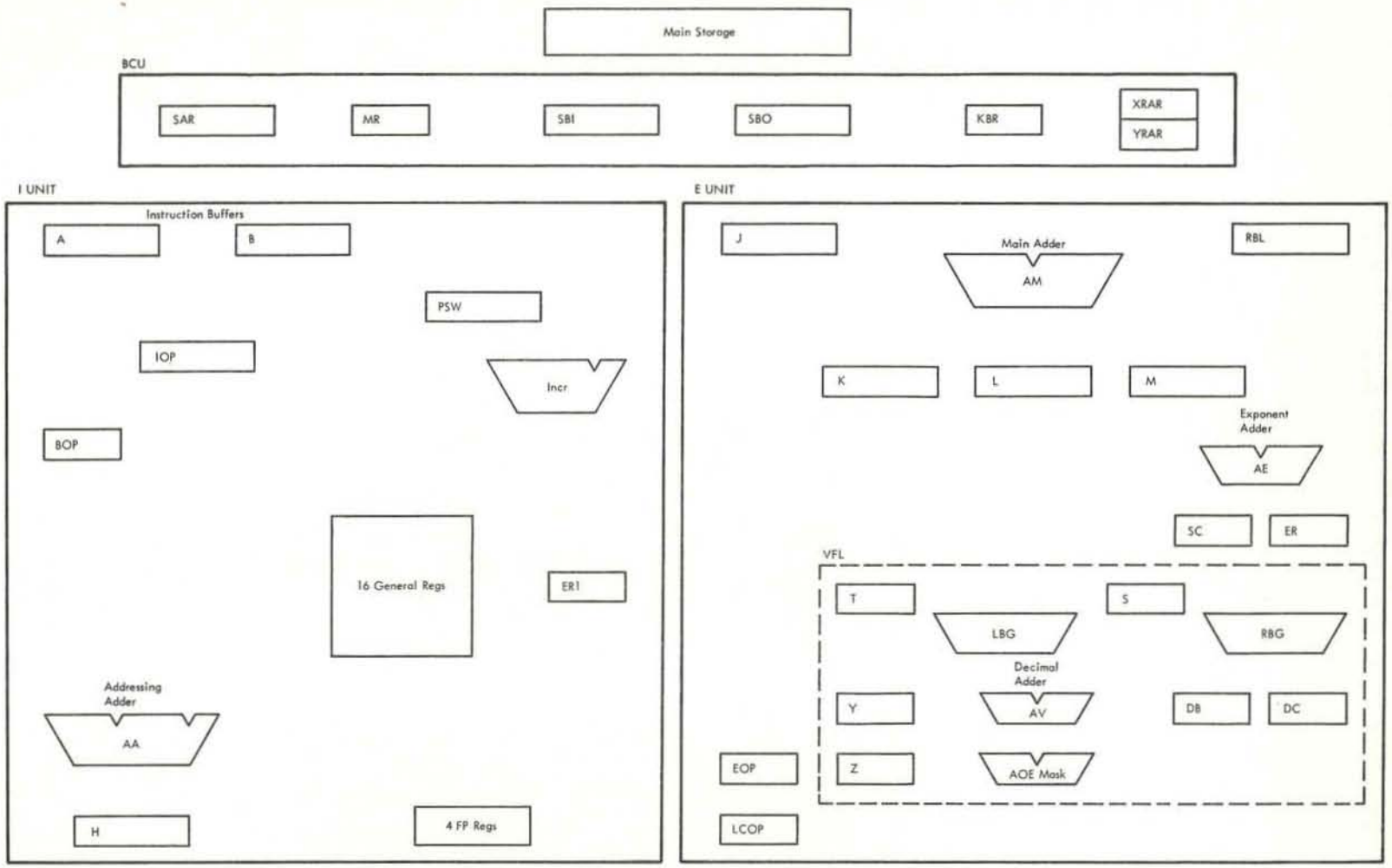


FIGURE 50. MAJOR CPU FUNCTIONAL UNITS

Instruction Fetch

- An instruction is 2, 4, or 6 bytes in length.
- Instructions are fetched a double word at a time.
- A and B registers buffer two double words of instructions.
- IC fetch fills an empty instruction buffer register.
- Incrementer generates the IC fetch address.
- Even-address storage words are set into A. Odd address storage words are set into B.
- IC fetch address is set into the storage address register (SAR).
- BCU returns fetched storage words under control of two return address registers.

An instruction is either two, four, or six bytes in length and can begin on any halfword boundary (address bit 23 is zero). In the Model 75, instructions are fetched one double word at a time and placed in one of the two instruction buffer registers (Figure 51).

Instruction fetching depends on the status of the A and B registers. Whenever all of the instructions contained in one of these registers have been processed, an IC fetch is made to replenish the empty register. The instruction count register (ICR), the last 24 bits of the PSW, keeps track of the instruction which is currently being processed. The ICR is incremented by 8 or 16 to generate the address of the next instruction double word. For example, assume that A is empty and that instructions are being processed out of B. An increment of 8 (address bit 20) is added to the ICR to fetch the next instruction double word. The true address generated for this fetch is a byte address eight bytes higher than the value of the ICR. However, because the three low-order address bits are ignored in addressing storage, the double word containing the specified byte address is always fetched.

In two situations, it is necessary to increment the ICR value by 16 to generate the correct IC fetch address. For example, assume that both A and B are loaded and instructions are being processed out of A. When the ICR is pointing to the last instruction in A, and this instruction has already been transferred to the op registers, an IC fetch can be initiated to reload A. In this case, it is necessary to add 16 to the ICR value to skip the double word already in B and reload A with the correct instruction word.

The second situation that necessitates incrementing the ICR by 16 occurs when one buffer is empty and the last instruction in the second is being processed. In this case, two instruction fetches are necessary to refill the buffers. The first IC fetch is made by adding 8 to the ICR; if the second fetch is made immediately following the first, 16 must be added to the ICR for the second fetch.

Notice from the figure that even-address instruction words (double words) always go into A and odd words go into B. The determination of even/odd is made from address bit 20, which changes at each multiple of eight byte addresses and, therefore, once each storage word (double word).

An IC fetch address is set into the storage address register (SAR), as is any address generated by the CPU for a fetch or store to main storage. The three low-order bits of SAR, which specify the byte within a storage word, are not transferred from SAR to main storage.

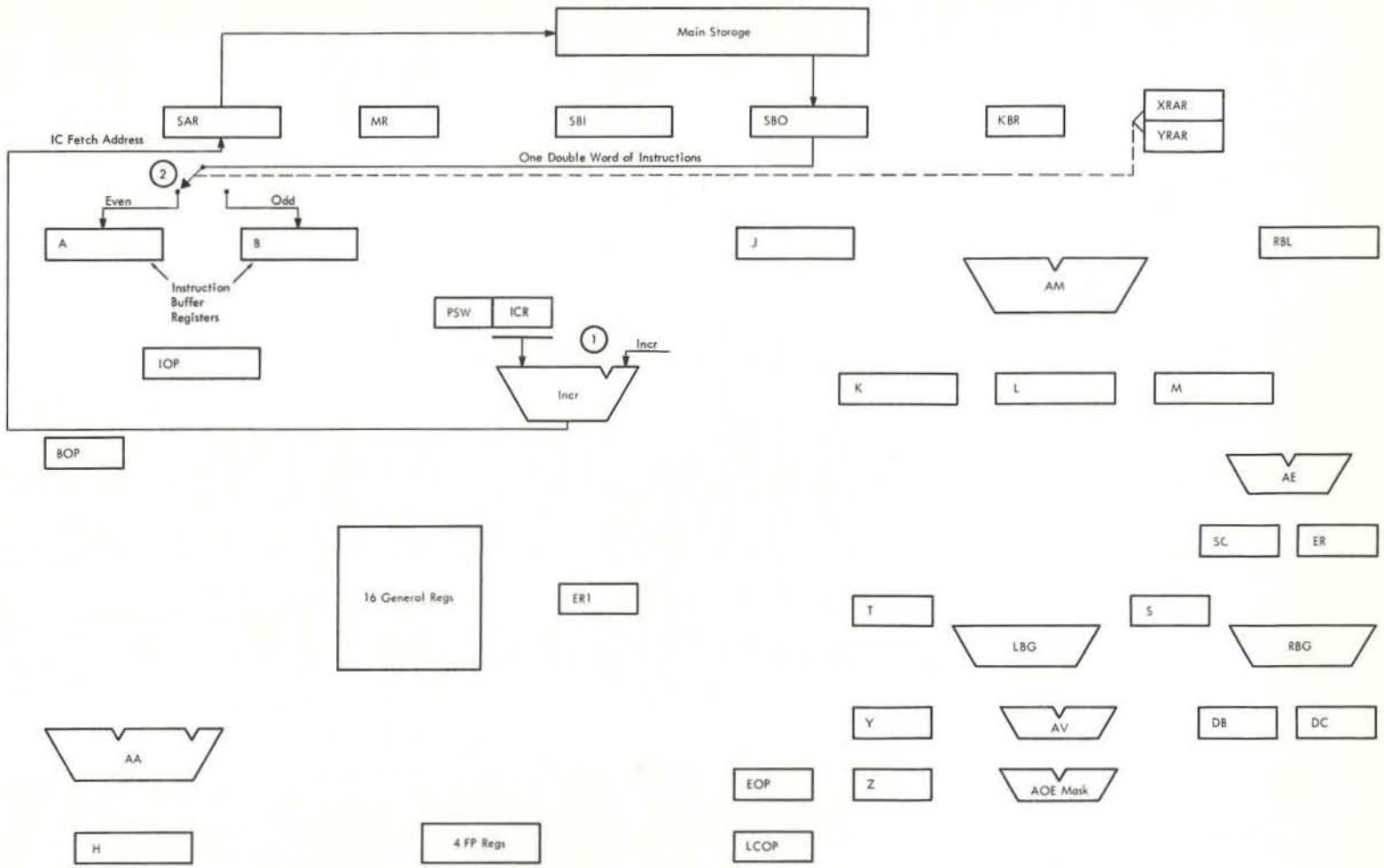
Storage returns fetched instruction words (and all other fetched words) to the storage bus out (SBO) latch register. From the SBO, the BCU routes the fetched data to its proper destination by using either the X or Y return address register (RAR). Two RAR's are necessary because of overlapped storage cycles (two fetches may be outstanding). The X and Y RAR's are used alternately to direct data and errors for each storage request (fetch or store).

Op Register Loading

- The CPU has seven op registers: IOP, BOP, EOP, LCOP, ER 1, Y, and Z.
- Instructions are gated from AB to IOP by the gate select register (GSR).
- BOP, EOP, Y, and Z are set from IOP.
- IOP is used for instruction preparation.
- BOP is used by the IE execution unit.
- EOP and LCOP are used by the E unit.
- ER 1 is used for local storage put-aways.
- Y and Z hold operand lengths for VFL operations.

The CPU has seven op registers (Figure 52):

1. I unit op register (IOP)
2. B op register (BOP)
3. E unit op register (EOP)
4. E last cycle op register (LCOP)
5. E time R1 field (ER 1)



(1) An increment of 8 or 16 is added to the ICR to fetch the next double word of instructions. The increment added depends on the remaining number of unused bytes in A-B.

(2) The X or Y return address register (RAR) routes the fetched double word into A or B depending on whether the address specified an even or an odd storage word. The two RAR's are used alternately on storage operations because of overlapped storage cycles.

FIGURE 51. IC FETCH

6. Y counter
7. Z counter

The gate select register (GSR) gates instructions from the instruction buffers into IOP. The GSR always gates two halfwords (four bytes) and these two halfwords are on half-word boundaries. The A-B registers are treated as a continuous ring such that the GSR may simultaneously gate the last halfword of A and the first halfword of B, or the last halfword of B and the first halfword of A. For single halfword instructions, the second half of IOP is ignored. For triple halfword instructions, the gating is switched from the second halfword to the third halfword during E-time whenever the information in the third halfword is needed.

The BOP and EOP registers are set from IOP. The BOP register is used during the E time of instructions executed by the IE unit. The EOP register is used during the E time of other execution unit instructions. The BOP and EOP registers are necessary because of the overlap of I and E time. Generally, IOP contains the next instruction to be executed and is used by I unit for instruction preparation functions such as operand fetches and register deliveries. The BOP and EOP registers hold the instruction currently being executed and provide the decoded operation line such as "RR add," during the execution of the instruction.

The LCOP register is set from EOP. Its purpose is exactly the same as EOP; however, EOP and LCOP are used at different times to prevent lost E time between two instructions. During the first E cycle, the LCOP is set from EOP. LCOP remain good until the first E cycle of the next instruction. Its main use is during ELC at which time EOP is already set for the next instruction to allow the new instruction to be completely decoded from EOP before its E time actually begins.

The ER1 op register holds the R1 field of an instruction to select a local storage put-away register.

The Y counter holds the L1 (length of operand 1) field for storage-to-storage operations.

The Z counter holds the L2 field for storage-to-storage operations. Both Y and Z can be decremented; these registers are used to determine when all bytes of both operands have been processed, and thus end the operation.

ICR Updating (Figure 53)

- ICR contains the address of the next instruction to be executed.
- ICR is updated by first updating the GSR.
- Updating the GSR is accomplished by adding the length of the current instruction to the GSR.
- The updated GSR is set into the ICR.
- If no carry is generated when the GSR is updated, the ICR updating is complete.
- If a GSR carry occurred, the ICR passes through the incrementer, where a 1 bit is added into position 19.

The ICR contains the address of the next instruction to the executed. Three bits, bits 20-22, of the ICR are duplicated in the GSR to gate the next instruction from the A-B registers to the IOP register. These bits point to one of eight halfwords in A-B. The halfword following the one designated by the GSR is also gated to IOP.

ICR updating is accomplished by first updating the GSR (Figure 53). The length of the current instruction in IOP (first two bits of the op code) is added to ICR bits 20-22. The result forms the new GSR contents. If a carry did not result from the addition, the ICR updating is completed by simply transferring the new contents of the GSR to bits 20-22 of the ICR. However, when GSR goes from pointing to the right end of B to pointing to the left end of A, a carry will result from the addition. In this case, GSR is again transferred to the ICR but a one must be added into ICR bit 19 to continue the updating. A bit 19 is added to the ICR by passing the ICR through the incrementer to complete the ICR updating.

RX Operand Fetch

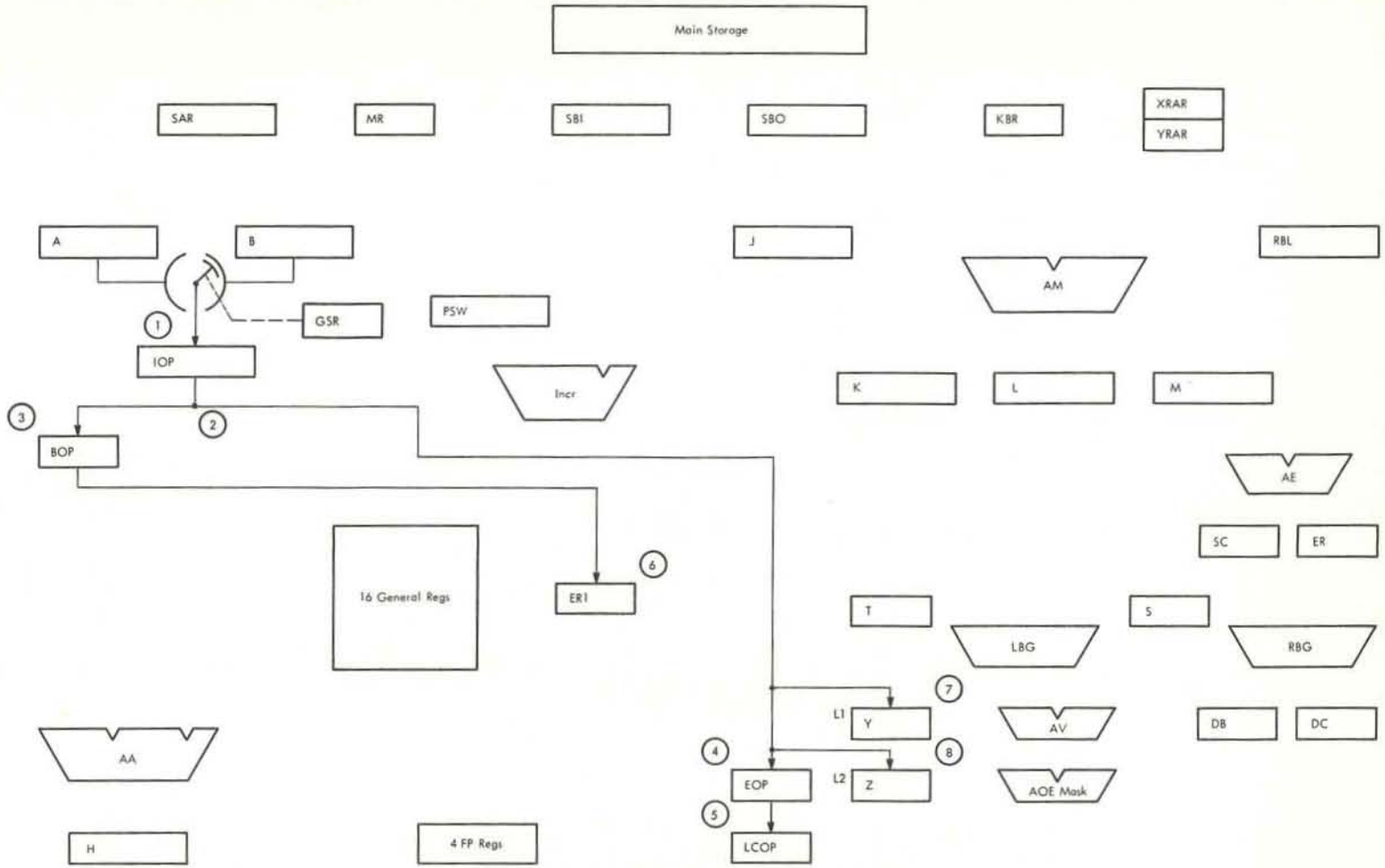
- RX format instructions use one operand from main storage and one operand from local storage.
- I unit fetches the operand.
- Operand is delivered to the J register.

On RX format instructions, one operand comes from main storage and the second operand comes from local storage. The I unit makes the fetch from main storage and instructs the BCU to deliver the fetched double word to the J register (Figure 54).

The address is calculated in the addressing adder by the addition of the contents of two general registers and the D2 field of the instruction. The general registers are selected by the B2 and X2 fields of the instruction decoded from IOP. The D2 field is taken directly from IOP.

Register Operand Deliveries

- I unit usually delivers the register(s) from local storage to RBL.



(1) An instruction to be executed is gated from the instruction buffers to IOP by the gate select register (GSR). IOP is used by I Unit (T1 and T2 functions).

(2) Six other Op regs are set from IOP. The portion of instructions held by each Op reg depends on the purpose of that resister.

(3) BOP holds the Op code and R1 field. BOP is used for instructions executed by the I-E execution unit and for some T2 functions.

(4) EOP holds the Op code only. EOP is used for E execution unit instructions until, but not including, ELC.

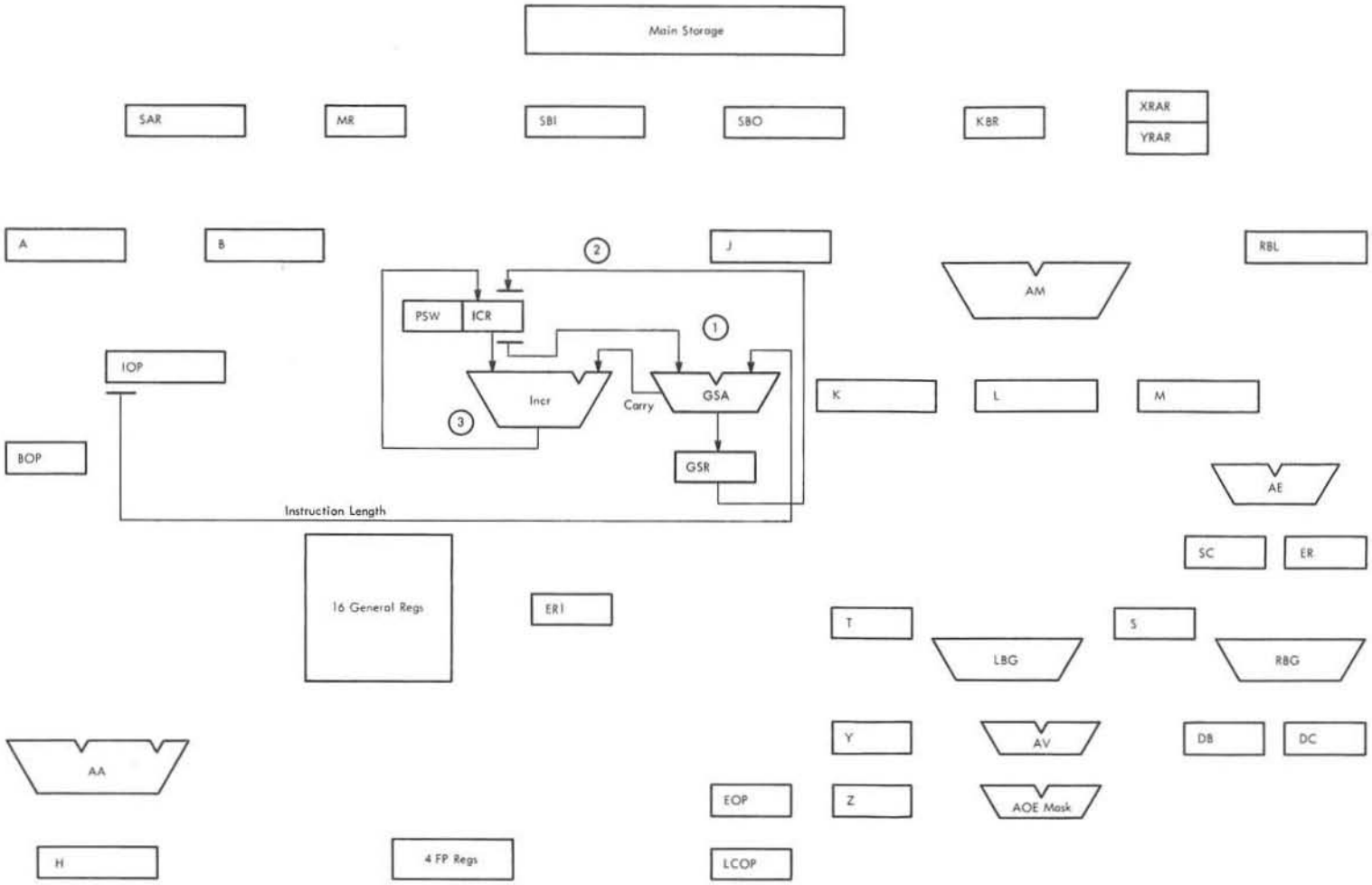
(5) LCOP holds the Op code only. LCOP is used during E execution including ELC to allow EOP to be set for the next instruction.

(6) ER1 holds the R1 field to select a put-away general or FP register.

(7) Y holds L1, the length of operand 1 on SS instructions.

(8) Z holds L2, the length of operand 2 on SS instructions.

FIGURE 52. OP REGISTER LOADING

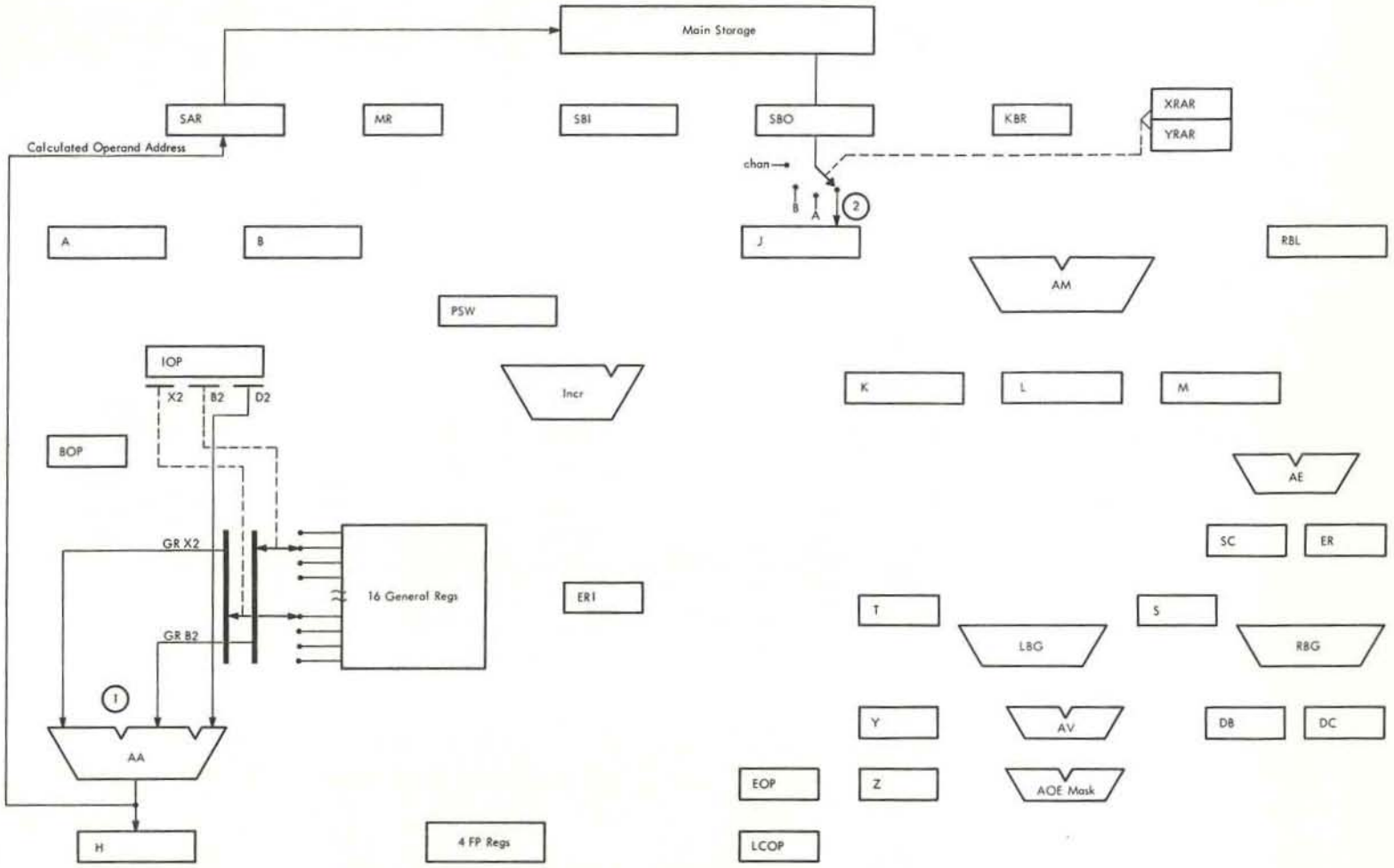


(1) ICR bits 20-22 are updated by the present instruction length to set the gate select reg (GSR).

(2) New GSR contents replace 20-22 of ICR.

(3) If a carry occurred from the gate select adder (GSA) when adding the instr length, the ICR is passed through the Incrementer and a 1 is added into position 19.

FIGURE 53. ICR UPDATING



(1) The Addressing Adder (AA) adds the D2 field of the instruction, the contents of B2, and the contents of X2 to calculate the effective address. The calculated address is set into SAR and H.

(2) The BCU delivers the contents of the calculated address to J.

FIGURE 54. RX OPERAND FETCH

- Some instructions use a pair of general registers as one operand.

- A control trigger, GROUT, is used whenever general registers must be gated out during E time.

- The FLOUT control trigger is used to gate floating-point registers out during E time.

Most instructions use at least one operand located in local storage (a general or a floating point register). The I unit usually delivers the register or registers that will be needed by the E unit to the register bus latch (RBL) register (Figure 55). As an example, consider a fixed point RR instruction (except multiply). In this instruction, the first operand is in the general register specified by the R1 field of the instruction. The second operand is in the general register specified by the R2 field of the instruction. The two general registers are selected simultaneously by decoding the R1 and R2 fields off of IOP. The selected registers are gated out on general bus left (GBL) and general bus right (GBR) to the RBL register.

Certain fixed-point instructions use a pair of general registers to hold a single operand. For example, on an RR multiply, the product is returned to a pair of general registers; on RS double shift instructions, two general registers are coupled; on an RX divide instruction, two general registers hold the dividend. For these instructions, the pair of general registers which hold the operand are designated by the R1 field and R1 + 1. These instructions must specify an even R1 general register such as 0, 2, 4, 6, --- 14. The second general register of the pair is, therefore, an odd register (R1 + 1). If R1 + 1 is not odd, it is not delivered to the RBL and a specification error is signaled to cause a program interrupt. On those instructions which use a register pair, the R1 + 1 general register (if odd) is substituted for the R2 general register and delivered to the RBL.

The general register deliveries discussed to this point are initial register deliveries made by the I unit as a service for the E unit. Some instructions require additional register deliveries. These additional deliveries are handled by the E unit and are signaled by the "general register out" (GROUT) control trigger. An example use of GROUT is on RR divide to deliver R1 + 1, which contains the second half of the dividend. Another example is on RR multiply to get R2, which is the multiplier.

Floating-point registers are handled similar to a pair of general registers because floating-point registers contain a double word. On FP RX instructions, where only one FP register is used, the I unit delivers FP R1 to the RBL. On FP RR instructions, the I unit delivers FP R2. The "floating-point registers out" (FLOUT) control trigger is turned on for FP RR instructions to gate FP R1 out during E time.

Result Storing

- Most instructions return a result to a local storage register.

- Store op codes transfer an operand from local storage to main storage.

- The K register is the origin for all data stored in either local or main storage.

Register Put-Away

- Returning a result to a local storage register is called register put-away.

- Data are taken from K for a register put-away.

- ER1 selects the put-away register.

Most instructions (other than VFL) return a result to a local storage register (Figure 56). The result is placed in the K register (with the exception of a FP exponent) and the ER1 op register selects the general or floating point register for the put-away.

Operand Store

- Data path is from K to the SBI to main storage.

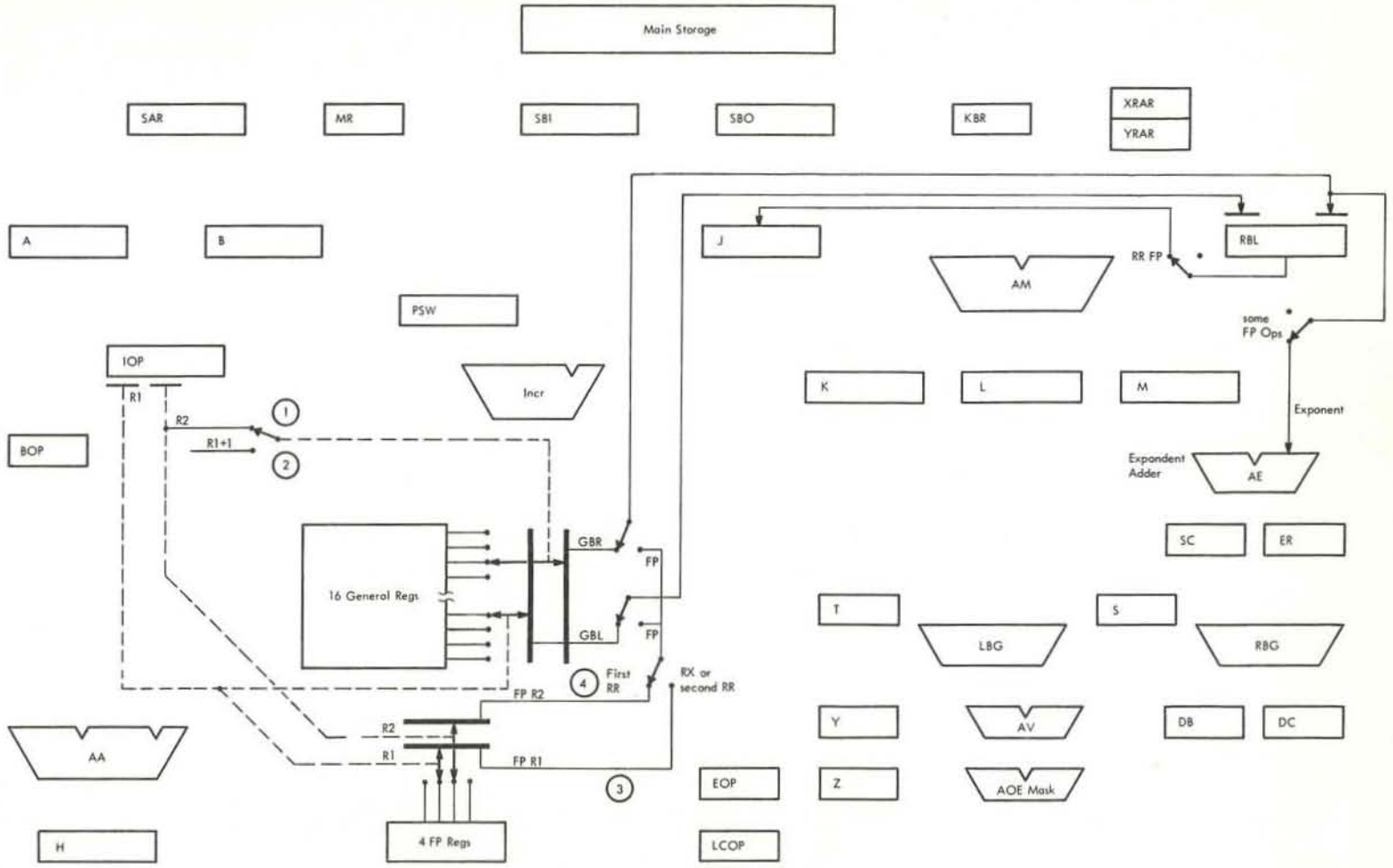
- Address is calculated by AA and set into SAR.

- A mark-register bit must be set for each byte of SBI to be stored.

- Bytes not having a corresponding mark-bit are regenerated in storage.

A store-to-main-storage is accomplished by putting the store data in the K register and setting the storage address into SAR (Figure 57). In addition, the mark register must be set to tell storage which bytes in the storage word are to be changed. The mark register contains eight bits, one bit for each of the eight bytes in a storage word. A mark-bit must be set for each byte to be stored. Bytes having a corresponding mark-bit of zero are regenerated in storage without change.

For an operand store, the storage address is calculated by the I unit exactly as for an operand fetch. The B2 and X2 fields are decoded from IOP and the D2 field is taken directly from IOP.



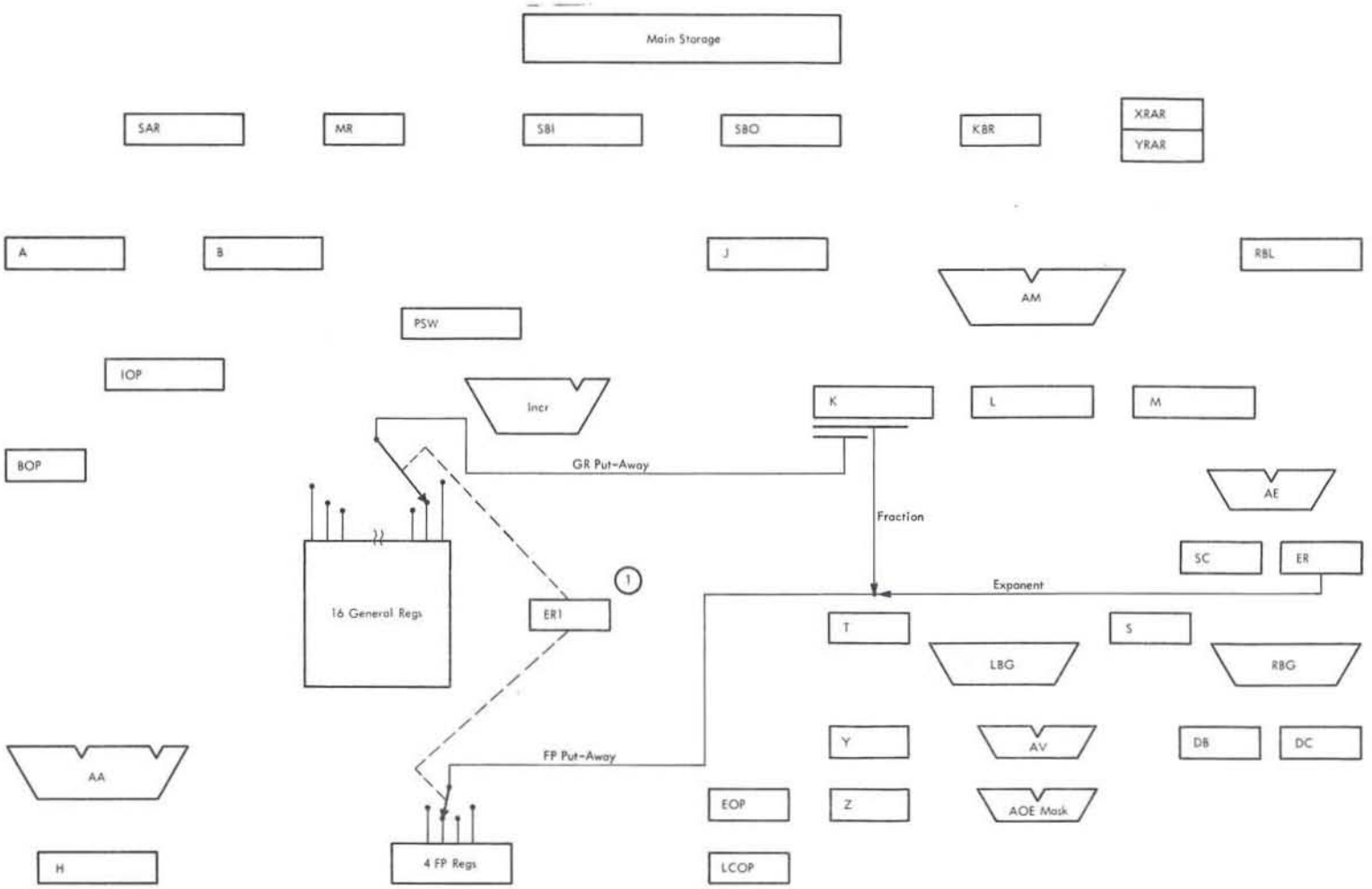
(1) For RR Fixed-point instructions (expect multiply), GR R2 is delivered to the right half of RBL and GR R1 is delivered to the left half.

(2) For RX Fixed-point instructions, RR multiply, and RS shift instructions, GR R1+1, if odd, is substituted for GR R2. On RR multiply, GR R2, the multiplier, is delivered later.

(3) For RX Floating-point instructions, FP R1 is set into the RBL.

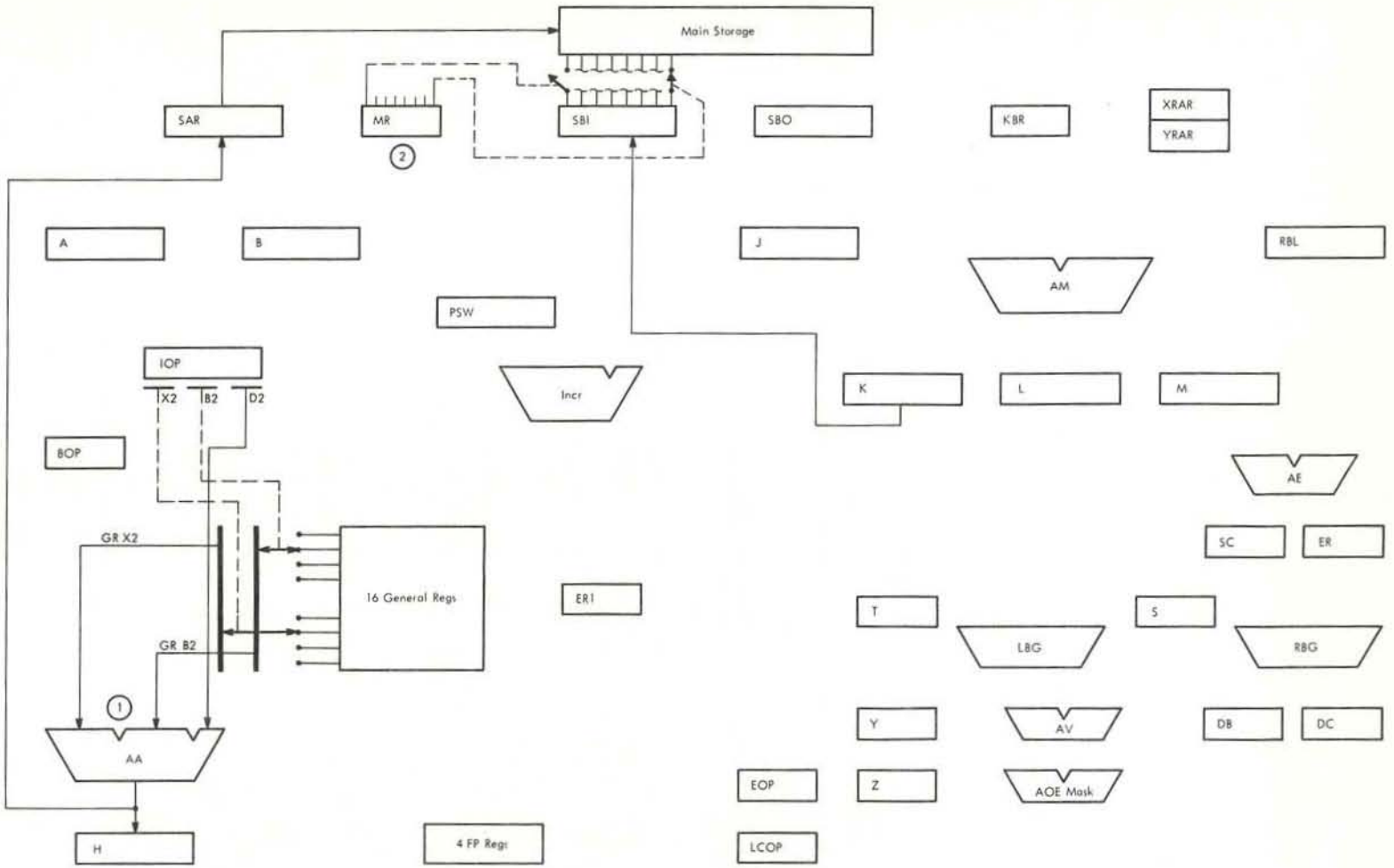
(4) For RR Floating-point instructions, FP R2 is first delivered to RBL. Later, after FP R2 has been transfered from RBL to J, FP R1 is delivered to RBL.

FIGURE 55. REGISTER OPERAND DELIVERY



(1) ER1 contains the R1 field of the instruction. For double GR put-aways, the left half of K is transferred to GR R1, then ER1 is incremented and the operation is repeated.

FIGURE 56. REGISTER PUT-AWAYS



(1) AA calculates effective address as an operand fetches.

(2) Mark register (MR) is set with bits corresponding to bytes of K to be stored. Mark bits set depend on Op code: Store sets 4 adjacent mark bits; Store half sets 2 adjacent mark bits; Floating-point store double sets all mark bits. Main storage regenerates all bytes not masked by mark bits.

FIGURE 57. OPERAND STORE

Instruction Execution Examples

CPU handles three kinds of arithmetic:

1. Fixed-point binary.
2. Floating-point binary.
3. Variable field length decimal.

Fixed-Point Add/Subtract

- At beginning of E time for RR instructions, R1 and R2 are in RBL.
- At beginning of E time for RX instructions, R1 is in RBL and operand 2 is in J.
- Operands are passed through AM and set into K for put-away into R1.
- All add op codes are added true.
- All subtract op codes are complement added.
- All negative fixed-point operands are kept in two's complement form in both main and local storage.

At the beginning of execution-time for an RR fixed-point add/subtract, the operands are both in RBL (Figure 55). At the beginning of execution time for an RX fixed-point add/subtract, R1 is in RBL (Figure 55) and operand 2 has been fetched from storage and set into J (Figure 54). In either event, the two operands must be combined (added or subtracted) and delivered to K for put-away into R1 (Figure 56).

For an RR add/subtract, the two operands are moved from RBL to M, and from M into the main adder (Figure 58). The result from the adder is delivered to K for the general register put-away to R1.

For an RX add/subtract, operand 1 (R1) is moved from RBL to M and then to the main adder just as on RR instructions; however, operand 2 (from storage) is taken from J to the main adder. The output on the main adder is set into K for the R1 register put-away just as on RR instructions.

Unlike many other computers, the decision of whether or not to complement one of the operands during the addition depends only on the op code; the second operand is complemented on all subtract op codes and it is added true on all add op codes. The operand signs need not be checked to determine whether or not to complement because all negative numbers are kept at all times, in local or main storage, in two's complement form. This means that a true algebraic addition or subtraction results from the corresponding op code.

Floating-Point Add/Subtract

- Floating-point numbers consist of a fraction and an exponent.
- Exponent represents a power of 16 by which the fraction is multiplied.
- Exponent is one byte; fraction is either three or seven bytes.
- Floating-point arithmetic is used for operands too large or too small to be handled by fixed-point.
- Operand exponents must be equalized before addition or subtraction.
- Exponent difference is calculated by the exponent adder.
- If exponents are unequal, one of the operand fraction is shifted until the exponents are equal.
- After factors are aligned, the fractions are passed through AM to perform the addition or subtraction.
- The result fraction is set into K for local storage put-away.
- Exponent of the result is the original larger exponent.

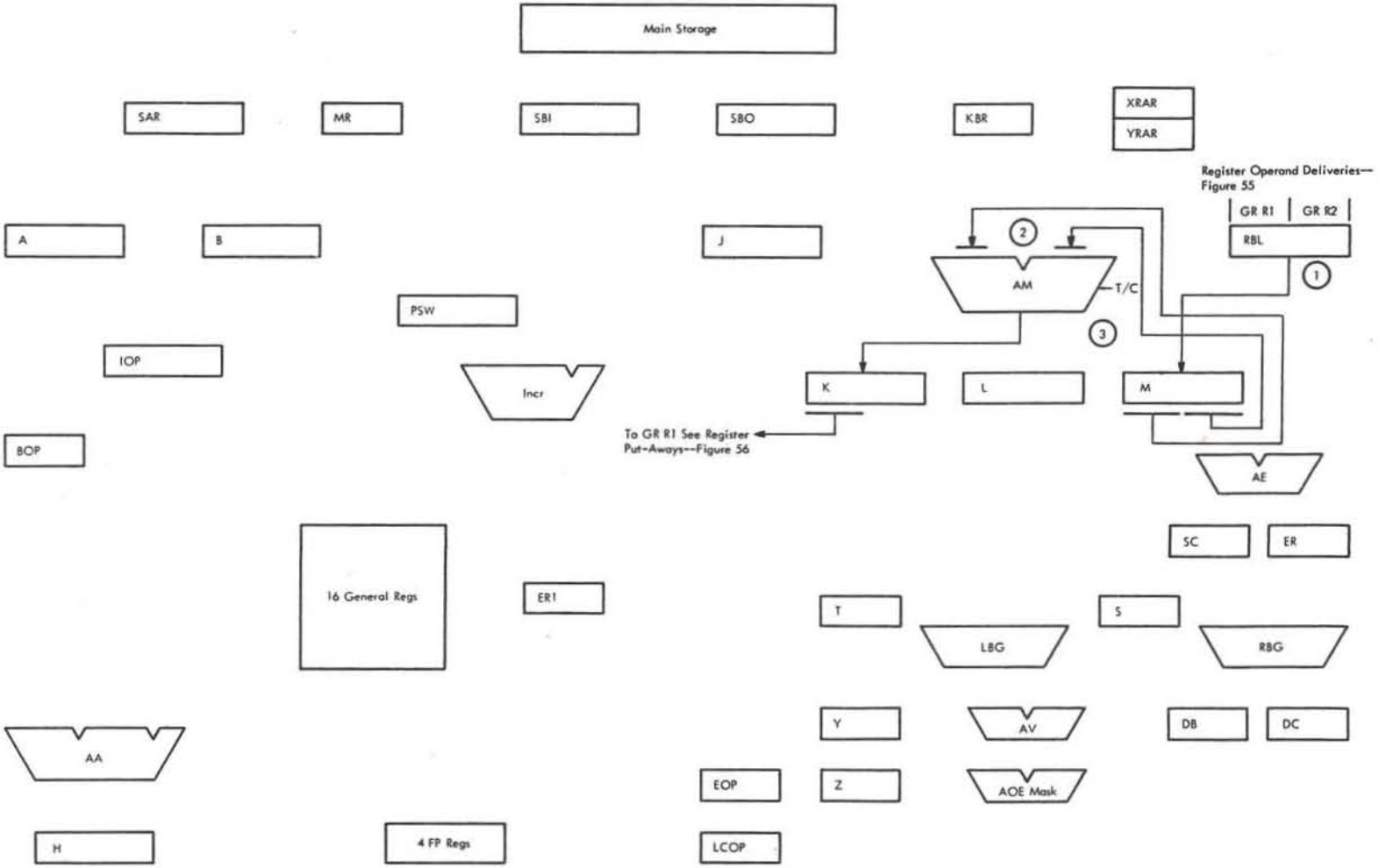
Floating-point numbers have two parts, a fraction and an exponent. The exponent is one byte; the fraction may be either three bytes (short operands) or seven bytes (long operands).

Following is a simplified explanation of floating-point operations.

The fraction of a floating-point operand represents a series of hexadecimal digits, just as does a fixed-point operand. The exponent of a floating-point operand represents a power of 16 by which the fraction must be multiplied in order to obtain the true value of the fraction digits. In practice, floating-point arithmetic is used to handle numbers which are too large or too small to be represented in fixed-point. The exponent automatically keeps track of the decimal (actually hexadecimal) point of these numbers.

To add or subtract two floating-point operands, the exponents must first be made equal by shifting one of the fractions. This is equivalent to aligning the decimal point when adding:

1.23	1.23	1.23
+ 12.3	+ 1 2.3	+ 12.3
can't add	can't add	13.53



(1) The contents of GR R1 (Operand 1) and GR R2 (Operand 2) are delivered to RBL. RBL is transferred to M.

(2) The operands are passed from M through the main adder (AM) and the result is set into K. From K, the result is put-away into GR R1.

(3) Operand 2 is complement-added (subtracted) on all subtract Op codes and true-added on all add Op codes.

FIGURE 58. RR FIXED-POINT ADD/SUBTRACT

The fraction which has has the smaller exponent is shifted to make its exponent equal to the other. The exponent of the result is the original larger exponent.

At the beginning of a floating-point add/subtract E time, operand 1 is in RBL and operand 2 is in J (Figure 59). For RR instructions, both operands came from local storage (Figure 55). For RX instructions, operand 1 came from local storage and operand 2 came from main storage (Figure 54).

Operand 2 is routed from the J register through the main adder to K (Figure 59). In the transfer, the exponent is removed so that only the operand 2 fraction is in K. Operand 1 fraction is moved from RBL to M. A parity bit is forced for bits 56-63. This allows M-register bits 56-59 to remain available for a possible guard digit.

The two operand exponents are routed through the exponent adder (AE) where one is subtracted from the other to determine the exponent difference. The difference is set into both the exponent register (ER) and the shift counter (SC). The result of the subtraction determines which operand must be shifted and the SC contains the number of shifts required. If the exponents are equal, no shifting is required. If shifting is required, the operand to be shifted is passed through AM one or more times under control of the SC to accomplish the required number of shifts.

The aligned fractions in K and M can now be added or subtracted (Figure 60). Both operand fractions are passed through AM to perform the addition or subtraction. The results are set into both K and M.

At this point, the result fraction is in K ready for put-away. The result exponent, however, is the larger of the two original operand exponents. The original operand 1 exponent is in the FP register specified by BR1. If this is the one required for the result, it is passed through the AE and set into ER. ER and K are then put-away.

If operand 2 originally had the larger exponent, it must be reconstructed because the operand 2 exponent has been lost. It is reconstructed by combining the operand 1 exponent with the exponent difference still in the ER. The result is equal to the original operand 2 exponent and is set into the ER for put-away.

VFL Add/Subtract

- VFL handles operations whose operands are a variable number of bytes.
- The two types of VFL operations are: logical and decimal arithmetic.
- Logical instructions use zoned format data.
- Decimal arithmetic uses packed format.
- Instructions are SS format.
- Decimal instructions generally require four execution sequences:
  a. Set-up : initial operand fetches.
  b. Iterations : perform the required operation, one byte at a time.
  c. Pre-fetch : fetches new operand 2 storage words prior to the time they will be needed.
  d. Store-fetch : stores finished result bytes and fetches new operand 1 words.
- On add/subtract, T and S pointers select operand bytes from operand storage words.
- Y and Z count the number of bytes processed to determine when to end the operation.

The variable field length (VFL) section of the CPU handles operations which require manipulation of a variable number of bytes. Processing is accomplished a byte at a time until all bytes have been processed. There are two types of VFL instructions : decimal arithmetic and logical instructions.

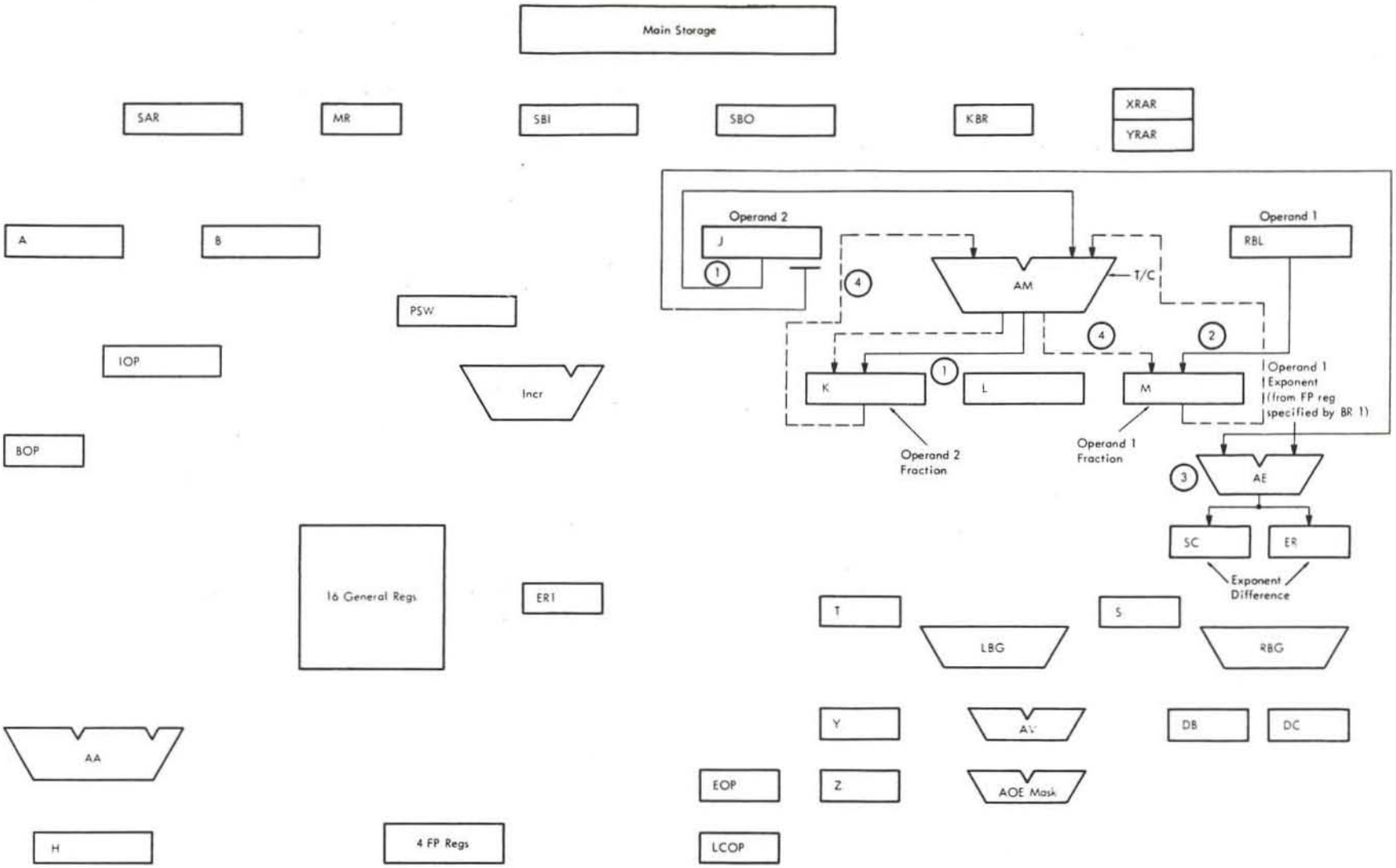
VFL logical operations are those intended primarily for manipulation of alphameric characters (zoned format). Examples are the edit and translate instructions.

Decimal arithmetic operates on data in the packed format. In this format, two decimal digits are placed in one eight-bit byte. Decimal arithmetic operations include add, subtract, multiply, and divide.

Decimal instructions are in the storage-to-storage (SS) instruction format. Both operands are taken from storage, the designated operation performed, and the result is returned to the storage location which originally held operand 1. An instruction generally requires four sequences.

A set up sequence brings the first word of each operand from storage and sets the necessary registers to control byte selection within each operand word and to determine when all requested bytes have been processed. An iteration sequence does the operation called for such as add or subtract. The operation is performed over and over on successive bytes until all bytes have been processed or until iterations must be suspended to store a double word of results and to fetch a new operand word.

A store-fetch sequence is used to put result-words back into the operand 1 storage location and to fetch the next operand 1 double word. Iterations must be suspended during this sequence. When all



(1) Operand 2 is routed from J to K. The exponent is removed in the transfer.

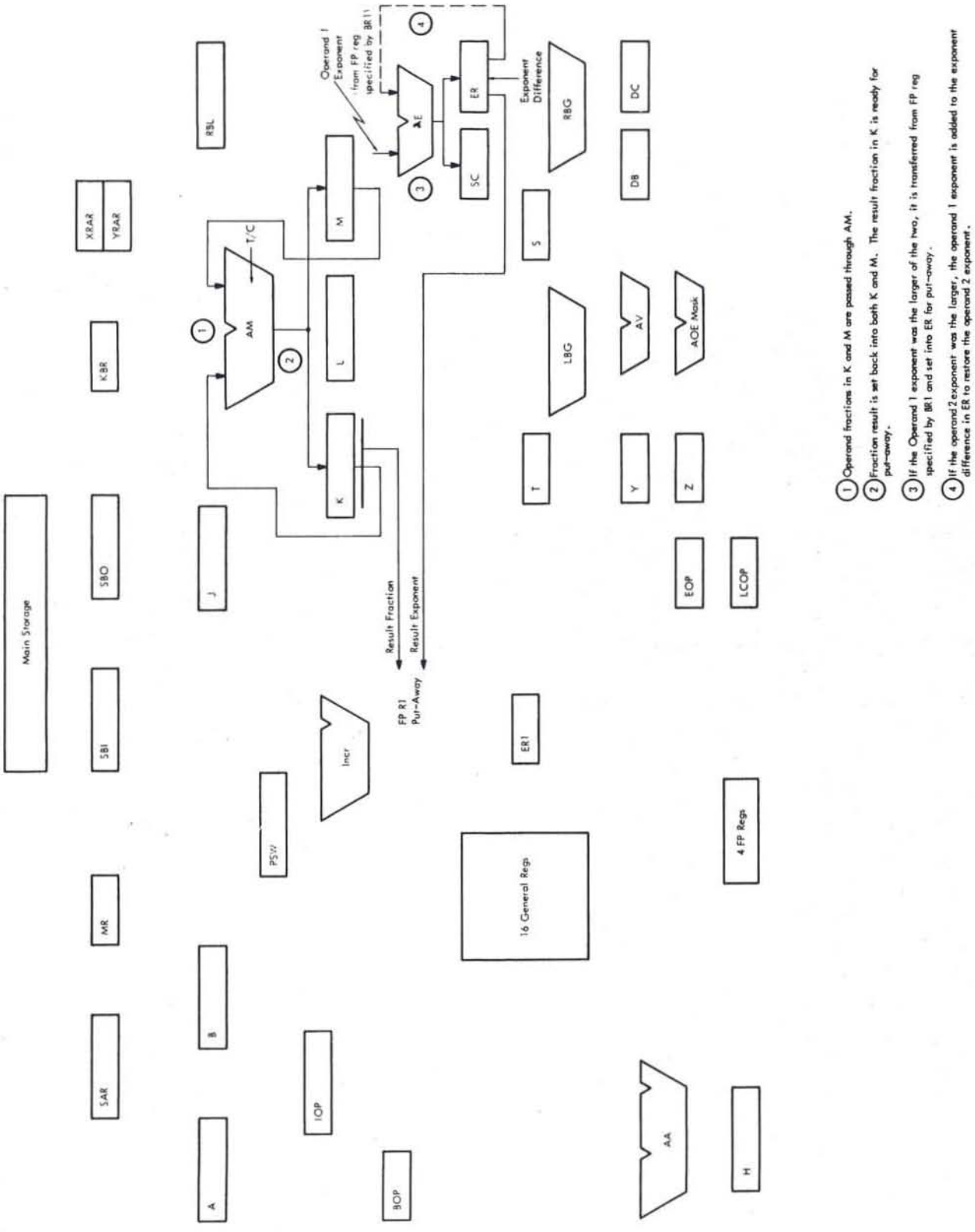
(2) Operand 1 is moved from RBL to M.

(3) Determine exponent difference.

(4) If exponents are not equal, shift either K or M by passes through AM until exponents are equal.

●FIGURE 59. BASIC FLOATING-POINT EXPONENT EQUALIZATION

Form 223-2872-1
FES S26-7033



(1) Operand fractions in K and M are passed through AM.

(2) Fraction result is set back into both K and M. The result fraction in K is ready for put-away.

(3) If the Operand 1 exponent was the larger of the two, it is transferred from FP reg specified by BR1 and set into ER for put-away.

(4) If the operand 2 exponent was the larger, the operand 1 exponent is added to the exponent difference in ER to restore the operand 2 exponent.

●FIGURE 60. BASIC FLOATING-POINT ADD/SUBTRACT

bytes have been processed, the last result word is stored without fetching another operand.

A pre-fetch sequence is used to replenish operand 2 double words. Unlike the store-fetch sequence, this sequence is handled by looking ahead to determine that another operand 2 double word will be required. Successive operand 2 double words are fetched before they are required and temporarily stored in a machine register not used by the iteration sequence. The pre-fetch sequence can run concurrently with the iteration or set-up sequence. When operand 2 must be replenished, iterations are interrupted only one machine cycle to move the pre-fetched operand 2 word into the proper register.

The following discussion is a simplified explanation of a typical VFL add/subtract operation. Factors such as overlapping operand fields will vary the operation described.

The basic VFL add-subtract set-up sequence fetches the first operand 1 double word from storage (Figure 61). The fetch address is calculated in the AA by adding B1, D1, and L1. L1 was set into Y from IOP during I time. When this word returns, it is set into K. The second operand address is calculated by adding B2, D2, and L2. L2 was set into Z during I time. When this word returns, it is set into L.

During the initial fetches, the calculated addresses are set into H. H 21-23 is set into T (T pointer) during the first fetch to indicate the starting byte of operand 1. H 21-23 is set into S (S pointer) during the second fetch to indicate the starting byte address of the second operand. Actually, the second fetch may be made before the first operand 1 word returns from storage because of overlapped storage operations.

With the first word of operand 1 in K and the first word of operand 2 in L, the iteration sequence can begin (Figure 62). The T pointer controls the left byte gate (LBG) to gate the first operand 1 byte into the decimal adder (AV). The S pointer controls the right byte gate to put the first byte of operand 2 into AV. The AV output is guided back into the starting byte position of K by the T pointer.

After the first add/subtract cycle, the T and S pointers are decremented to gate the next operand bytes. The bytes gated on the second cycle will be the two to the left (lower register positions, but higher-order digits) of the starting operand bytes. The T pointer is also used to set a mark-bit so that the result byte just generated will be stored at the end of the operation or when a store-fetch sequence is initiated. At the end of the first add/subtract iteration, the Y and Z counter are also decremented to indicate that one byte of each operand has been processed. The iteration cycles continue until both Y and Z counters show that all bytes of both operands have been processed.

PURPOSE OF CPU FUNCTIONAL UNITS

This section contains a brief explanation of the major CPU functional units. The functional units are divided into four sections; BCU, I unit, E unit and VFL.

Purpose of BCU Functional Units

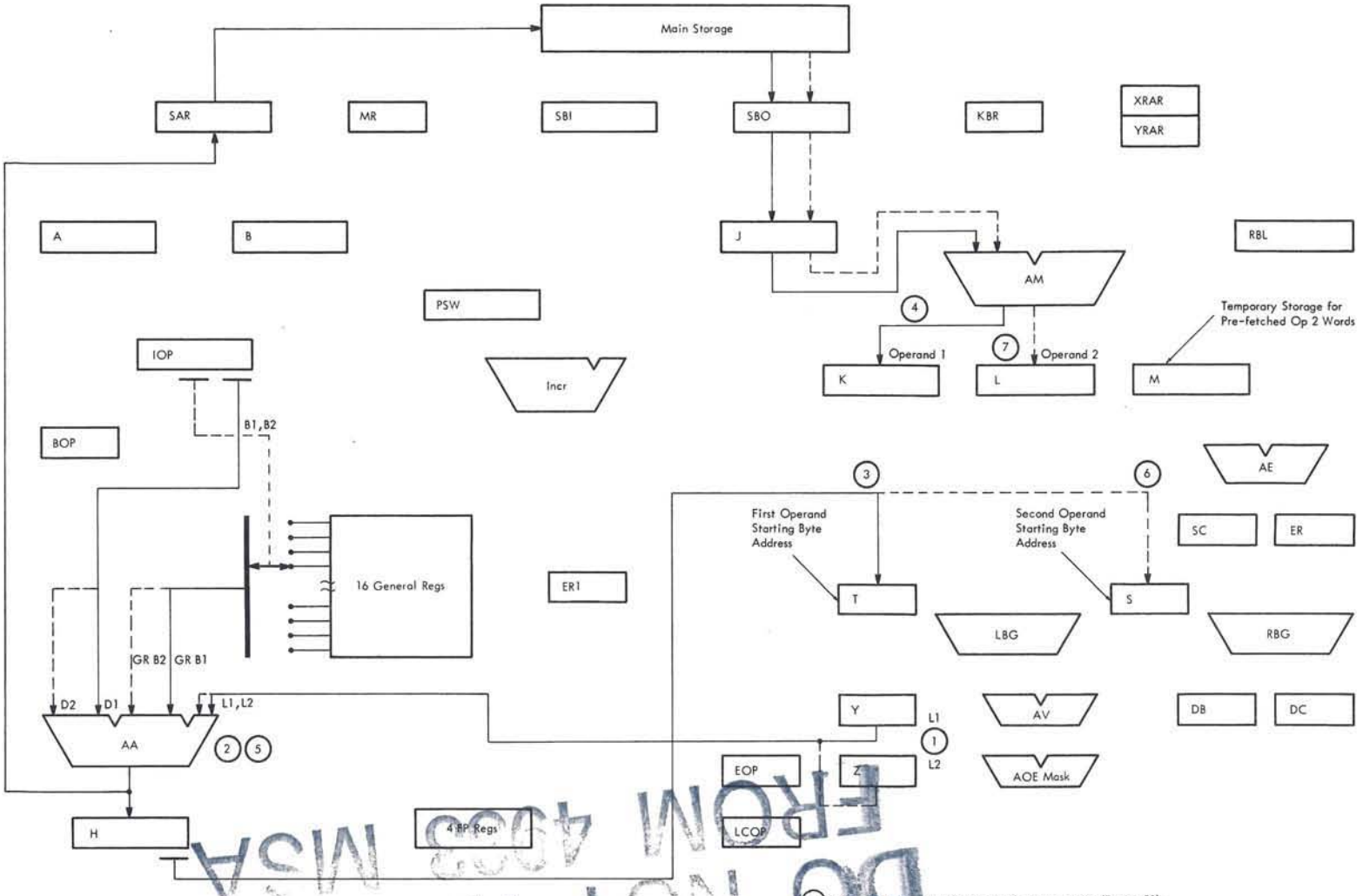
- Key buffer register.
- Mark register.
- Return address registers.
- Storage address register (SAR).
- Storage bus in (SBI) latch register.
- Storage bus out (SBO) latch register.

Key Buffer Register: This five data-bit plus parity-bit register holds the storage protect key from SP storage on an insert key instruction. The fetched key is transferred from the key buffer register to the AOE mask.

Mark Register: The one-byte (8 data-bits plus 1 parity-bit) mark register (MR) is used during store operations to specify which byte(s) of the storage bus in (SBI) is to be stored. Each bit in the mark register corresponds to one byte in the storage word. Each one-bit in the mark register specifies a byte in the specified address that is to be replaced by the corresponding byte on the SBI.

Return Address Registers: There are two 5-bit return address registers, X and Y. They indicate the register to which a fetched storage word is to be returned. The two registers are necessary because of the overlapping storage operations. These registers are used alternately with each successive storage request.

Storage Address Register: The storage address register (SAR) is a 24 data-bit plus 3 parity-bit register that holds all addresses for core storage operations initiated by the central processing unit (CPU) and performed by the bus control unit (BCU).



(1) Y and Z are loaded with L1 and L2 during I-time (Figure 52).

(2) AA adds contents of B1 to D1 and L1 to calculate effective address for first operand fetch.

(3) H 21-23, the first operand starting byte address, is set into T.

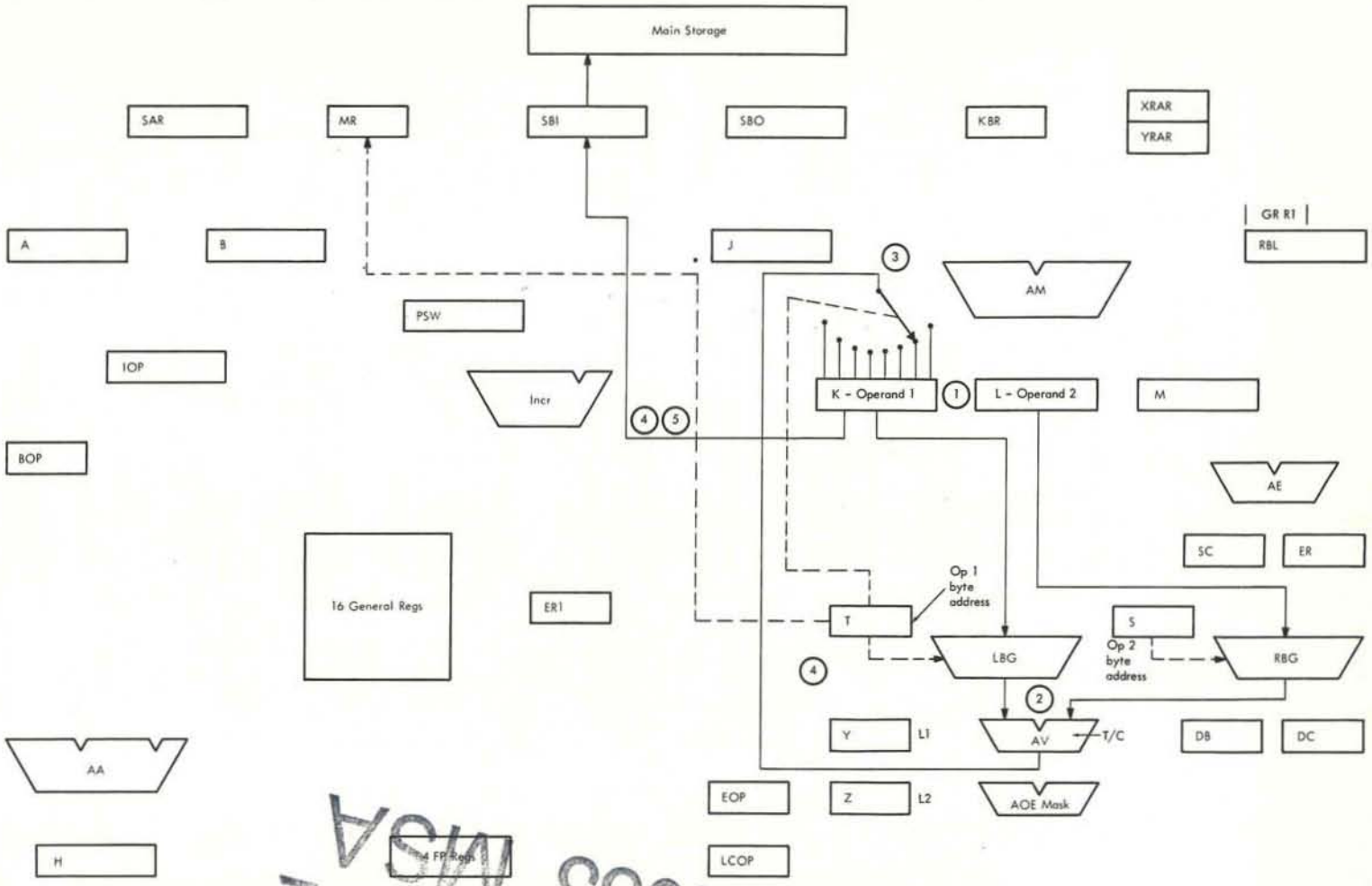
(4) Operand 1 returns to J and is set into K.

(5) AA adds contents of B2 to D2 and L2 to calculate effective address for second operand fetch.

(6) H 21-23, the second operand starting byte address, is set into S.

(7) Operand 2 returns to J and is set into L.

FIGURE 61. BASIC VFL ADD/SUBTRACT SET-UP



(1) Set-up has placed operands in K and L and starting byte addresses in T and S.

(2) One byte from K and one byte from L are gated to the decimal adder through the left and right byte gates (LBG and RBG).

(3) Sum from the decimal adder is routed back into the byte of K that contained the Op 1 byte just routed to the adder. The mark-bit corresponding to this byte is set.

(4) T, S, Y, and Z are decremented and the add cycle is repeated until Y and Z indicate that all bytes have been processed. During the processing, new Op 1 and Op 2 words are fetched whenever necessary. If a new Op 1 word must be fetched, the partial result in K is first stored.

(5) When Y and Z indicate that all bytes have been processed, K is stored to put the final result bytes back in storage.

FIGURE 62. BASIC VFL ADD/SUBTRACT ITERATIONS

Storage Bus In Latch Register: The storage bus in (SBI) latch register, located in the MCF unit, is an eight byte (64 data-bits plus 8 parity-bits) buffer between its inputs from the E unit K register and the channel storage bus in, and its output, the storage bus in (SBI) bus. The storage bus in (SBI) bus feeds core storage.

Storage Bus Out Latch Register: The storage bus out (SBO) latch register contains 64 data-bit latches and 8 parity-bit latches (eight bytes). The storage bus out (SBO) serves as a buffer for data between high-speed core storage, large capacity core storage, system control panel keys, and the central processing unit (CPU) instruction buffer (AB registers), operand buffer (J register), and the channel bus out.

Purpose of I Unit Functional Units

- I unit functional units consist of registers, arithmetic units, and controls to handle instruction preparation.
- Address adder
- AB registers
- BOP register
- BR1 incrementer
- ER1 register and incrementer
- Floating-point registers
- Gate select adder and register
- General purpose registers
- H register
- Incrementer
- Incrementer extender
- IOP register
- Program status word register

The instruction unit (I unit) contains registers, arithmetic units, and necessary controls to handle instruction sequencing, address preparation, and some instruction executions. The bus control unit (BCU) is sometimes considered a part of the I unit. The I unit also contains the general purpose registers (GPR), the program status word (PSW) register, and the central processing unit (CPU) clock.

Address Adder: The addressing adder is a three input, 24-bit adder for address arithmetic. Two 24-bit inputs are provided by the low-order 24-bits of the GBL and GBR. The third input comes from the IOP displacement (D) field and is added to the 12 low-order bits (bits 13-24). Thus, an index quantity and a base address each from a general register are simultaneously added to the displacement.

Eight-bit parity is maintained through the adder and input parity is checked. A check is also performed on the internal carry-lookahead circuits.

AB Registers: The A and B (AB) registers each hold 64 data-bits and 8 parity-bits. The AB register is used as a buffer for pre-fetched instructions. The A register obtains its data from the even storage locations while the B register receives its data from the odd storage locations. The 64 bit plus 8 parity-bit data requested by a branch instruction may be transferred from the J register to the AB register on a successful branch instruction. The output of the AB register is gated to the I operation (IOP) register.

BOP Register: This 12-bit register serves as an operation register for those instructions executed by the I execution (I-E) unit. BOP contains the op code and the R1 field.

BR1 Incrementer: The BR1 incrementer is a 4-bit, latched output half-adder, and provides for incrementing, by one, the R1 address field of the BOP register during the store multiple instruction and all instructions requiring a R1 + 1 operand.

ER1 Register and ER1 Incrementer: The ER1 register is a four-bit register used to hold the R1 address field after instruction control is passed from the I unit to the E unit. The ER1 register indicates the general or floating-point register which receives the instruction result.

The ER1 incrementer is a four-bit, latched output half-adder. The half-adder provides for incrementing the ER1 register by one during multiple load instructions and instructions requiring an R1+1 operand.

Floating-Point Registers: There are four floating-point (FLP) registers each containing eight bytes (64 data-bits plus 8 parity-bits) of information. The floating-point registers serve as source and destination (accumulators) for floating-point instructions. The K register and exponent register (ER) are the only input sources for the floating-point registers, and their output is gated either to the J register or

the M register via the RBL register. The floating-point register addresses are 0, 2, 4, and 6.

Gate Select Adder and Register: This three-bit adder and register provides a means for advancing the ICR by one to three halfwords. The result is returned to the gate select register (GSR) where it is decoded to select the proper 32-bit AB register output to IOP. The decoder off the gate select register selects thirty two bit fields in the AB register for gating into IOP as follows:

GSR	Field Selected
000	A00-31
001	A16-47
010	A32-63
011	A48-63, B00-15
100	B00-31
101	B16-47
110	B32-63
111	B48-63, A00-15

General Purpose Registers: The 16 general purpose registers (GPR) each contain 32 data-bits and 4 parity-bits. The general purpose registers may contain index quantities, base addresses, or fixed-point arithmetic operands. To facilitate the various uses, separate GPR address decoders are provided on the IOP R2 (or X) field, the IOP B field, the BOP R1 field, and the ER1 register. The first three are select out gates; ER1 gates into the GPR.

The general purpose registers are loaded from the high-order half (bits 0-31) of the K register located in the E unit. There are two independent 32-bit out buses; general bus left (GBL) and general bus right (GBR). Any two general purpose registers may be simultaneously gated onto general bus left and general bus right. General bus left and general bus right are gated into the high and low order halves, respectively, of the register bus latch (RBL) register.

H Register: The H Register (HR) is a 24 data-bit plus 3 parity-bit register used to hold addresses during branch, shift, multiple load/store, and other instructions.

Incrementer: The incrementer (incr) is a 24-position adder with latched outputs. A 24-bit field from either the instruction counter (IC) register (PSW positions 40-63) or the H register is incremented by 0, 8, 16, or 24. The result is returned to the H register, the storage address register (SAR) or the ICR, as well as the E unit K register high-order bit positions (bits 0-31).

The incrementer is used to update the ICR on high order advances and to generate addresses from the ICR for instruction requests. During branch instructions, the incrementer adds eight to the H register value for a branch plus one request (BR + 1). The incrementer is also a path from the H register to the ICR for inserting the new instruction counter value on a successful branch. Multiple load and store instructions use the incrementer to generate the request or store addresses. The path used is from the H register to the incrementer, where 8 is added, and then to SAR and the H register. The incrementer also has gates to the K register which are used in transferring VFL addresses. On a load PSW instruction, the incrementer provides a means of parity checking the new PSW. Parity checking is done in two steps. Likewise, on a store PSW instruction, the PSW is transferred through the incrementer in two steps.

Incrementer Extender: The incrementer extender is an eight-bit extension of the incrementer used as a transfer path for the PSW to the E unit K register for the store PSW instruction. It is also used for loading the PSW register from storage and for parity checking of the PSW.

IOP Register: The I operation (IOP) register holds a one or two halfword instruction for initial decoding and I unit processing. Provisions are made to logically OR bits 24-31 of a specified general register with the contents of IOP positions 8-15 on an execute instruction.

Program Status Word Register: The program status word (PSW) register contains the program status word. The register is eight bytes in length (64 data-bits plus 8 parity-bits), and may be loaded from the J register. Either half of the PSW may be stored into the high-order half (bits 0-31) of the K register via the incrementer. Positions 40-63 of the PSW serve as the instruction counter.

Purpose Of E Unit Functional Units

- E-op register (EOP)
- Exponent adder (AE)
- Exponent register (ER)
- J register
- K register
- L register
- Last cycle op register (LCOP)

- M register
- Main adder and shifter (AM)
- Register bus latch (RBL)
- Shift counter register (SC)

E Unit Operation Register: EOP is an eight data-bit plus parity-bit register which contains the operation code during E time. The EOP register is set from the I unit IOP register at least one cycle before the I to E transfer.

Exponent Adder: The AE is an eight-bit binary adder used primarily for the calculation of floating-point result exponents. In addition, it is used to form the exponent difference to determine floating-point add/subtract preshift requirements and for reducing this preshift and other shift counts. One input is a true/complement input and its latched output (AEOB) feeds the exponent register and shift counter. The exponent adder parity is checked in a manner similar to that of the main adder.

Exponent Register: The ER is an eight bit register used to hold the result exponent during floating-point instructions. Its output feeds both exponent adder inputs and bits 56-63 of the floating-point registers; its input is the output of the exponent adder output latches (AEOB).

J Register: This register receives operands from core storage via the storage bus out (SBO) latch register and general registers (GPR) and floating-point (FLP) registers via the register bus latch (RBL) register. The multiplier is contained in the J register during multiply operations, and the quotient during divide operations. The J register is shifted right four bits or left four bits via the register bus latch (RBL) register. The right four (R4) shift is accomplished by an unconditional left four (L4) shift to the RBL register and then a right eight (R8) shift back to the J register. The L4 shift is the unconditional L4 shift of the J register to the RBL register and a straight transfer from the RBL register to the J register.

Single and double word operands are passed from the J register through the main adder to the K register on load and store operations. The storage operand for fixed-point add, subtract, and compare instructions (RX format) is contained in the J register.

K Register: This register is an originating point from which stores are made to core storage via the storage bus in (SBI) latch register, and directly to the floating-point and general registers. During multiply, the product is located in the K register, and during divide, the K register temporarily retains the divisor and contains the dividend. The output of the K register can be gated right four (R4), left two (L2), or straight to the main adder. It can accept the full eight-byte (64 data-bits plus 8 parity-bits) output from the main adder sum latches (AMOB), and in addition, it is provided with output gates for variable field length (VFL) operations and divide divisor gating to the digit buffer (DB) and digit counter (DC) via the VFL gates. The K register is also set by bytes from VFL operations.

L Register: This register is set from the main adder sum latches (AMOB), and its entire output may be shifted left one position or its low-order 40-bits (24-63) may be shifted left three positions to the main adder. The three-halves and three-fourths divisors are obtained by gating the L register straight or right one respectively to the main adder. The L register also holds the 12 times multiplicand on multiply operations. The L register feeds the right byte gate (RBG) for VFL operations.

Last Cycle Operation Register: The LCOP is set directly from EOP so that EOP can simultaneously hold the operation code of the next sequential instruction. This allows the E time of one instruction to be completed while the E time of the next instruction is being set up for execution.

M Register: M is a working register which is involved in most shift operations. The M register receives data from the general purpose and floating-point registers via the RBL register. The M register has the ability to gate its contents out one, two, or three positions to the right. When this gating is used in conjunction with the shifter in the main adder (AM), a right shift of 1-8 or a left shift of 1-8 is possible.

The M register contains the multiplicand during multiply operations, temporarily retains the dividend, and contains the divisor during divide operations. In addition, its full word transfer path has the ability to send the high-order half (bits 0-31) of the register to the normal input of the main adder, and the low-order half (bits 32-63) of the register to the true/complement input of the main adder (AM) during the RR format fixed-point add/subtract instructions.

Main Adder and Shifter: The main adder (AM) is used for fixed-point and floating-point arithmetic. It is a two-input (64 bits plus 8 parity-bits), parallel carry-propagate binary adder. Its inputs come from the J, K, L, and M registers. Parity is checked on a byte basis and is combined with the parity of the

inputs to produce a predicted sum byte parity. The predicted parity is compared with the actual sum parity for checking purposes. The main adder output can be gated straight or shifted to the main adder output latches (AMOB), which can be transferred to the K, L, or M register.

The main shifter is logically located between the main adder inputs and the adder sum latches (AMOB). The shifter provides for shifting the eight bytes either right four, eight or 32 bit positions, or left four or eight bit positions. Parity checking is accomplished by matching an independently derived predicted parity with a generated output parity.

Register Bus Latch Register: The RBL register is an eight-byte (64 data-bits plus 8 parity-bits) latch register which provides buffering between its inputs: the floating-point (FLP) registers, the general purpose registers (GPR), the J register, and the K register; and its outputs: the J register and the M register. The RBL register allows a transfer from one of its input registers to one of its output registers while the input register is being set with information from another source.

Shift Counter Register: The SC register is an eight-bit register used to hold the shift count for multiply, divide, floating-point add/subtract, and for shift operations. The input to the shift counter is supplied by the exponent adder (AE). The SC feeds the exponent adder and the shift decoder.

Purpose Of VFL Functional Units

- AOE mask
- Decimal adder (AV)
- Digit counter (DC) and digit buffer (DB)
- Direct data register (DD)
- Left byte gate (LBG)
- Right byte gate (RBG)
- T and S pointers
- Y and Z counters

AOE Mask: All VFL logical operations are performed by the AND-OR-Exclusive OR (AOE) logical unit. The masking function for test under mask, the insert key operation (fetched storage protect key) and the read direct storing of the direct data lines are all executed through the AOE mask unit. The AOE results generate a parity bit. Checking of the operation is accomplished by passing the input operands to the decimal adder and utilizing the half-sum parity-check of the adder.

Decimal Adder: The VFL arithmetic operations are performed one byte (two digits) at a time by the decimal adder (AV). The AV is an eight-bit binary adder with a decimal correction made on the output sum. The adder input gates on both sides are split for the two digits in each byte. The right hand adder inputs are gated either true or one's complement. On true add, six is added to the right digit inputs to force decimal carries. This causes sums of ten and above to be correct, but sums of less than ten must be reduced by six by the decimal correction circuits.

Digit Counter and Digit Buffer: The DC is a four-bit counter, and the DB is a four-bit register. They are used for decimal multiply and divide operations, and in combination during edit, move with offset, pack, and unpack operations. The digit counter counts multiplier digits during multiply and generates the quotient digits during divide. The first complete quotient digit is held in the digit buffer while the second digit is generated in the digit counter. Parity is generated for the complete byte as the digits are generated. When the DC and DB are filled, the byte is transferred to the K register.

During edits, the DB and DC hold the fill character; during move with offset, the DB and DC hold the high-order digit of a byte being shifted left. The digit in the DB is gated to the decimal adder as the low-order digit on the following cycle. For pack operations, the even numbered digits are held in the DC and their parities are held in the DB; they are gated to the decimal adder and combined with the odd numbered digits to form packed bytes.

During unpack, the high-order digit of a packed byte is contained in the DC and their parity is held in the DB while the low-order digit and its parity is sent to the unpacked byte in the K-register. During the next cycle, the digit in the DC and its parity is sent to the next sequential byte in the K-register.

Direct Data Register: The DD register is a one-byte register without a parity-bit. The DD register is set with a byte from the K register on a write direct instruction. The contents of the DD register remain fixed until another write direct is executed.

Left Byte Gate: The 72 bits from K and 8 bits plus parity from DB/DC are brought into the LBG. Two gating triggers determine whether K register is gated with the T pointer or DB/DC is gated through the LBG. The value in the T pointer determines which byte of the K register is gated through the LBG.

Right Byte Gate: All 72 bits from K and L registers are brought into RBG. Two gating triggers, "gate K with S" and "gate L with S " determine whether K or L register is gated by the S pointer. The value in the S pointer determines which byte, 0 through 7, of the K or L register is gated through the RBG.

T and S Pointers: The T and S pointers are 3-bit counters containing the byte addresses of the VFL operands 1 and 2 in the K and L registers, respectively. These counters are capable of counting up or down and control the K and L register byte gates. The S counter also has the ability to gate bytes of the operand in the K register for overlapped operands.

Y and Z Counters: The Y and Z counters are 4-bit counters containing the initial field lengths from the IOP register. They are decremented and determine when the VFL decimal operation is completed. For decimal divide, L2 is placed in the Y counter and incremented until L2 is equal to L1. During logical operations, the counters are combined and used as an eight-bit counter.

ADDERS

Addressing Adder

- The addressing adder is a three input adder.

The addressing adder is a 24-bit, three input adder used for address calculations. The three inputs are required for calculating the operand address in the RX format (X + B + D) and the SS format (L + B + D). Figure 6000 is a flow diagram of the entire addressing adder, and Figures 5000 and 5001 are simplified logic diagrams, showing sum generation and parity prediction.

Inputs

The first input to the adder consists of the final OR for the general bus left (GBL). The second input consists of the final OR for the general bus right (GBR). A 9-bit bus containing a length field from variable field length (VFL) is also OR'ed into this input at bit positions 23-31. The third input is fed by the instruction operation (IOP) D field in positions 20-31 and by the interrupt controls at positions 25-28. The interrupt controls have access to the addressing adder for generating implied addresses during interrupts. Bits 0-7 of the general bus left and general bus right do not feed the addressing adder but their final OR's are located on an adder board. The output of the 36 bit (32 data-bits plus 4 parity-bits) final OR's for general bus left and general bus right feed the adder and condition the gates to the register bus latch. Final OR's for bits 2-7 are also sent to the program status word for the program mask. The OR's for bits 24-31 are sent to the instruction operation register for use during the execute instruction.

The Sum

The addition of the three inputs is accomplished by the use of a serial combination of a carry save adder and a carry propagate adder. The carry save adder (1 on Figure 6000) combines the three inputs into two outputs; namely, the sum per bit position and the carry per bit position. The sum output of the carry save adder (CSA) is the "exclusive OR" of its inputs and becomes one input to the carry propagate adder (CPA). The carry output represents a carry generated from the particular bit position and becomes the second input to the carry propagate adder after being shifted one position to the left (towards the high order end of the adder). In other words, the carry from one carry save adder bit position becomes an input to the adjacent higher order bit of the carry propagate adder. The carry is generated if two or more carry save adder inputs are "ones." The carry propagate adder forms the binary sum of its two inputs by parallel carry lookahead techniques.

An example of the operation of the carry save adder-carry propagate adder combination is:

	1	0	1	0	GBL input
	0	0	1	1	GBR input
	0	1	1	1	IOP D input
	1	1	1	0	CSA Sum
0	0	1	1		CSA carry (shifted)
1	0	1	0	0	CPA sum

Checking

The checking stations within the carry save adder-carry propagate adder consist of a half sum parity check (16 on Figure 6000) in the carry propagate adder and a serial carry check (18 on Figure 6000) in the lookahead. The half sum is the "exclusive OR" of the input data to the carry propagate adder. The parity of the half sum may be predicted as the "exclusive OR" of the input parity bits; which in this, is the sum and carry parity from the carry save adder. The predicted half sum parity is compared to the actual (generated) half sum parity. Any discrepancy between the two might be traced to an odd number of errors in any of the following areas:

1. The input data
2. The carry save adder
3. The bit lookahead in the carry propagate adder
4. The half sum generation in the carry propagate adder
5. The duplicate carry logic in the carry save adder
6. The carry save adder sum parity prediction
7. The carry parity generation in the carry save adder
8. The half sum parity generation in the carry propagate adder
9. The half sum parity prediction in the carry propagate adder, or
10. The checking circuitry.

The carry lookahead check involves a comparison between the group input carry and the carry out of the high order bit within the adjacent lower order group. Since a carry out of the aforementioned bit

is, in effect, a carry to the high order group, it should be equal to the carry propagated from the lookahead logic. A discrepancy would catch an error in the carry lookahead circuitry.

The parity for the sum output of the carry propagate adder (13 on Figure 6000) is predicted from the bit lookahead functions (generate and transmit), from the half sums, and from the group input carry, all of which are checked by the previously described schemes. This predicted sum parity is thus independent of the sum generation logic. A parity check of the sum is done in the bus control unit (BCU) or at the incrementer depending on how the sum is used.

AND-OR-Exclusive OR

- The AND-OR-exclusive OR is an eight-bit latched data path.
- The function performed by the AND-OR-exclusive OR is controlled by the control lines.
- The byte output of the AND-OR-exclusive OR is gated to the digit buffer and digit counter or the K register.

The AND-OR-exclusive OR (AOE-Mask) is an eight-bit latched data path that provides the AND, OR, and Exclusive OR functions necessary to execute the VFL connective instructions. Similar to the decimal adder, the AND-OR-exclusive OR has a right and left data input. A data byte from the left byte gate or from the YZ counters are gated into the left AND-OR-exclusive OR input. A data byte from the right byte gate or from the direct data register is gated into the right AND-OR-exclusive OR input.

The function performed by the AND-OR-exclusive OR is controlled by the active status of the AND, OR, or exclusive OR (OE) control lines (Figure 5002).

OR Function: The OR function to the AND-OR-exclusive OR is always gated except when the AND-or-exclusive OR is active. The OR function is determined by the active status of the OR op and the OR or OE lines. When the OR function and one AND-OR-exclusive OR input is gated, the AND-OR-exclusive OR latches are set to the input bits; thus, the AND-OR-exclusive Or becomes a path for one data byte. With the OR function and both AND-OR-exclusive OR inputs gated, each AND-OR-exclusive OR latch sets to the OR'ed condition of the two input bits. For example, for each bit position, the AND-OR-exclusive OR latch is set to 1 if either or both inputs are 1; if both inputs are zero, the AND-OR-exclusive OR latch is reset to 0.

Example:

Right Input	0	1	0	1		0	1	1	0	
Left Input	1	1	0	0		0	0	1	1	
	1	1	0	1		0	1	1	1	AOE Latches

AND Function: When the AND function is gated, the AND-OR-exclusive OR latches are set to 1 for those positions where both input bits are 1's.

Example:

Right Input	0	1	0	1		0	1	1	0	
Left Input	1	1	0	0		0	0	1	1	
	0	1	0	0		0	0	1	0	AOE Latches

Exclusive OR Function (OE): The XI and XC instructions are the only instructions that use the exclusive OR function of the AND-OR-exclusive OR. The OR Op and OR or OE Op lines to the AND-OR-exclusive OR are active to gate the OR function. When the exclusive OR function is gated, the OR Op function control is suppressed and only the OR or exclusive OR control is active. For each position of the AND-OR-exclusive OR, the latch is set to 1 if either bit input, but not both, is a 1. If both input bits are 1's or both are 0's, the AND-OR-exclusive OR latch is reset to zero.

Example:

Right Input	0	1	0	1		0	1	1	0	
Left Input	1	1	0	0		0	0	1	1	
	1	0	0	1		0	1	0	1	AOE Latches

AOE Output

The data byte output of the AND-OR-exclusive OR is gated to the digit buffer (DB)-digit counter (DC) or to the K register. A parity generator monitors the latched output of the AND-OR-exclusive OR and provides correct parity for the output byte. When the AND-OR-exclusive OR is used with the insert storage key instruction, the parity bit from the direct data register is used instead of the generated parity.

AOE Mask

The test under mask instruction uses the AND-OR-exclusive OR circuits to select and test the bits of a storage byte to determine condition code settings. See test under mask instruction.

Decimal Adder

- The decimal adder is an eight-bit binary adder with two eight-bit inputs.

- Modified binary circuits are used for decimal addition.
- The adder is used as a decimal adder or as a latched data path.
- The adder adds two bytes and provides one byte sum.

The variable field length (VFL) decimal adder (AV), (Figure 2040) is an eight-bit binary adder with two data inputs and an eight-bit latched output. Internal circuit functions of the decimal adder are the same as eight positions of the main adder. The parity predict and output circuits, however, are modified to enable decimal operations. In addition, to provide carries from byte to byte a carry trigger is associated with the decimal adder. Because variable field length additions move through the operands serially, adding a byte at a time, a carry-out of the decimal adder is set into the decimal adder carry trigger and used as a carry-in during the next byte addition.

Each of the two eight-position inputs to the decimal adder is numbered zero through seven. A data byte gated through the left byte gate enters the left data input of the decimal adder. A data byte gated through the right byte gate enters the right data input to the decimal adder through the true/complement plus six gate. Because a data byte contains two four-bit binary coded decimal digits, and because a digit is independently gated through the right byte gate and/or the left byte gate, each digit within a byte is identified as the high-order digit (HOD) and the low-order digit (LOD). Positions zero through three are the high order digits of either input to the decimal adder and positions four through seven are the low order digits.

The decimal adder is used to add or subtract two eight-bit decimal bytes and provides an eight-bit sum, or it is used as a data path for one data byte or one four-bit digit. Data are gated into and out of the decimal adder in either binary or decimal form. Input and output controls to the decimal adder enable it to be used as a latched data path for one byte or one digit of either binary or decimal data, and as a decimal adder when performing decimal arithmetic. When used as a latched data path, each data bit is gated through the decimal adder unchanged. When used as a decimal adder, excess-six (TC + 6) gating on the right side input and decimal correction on the latched output provide decimal correction to the binary adder.

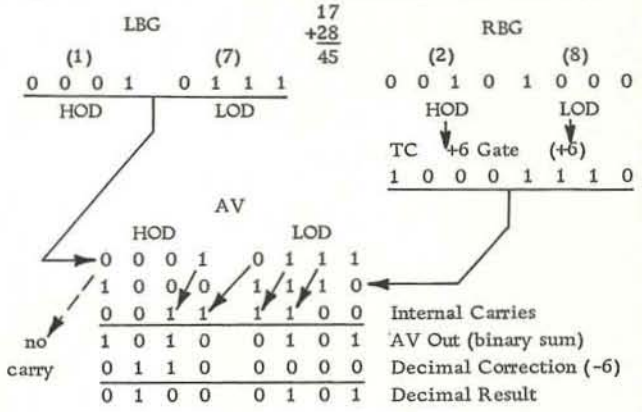
True-Complement Plus Six Gate (TC + 6)

The true-complement plus six input gate (Figure 5003) to the right side of the decimal adder provides the plus-six correction to the high-order digit and low-order digit inputs for decimal addition, and provides complement control for subtraction. Bit seven of the low order digit (bits 4-7) has a separate complement control used to invert the sign of a decimal result, if needed.

When binary coded decimal (BCD) numbers are added in binary adder, the result digit is an invalid binary coded decimal number if the bit-sum is greater than nine. Consider the addition of decimal numbers 17 plus 28:

17	=	0	0	0	1	0	1	1	1
+28	=	0	0	1	0	1	0	0	0
45		0	0	1	1	1	1	1	1
		HOD				LOD			

In the example, the low-order digit contains the correct binary sum of seven plus eight; however, the bit structure of the low-order digit does not conform to the binary coded decimal coding for decimal numbers. Note that the high-order digit (decimal 10's position) is one short and the low-order digit (decimal ones position) is excess ten. This has occurred because a four-bit binary position does not carry out until the sum exceeds 15 (1111). In decimal arithmetic, a carry out of any single position occurs when the sum exceeds nine. Therefore, when decimal addition is performed, the TC + 6 gate on the right input of the decimal adder elevates the value of the high order digit and the low order digit plus six. In this manner, a carry out of each digit position is forced when the decimal sum exceeds nine. Consider the problem of 17 + 28 as executed on the 360/75.



In the example, the decimal number 17 is gated through the left byte gate to the left input of the decimal adder, and the decimal number 28 is gated through the right byte gate and the TC + 6 gate to the right input of the decimal adder. The TC + 6 gate is controlled to gate decimal and adds plus 6 to the value of each decimal digit. Through the internal circuits of the decimal adder, the inputs from the left byte gate and the TC + 6 gate are added in binary. Bit carries are generated and transmitted, as shown in the above example, to combine with the input sums and produce a binary-sum decimal adder output for each decimal digit. Because the decimal sum of the low order digit inputs (7 + 8) exceeds 9, the +6 added to the low order digit forces a carry from the low-order digit to the high-order digit. The decimal adder low-order digit output sum is the correct decimal digit and does not require additional decimal correction. The high-order digit decimal adder output, in the example, requires decimal correction. Because the sum of the two high-order input decimal digits (1 + 2) does not exceed nine, the +6 added to the high-order digit input does not cause a carry-out from the high-order digit. Therefore, the binary-sum of the decimal adder high-order digit output is six greater than the decimal value. Decimal correction circuits on the decimal adder output then removes the excess-six to provide a true decimal value.

The decimal correction circuits (Figure 5004) on the output of the decimal adder monitor the carry-out signals from the high-order digit and low-order digit positions when either position is gated for decimal operations. A carry from either position indicates the decimal adder output sum for that position requires no correction. The absence of a carry indicates the output sum must be corrected by removing the excess-six.

Complement Control: The complement control to the TC + 6 gate enables the decimal adder to be used for decimal or binary subtraction. A separate complement control is provided for each input digit to the decimal adder (Figure 5003). When the complement control to either digit is active, the data bits of the digit are inverted to the binary one's complement.

Decimal subtraction is accomplished by the nine's complement method. Briefly, this entails the addition of the nine's complement (the difference between the decimal number and nine) of one decimal number and the true value of another decimal number. In binary subtraction, the one's complement of one binary number is added to another binary number. In either binary or decimal subtraction, the complement addition produces a result value equal to the difference of the two numbers.

For each four-bit decimal number, the nine's complement plus six is equal to the binary one's complement of the same number, see the following chart. For example, the nine's complement of the decimal number 7 (0111) is 2 (0010); 2 (0010) plus 6 (0110) equals 8 (1000); the binary one's complement of 7 (0111) is also 8 (1000). Therefore, the TC + 6 complement gate is used for both binary and decimal operations.

Decimal Number	Nine's Complement	+6
0 0 0 0	1 0 0 1	1 1 1 1
0 0 0 1	1 0 0 0	1 1 1 0
0 0 1 0	0 1 1 1	1 1 0 1
0 0 1 1	0 1 1 0	1 1 0 0
0 1 0 0	0 1 0 1	1 0 1 1
0 1 0 1	0 1 0 0	1 0 1 0
0 1 1 0	0 0 1 1	1 0 0 1
0 1 1 1	0 0 1 0	1 0 0 0
1 0 0 0	0 0 0 1	0 1 1 1
1 0 0 1	0 0 0 0	0 1 1 0

Although, during decimal subtraction, the TC + 6 gate does not add +6, the decimal adder output sums for each digit must be considered for decimal correction.

The same decimal correction rules apply for subtraction as applies to decimal addition. For each digit position, if a carry-out occurs, the decimal adder binary sum is the correct decimal sum; if a carry-out does not occur, the decimal adder binary sum is not the correct decimal sum, and receives decimal correction by removal of excess-six. The following example shows how decimal number 0784 is subtracted from 1728.

	2nd Byte Addition		1st Byte Addition		
	0 0 0 1	0 1 1 1	0 0 1 0	1 0 0 0	Left AV input
1728	1 1 1 1	1 0 0 0	0 1 1 1	1 0 1 1	Right AV input from TC + 6 (comp)
-0784	1 1 1 0	0 0 0 0	1 1 1 1	0 1 1 1	Internal AV carries
0944	0 0 0 0	1 1 1 1	1 0 1 0	0 1 0 0	AV binary sum out
	0 0 0 0	0 1 1 0	0 1 1 0	0 0 0 0	-6 decimal correction
	0 0 0 0	1 0 0 1	0 1 0 0	0 1 0 0	AV latch sum
	(0)	(9)	(4)	(4)	

Gate Binary True: The Gate Binary True controls to the TC + 6 gate enables either or both digits of a

data byte to be gated into the right decimal adder input with all bits unchanged. Two controls, Gate HOD Binary True and Gate LOD Binary True (Figure 5003), are independent; they are gated simultaneously separately, or in combination with decimal gating. For example, the low-order digit of each decimal operand contains the sign of the decimal number; if the data byte that contains the sign digit is routed through the decimal adder, the low order digit is gated Binary True to prevent the sign digit from receiving decimal correction. The high-order digit, in this case, is gated Decimal True and receives decimal correction. When the decimal adder is used as a data path for non-decimal data, then both controls, gate high-order binary true and gate low-order digit binary true, are active.

Invert Sign: The invert sign control (Figure 5003), when active, causes the low-order bit of a sign digit to be inverted. When a sign digit enters the right low-order digit input to the decimal adder, the polarity of the sign is changed by inverting bit seven's input in the TC + 6 gate.

Parity Adjust

Byte parity is adjusted whenever a partial byte is gated to the decimal adder or when the number of data bits is altered as they pass through the decimal adder. For those occasions when the decimal adder is used as a data path, byte parity is routed through without adjustment. When two bytes or partial bytes are added with decimal correction in the decimal adder, both input and output parity is corrected.

Right Side Parity Adjust: (Figure 5005). The right byte gate is connected to the TC + 6 input to the decimal adder. From the right byte gate, the data lines are split for the high-order digit (0-3) and the low-order digit (4-7). Through the right byte gate, the high-order digit and low-order digit are gated either together or separately. The byte parity is adjusted when a partial byte is gated or bits are altered at the decimal adder input.

When gating digits decimal true through the decimal adder, the TC + 6 gate adds plus-six to each decimal digit. Decimal digits four or five are the only ones that change parity when the plus-six is added. Two gating combinations that require parity adjustment are:

1. (HOD DT) · (LOD BT)--This gating is used when the low-order byte of the decimal operand is processed. LOD BT gates the sign digit to the decimal adder unchanged, while HOD DT provides decimal correction to the HOD of the byte (low-order decimal digit of the operand).

2. (HOD DT) · (LOD DT)-- This gating is used when both digits of the byte are gated decimal true; in this case, only decimal digits four or five change parity.

All other parity adjustments are made because either the high-order digit or low-order digit is not gated. Figure 9050 shows the possible gate combinations on the adjusted parity.

Left Side Parity Adjust: (Figure DM5006). The left byte gate is connected straight to the left input of the decimal adder. Left byte gate controls enable separate gating of the high order digit and the low order digit of the byte. Parity adjustment on the left decimal adder input has the following combinations:

1. P Straight--This parity gate is used when all eight bits (high-order digit and low-order digit) of the byte are gated to the decimal adder.
2. P_H--This gate is used to provide adjusted byte parity when the high-order digit of the byte is removed and only the low-order digit is gated to the decimal adder.
3. P_L--This gate is used to adjust byte parity when only the high-order digit of the byte is gated to the decimal adder.

For either case of two or three above, the data bits of the unused digit are exclusive OR'ed with byte parity to determine the correct parity for the partial byte used. For example, if only the low-order digit of the byte is gated to the decimal adder, the data bits of the high-order digit are used to adjust byte parity. This procedure precludes the possibility of adjusting parity upon a digit that has gained or lost a bit, then gate the digit through the variable field length circuits without detecting the error.

Decimal Adder Errors

When the decimal adder is used as a data path or when two decimal bytes are added, error detection circuits monitor the parity of the input data and the functions of the carry lookahead circuits. Incorrect parity of either input to the decimal adder causes a decimal adder half sum (HS) check. An erroneous carry from either digit within the decimal adder causes a carry error. The occurrence of either error condition causes a machine check, and unless masked out by the panel key or the program status word, an automatic log-out occurs.

Decimal Adder Half-Sum Check (HS Check): During each cycle that the decimal adder is used as a data path or decimal adder, the half-sum parity is checked with the input parity. A half-sum check occurs if either decimal adder input byte contains wrong parity.

For each bit position within the decimal adder, a half-sum is developed if either input, but not both, is a one. The eight half-sums, without carries, are exclusive OR'ed to determine the odd or even half-sum parity. Because half-sum parity is always the complement of the exclusive OR'ed input parity bits, the half-sum parity is compared with the complement of the exclusive OR'ed left byte gate and right byte gate parity bits (Figure 5007). An unlike condition causes a half-sum error.

Decimal Adder Carry Error: A decimal adder carry error indicates a failure within the decimal adder; a carry-out from either digit indicates the decimal adder has failed when needed or an extraneous carry has occurred.

Within the decimal adder, generate and transmit functions are developed for each bit. The bit-functions then become inputs to the carry-lookahead circuits. In the carry-lookahead circuits, the bit functions within each digit are AND'ed to provide bit-to-bit carries and a carry-out from the digit. For example, consider the low-order digit, bits 4-7; if the function of bits 4-6 is transmit and bit 7 generates (Figure 5008), a carry-out from the low-order digit occurs. At the same time and at different logic levels, a duplicate carry is developed and compared with the digit carry generated. Both carries should occur at the same time; if either carry occurs separately, the decimal adder carry error trigger is set and the execution unit check circuits are signalled.

Exponent Adder

- A seven position adder used primarily for exponent arithmetic.
- It is used to increment or decrement the exponent register or shift counter.

The exponent adder provides a nine bit, fully checked data path between the source registers and the exponent and/or shift counter registers. The low-order positions (1-7) comprise a seven bit binary adder, the eighth position (0) is used for floating-point operations, and the ninth position is the parity bit.

The primary function of the exponent adder is to perform exponent arithmetic for such purposes as incrementing and decrementing of shift amounts and iteration counts.

Data Flow

The basic data path (Figure 6001) is from the source register to the exponent adder input OR's, through the exponent adder to the exponent adder output bus (AEOB) latches, and then to the exponent and/or shift counter registers. The input OR's are referred to as the AE-OR and the AET/C-OR. The source registers to the adder inputs are:

AE-OR	AET/C-OR
J 0-7	M 56-63
J 32-39	FLP 56-63
ER 0-7	ER 0-7
	SC 0-7

Additional inputs to the AE-OR are received from the M and SC/H decrement decoder, and from the control area. The AET/C-OR receives additional inputs from the variable field length area in addition to the source registers listed.

Binary Adder

The seven position adder employs parallel carry lookahead to resolve the bit carry functions. The sums are generated as the "exclusive OR" of the data inputs and the bit carries.

Note on Figure 6002 that the two operands are combined to generate bit functions. The bit functions then produce the sum along two independent paths. One path produces the sum before carries (half sum). The other path, carry lookahead, resolves the carries to each bit position.

Half Sum: The half sum for each bit position is a function of the input bits which follow these rules:

AE Bit	+	AET/C Bit	=	Half Sum
0	+	0	=	0
0	+	1	=	1
1	+	0	=	1
1	+	1	=	0

The half sum is the "exclusive OR" ($\not\vee$) of the data inputs. Figure 5009 shows the logic used to produce the half sum for each position.

Bit Functions: The carry to each bit position (bit carry) is a function of the input bits and is developed by the lookahead circuits. Each position delivers a carry to the next higher order position according to the following rules:

AE Bit	AET/C Bit	
0	0	Will never produce a carry
0	1	Will produce a carry if the position receives a carry
1	0	
1	1	Will always produce a carry

Note that two different predictions concerning a carry from each position can be made:

1. If either input is a "one state," the position produces a carry if it receives a carry. The position is said to transmit (T).
2. If both input bits are in a "one state," the position always produces a carry. The position is said to generate (D).

Figure 5009 shows the logic used to produce the transmit and generate functions for each bit position.

The "OR" function, rather than the "exclusive OR" function, is used in producing the transmit. This does not interfere with the operation of the lookahead circuits since the generate function determines the carry whenever both bits are present.

Carry-Lookahead: The bit functions for each position are delivered to the lookahead circuits. The lookahead circuits deliver a carry to each bit position if two conditions are met:

1. Some lower order bit position generates.
2. All bit positions, between the generating bit position and the position to receive the carry, transmit.

To implement the logic to resolve the bit carry to the high-order position, as thus far described, would require six AND circuits, any one of which could produce the carry. Further, the circuit testing for the condition in which the low order bit generates and all other bits transmit would require six inputs. These cumbersome circuits are avoided by uniting the adder into two groups; three positions to the first (5-7) and four positions to the second (1-4). Generate and transmit functions are produced for each group and these are combined to produce carries into the groups (group carry). The bit carries are developed from the group carries and the bit functions.

Figure 6003 is a block diagram of the lookahead function. Within these blocks are references to Figures 5010, 5011, and 5012 which show the logic within these blocks.

End Around Carry: The end around carry functions, when enabled, essentially removes the high/low order relationship of the adder positions with respect to the carry functions. That is, to resolve the carry to each bit position, all other positions are treated as lower order positions. For example, if position 4 generates and positions 3, 2, 1, 7, and 6 transmit, a carry is delivered to position 5. ($D_4 \cdot T_3 \cdot T_2 \cdot T_1 \cdot T_7 \cdot T_6 = C_5$)

The end around carry function is incorporated in the group carry generation and is enabled or disabled by the allow end carry (AEC) control function.

Sum Generation: The final sum is generated from the half sum and the bit carry, following the same rules used to produce the half sum. That is, the final sum is the "exclusive OR" of the bit carry and the half sum. Figure 5009 shows the logic used to produce the final sum.

Sum Parity: The generation of the sum parity depends upon a relationship that exists between the parity of the half sums, the parity of the bit carries, and the parity of the final sum. This relationship is:

$$\text{Parity}_{HS} \not\vee \text{Parity}_{Carries} = \overline{\text{Parity}_{Sum}}$$

The half sum parity and the parity of the carries within each group are determined comparatively early in the adder path. The group carries, developed by lookahead, are not available until later in the adder path. Therefore, the sum parity is developed as follows:

1. The parity of the carries within each group is generated (internal carries parity).
2. The parity of the half sum is determined from the parities of the input operands (predicted half sum parity). The relationship of the half sum parity to the parities of the input operand is:

$$\text{AET/C Parity} \not\vee \text{AE Parity} = \overline{\text{HS Parity}}$$

The predicted half sum parity is the "exclusive OR" of the input parities.

3. The parities generated in 1 and 2 are combined according to the relationship previously cited to produce the parity of the final sum before group carries (internal sum parity).
4. The parity generated in 3 is changed if the number of sums changed by the group carries is odd (change parity). This condition is determined by examining certain half sums within each group, and the group carries.

The sum parity, therefore, is a function of the following, and is generated independently of the sums themselves:

1. The parity of the internal carries.
2. The predicted half-sum parity.
3. The change parity.
4. The sign control and AEOB complement functions, not related to the add function. The manner in which these functions modify the parity is defined under "Sign Control" and "AEOB Complement." Figures 5013 and 5014 show the logic used to produce the sum parity.

Checking: The checking functions are designed to detect the following:

1. A mismatch of the input data and its associated parity.
2. A failure within the adder which could result in multiple sum failures.

Input Parity Check

The checking of the input parities depends upon the relationship of the input parities and the half sums, as previously stated. To check the input parities:

1. The parity of the half sums is generated.
2. The parity generated in 1 is compared with the predicted half sum parity. An inequality defines an error condition.

Figure 5015 shows the logic used to generate the half sum parity, Figure 5016 shows the comparison.

Bit Function Check

The adder contains circuitry which detects all single bit function failures. The half sum functions are generated in a manner such that a bit function failure causes:

1. The half sum functions to be the inverse of the proper result. This failure is detected by the input parity check.
2. The half sum and its complement to be equal. This failure, as well as a failure within the half sum parity generation circuitry, will cause the half sum parity and its complement to be equal. A comparison is made of the half sum parity and its complement. An equality defines an error.

Group Carry Check

The group carry developed by the lookahead circuits is equivalent to a carry from the high-order bit position of the lower-order group. This carry (K_G) is developed from the bit functions of, and the carry into, the high-order position of each group. The predicted group carry (K_G) is compared with the equivalent group carry and an inequality defines an error.

Byte Check

The bit function check and the group carry checks are OR'ed to produce the function byte check. This function defines a failure internal to the adder circuitry. Figure 5016 presents the logic used to implement the checking functions.

Sign Control

Sum 0 (sign) (Figure 5017) is a function of its half sum and the sign control functions. If the resulting sum is different from the half sum, the sum parity is changed. The following chart defines these relationships:

Control Functions	HS 0	Sum 0	Change Parity
None	0	0	No
None	1	1	No
Set Sign Plus	0	0	No
Set Sign Plus	1	0	Yes
Set Sign Minus	0	1	Yes
Set Sign Minus	1	1	No
Invert Sign	0	1	Yes
Invert Sign	1	0	Yes

Complement Gates

Complement gates at the exponent adder input and output provide the ability to perform complement arithmetic in both binary and "excess 64" forms.

AE Complement: A complement gate is implemented in the exponent adder following the "AET/C-OR." This gate extends the full width of the "AET/C-OR," including position 0 (sign). This allows for complementing the AET/C input operand, and is enabled by the select AE complement control function.

AEOB Complement: A complement gate is incorporated in the AEOB latch, providing for the recomplementation of the adder sum. It extends from position one through seven (position zero is excluded) and is split after position one. The two components, complement AEOB 1 and complement AEOB 2-7, are selected singly or together. The selection of the complement AEOB 1 gate causes the inversion of an odd number of sums (1), and a corresponding change in the sum parity. This feature provides the ability to perform the arithmetic operations defined later in this text.

Output Signals

Certain conditions are detected by the exponent adder circuitry which are of significance in the performance of particular instructions.

Half Sum One's Detector: A signal is generated when the half sums of positions one through seven are all one's. With the exponent adder complement gate selected, the presence of the half sum positions one through seven being all ones signal indicates the equality of the two input operands (positions 1-7).

High Order Carries Detector: The carries from position 1 and position 2 are detected by the adder look-ahead circuits (Figure 5012).

Overflow/Underflow Detector: The result of exponent arithmetic can, of course, exceed the limits of exponent values. This occurrence is detected by the adder circuitry. A result in excess of the upper value limit defines an overflow condition. A result which exceeds the lower value limit defines an underflow.

Floating-Point Operation

For all floating-point operations, the exponent adder performs "excess 64" arithmetic. All adder operands, including increment and decrement amounts, are considered as "excess 64" values. Four operations are performed by the exponent adder:

1. Exponent transfer
2. Exponent comparison
3. Exponent subtraction
4. Exponent addition.

Exponent Transfer: The exponent adder serves as a data path through which the operand passes to the exponent register and/or the shift counter register.

Exponent Comparison: The adder result is equal to the absolute difference of the input operands. This result is obtained as follows:

A complement add, with end around carry enabled, is performed. The result is the true binary difference if the exponent adder input operand is the larger. Complementing Sum 1 (Compl AEOB 1) yields the desired result.

The result is the complement of the binary difference if the AET/C input operand is the larger. Recomplementing yields the true binary difference. Complementing Sum 1 again yields the desired result. However, this is the same result as obtained by complementing sums 2-7 (Compl AEOB 2-7).

Therefore, the sums are complemented dependent upon the form of the add result, defined by the carry from position 1. Examples are shown in Figure 9051.

Exponent Subtraction: The adder result is equal to the algebraic difference of the exponent adder input operand subtracted from the AET/C input operand. The result is obtained as follows:

A complement add is performed and sums 2-7 are complemented (AEOB Compl 2-7).

Exponent subtraction is performed to resolve the result exponent for divide, and to decrement exponents, preshift amounts and iteration counts. Figure 9052 presents some examples.

Exponent Addition: The adder result is equal to the sum of the input operands. The result is obtained as follows:

A true add is performed and sum one is complemented (Compl AEOB 1). Exponent addition is performed to resolve the result exponent for multiply and to increment exponent values. Figure 9053 presents some examples.

Binary Operation

For all non-floating-point operations, the exponent adder performs as a conventional binary adder.

Gate Select Adder

- Central to data loop used to change gates to IOP at TN T2.
- Central to data loop used to update ICR L0 at I to E transfer.
- Constantly looks at three prelatch positions and two control inputs and delivers three sum bits and a carry to the CSR.
- Corrected parity for the changed byte (eight bits) is delivered with the sum.
- The prelatch receives inputs from either ICR 20-22 or the CSR.

The use of the gate select adder in the 2075 and the data flow in the adder are shown in Figure 5018.

Incrementer Adder

- The incrementer is a 24-bit adder.
- There is an eight bit extension on the high-order end.
- There are three inputs to the incrementer.
- The incrementer results are checked on a byte basis.

The incremented (INCR) is a 24-position adder with latched outputs. The incrementer adds zero, eight,

sixteen, or twenty-four to its input data. Bits 21, 22, and 23 are not added into, but are used in the data path and for parity purposes.

Incrementer	Data Inputs	0.....18	19	20	21	22	23
Adder	-8	0.....0	0	1	0	0	0
Inputs	-16	0.....0	1	0	0	0	0
For	-24	0.....0	1	1	0	0	0

There is also an eight bit extension to the high-order input of the incrementer. It is used to transfer the program status word to the K register and to generate or check parity on these eight bits.

The incrementer is used to update the instruction counter on high-order advances and to generate addresses from the instruction counter for instruction fetches. During branch instructions the incrementer adds eight to the H register value for the branch plus one fetch. It also is the path from the H register to the instruction counter for inserting the new instruction counter value on successful branches. Multiple load and store instructions use the incrementer to generate the fetch or store addresses. The path used is from the H register to the incrementer, where eight is added, and then to the storage address register and the H register. The incrementer output has gates to the K register which are used in transferring variable field length and decimal addresses from the addressing adder to the variable field length unit. On a "load PSW" instruction, the incrementer and incrementer extender are used to parity check the new program status word. As the program status word is 64-bits wide and the incrementer with the incrementer extender is only 32-bits wide, the check is done in two steps. The right hand program status word (bits 32-63) is checked first and then the left hand program status word (bits 0-31). Likewise, on a "store PSW" the program status word is transferred through the incrementer to the K register in two steps. See Figure 6004.

The H register, instruction counter, and program status word bits 8-31 are the three sources of input data to the incrementer. All three inputs are gated at their source, and feed a three way "OR" into the incrementer. The incrementer extender is fed by the program status word bits 0-7 and 32-39. Both of these are gated at the extender. The aforementioned gates to the K register are one output of the incrementer. All 24 incrementer bits are sent to the storage address register (SAR) and the H register. The incrementer data are gated at the storage address register and the H register. Incrementer bits 0-19 plus bit 23 are sent to the instruction counter register which does the gating in. Incrementer bits 0-20 are sent D.C. to the program store compare circuits. The 24 incrementer bits and the three parities have an input from the J register. This data are gated through the incrementer to the storage address register and the H register during scanning.

The 24 incrementer bits are packaged with a byte on each of three boards. Bits 0-7 are on 01G-B1, bits 8-15 are on 01G-A2, and bits 16-23 are on board 01G-A1. Bits 0-19 are broken into five four-bit groups for packaging purposes. Bit 20 is packaged separately. Carries into each group are generated independently by look-ahead circuitry. The carry into group 16-19 is the function: ADD 16 + (ADD 8 · INCR INPUT 20). The group carry into any other group is generated whenever all higher numbered bits up to 19 are 1's and there is a carry into group 16-19. For example, there would be a carry into group 8-11 if input bits 12-19 are all ones and 16 is being added. A carry into a bit position is generated if there is a group carry into the group and if all the bits in the group to the right of the bit in question are ones. For example, bit eight would have a carry into it if bit nine, ten, and eleven are ones and a carry into the group is present. Sum bit 20 is formed by exclusive OR'ing the +8 input with data input 20. Twenty-four is added to the incrementer, only during an IC HO advance. Since an IC HO ADV occurs only if the input to the incrementer position 20 (from ICR 20) is zero, it follows that it is never necessary to add 24 in the incrementer if the incrementer input from position 20 is a one. This simplifies the adder logic for bit 19 because there are never three ones into position 19 at the same time. The logic for position 19 becomes data input 19 exclusive OR'ed with (ADD 16 + ADD 8 · INCR INPUT 20). See Figures 5019, 5020, and 5021.

The incrementer results are checked on a byte basis by comparing a parity predicted from the input data to a parity generated from the incrementer sum. The predicted parities for bytes 0-7 and 8-15 are obtained from the following expression: INPUT PARITY CHANGE = (CARRY INTO 4-7) ($\overline{7}$ + $\overline{5}$ · 6 + $\overline{3}$ · 4 · 6 + $\overline{1}$ · 2 · 4 · 6). The bit positions in the formula shown are for byte 0-7. For byte 8-15 use the corresponding values. Byte 16-23 is unique in that the +8, +16, and +24 conditions are taken into account. Bits 21-23 do not change so they are not considered. The expression for the parity predict on byte 16-23 is: Input Parity Change = $\overline{20}$ (+8) ($+\overline{16}$) + $\overline{19}$ ($+\overline{8}$) (+16) + 17 + 19 (+8) + $\overline{18}$ 19 (+8) + $\overline{17}$ 18 ($+\overline{8}$) (+16) + $\overline{17}$ 18 $\overline{19}$ 20 (+16). This expression reflects the fact that bit 19 is not a full adder position. For each byte an eight way exclusive OR generates a parity from the incrementer sum bits and this generated parity is compared to the corresponding predicted parity. See Figure 6005.

Byte 16-23 is provided with a second predicted parity bit called P1. Whenever the instruction counter

register is set from the incrementer, either zero or sixteen is added to the incrementer input. The quantity 16 is added on an IC HO ADV only. An IC HO ADV takes place when bit 20 of the instruction counter register is zero. At the same time, as an IC HO ADV, the controls allow an instruction counter fetch address to be generated in the incrementer for setting into the storage address register. In this case the controls force a plus eight into the incrementer, resulting in a net +24. However, the plus eight must not effect the result sent to the instruction counter register. This is accomplished with the following features:

1. Bit 20 of the incrementer is not gated into the instruction counter register. Therefore, on an instruction counter HO ADV bit 20 of the instruction counter register remains a zero.

2. The predicted parity P1 is generated by using the plus 16 input and ignoring the plus eight input. Therefore the correct parity for bits 16-23 is set into the instruction counter register in all cases.

There are occasions when it is necessary to have the parity bit P1 available directly at the gate select register. Such a case occurs when the gate select register is set at the end of the IC HO ADV. There is not time for the new parity bit to pass through the instruction counter register and then the gate select register. Therefore, P1 of the incrementer is sent directly to the gate select register.

Main Adder-Shifter

- The adder is a 64-position binary adder with inputs from the J, K, L, and M registers, and outputs to the K, L, and M registers.

- The shifter provides shift amounts of L4, L8, R4, R8, and R32.

The main adder-shifter takes part in most data transfers within the execution unit. The operations performed by the adder-shifter, the controls used to execute each operation, and the data paths within the adder-shifter used for each operation are shown in Figure 63.

Data Paths and Control Scheme

The adder performs all of its operations with a minimum of control. The true add operation is the basic function of the adder and is performed with no control other than input and output gating. The heavy portions of the figure show the data path for an add operation. When operands are gated to each of the input OR's and no adder control is exercised, the sum of the operands is generated and gated to the main adder output bus latches. From the output bus latches, the sum is delivered to the K, L, and/or M registers.

To perform an operation other than the true add one or more of the controls indicated by the circled numbers is used. For example: to perform a shift, control number four sets the shifter to give the desired output. The same control gates the shifter to the main adder latch and blocks gating the final sum to the main adder output bus latches. The data path is from the bit functions to the normally selected ORE gate, through the shifter and the shift gate to the main adder output bus latch.

On adder operations, input parity is checked and output parity is generated. The check circuits require that correct parity is gated to both input OR's. For this reason, on single operand operations, such as shifting, parity is forced to the input OR not receiving the operand.

Data Flow and Timing Example

The adder is a one cycle data path. The sum of operands gated to the adder during a machine cycle is latched in the main adder output bus latch at L time following the cycle. This point is illustrated on Figure 6007 for the AR add instruction. The add takes place on the first fixed point cycle. The sum is delivered to the K register at A time of the next cycle.

On the AR add instruction, the adder receives inputs of two thirty-two bit operands and delivers a thirty-two bit useful sum. The operands are gated to the left or high order end of the adder input OR's. Parity is forced to the low order half of each input OR. A 64 bit sum having significant digits in the high order 32 bits is gated to the K register. Only the left 32 bits of the K register are gated to the general register, R1.

The control of gates to the adder is external to the adder. The gating depends not only on the adder operation being performed but also upon the context of the operation. The gating for the true add performed when the AR add instruction is being performed is shown on Figure 6007. The gating for a true add being performed during the execution of a multiply or divide instruction is different and may involve shifted as well as straight gates.

Bit Functions and the Add Operation

Note on Figure 63 that the two operands are combined to generate bit functions. The bit functions then produce the final sum along two independent paths. One path produces the sum before carries. The other path produces the carries to each bit position.

The carry to each position is determined without waiting for the generation of the final sums in the

To Use the Adder-Shifter

A → Both input ORs must receive either data or all parity bits.

B → The AM latch must be sent to a register and the register must be released.

To Add

No other control is needed. The data path is in heavy outline.

To Complement Add, Two's Complement

(1) → AM T/C OR input is inverted.

(2) → A "hot one" is forced to the low-order position.

To Complement Add, One's Complement

(1) → AM T/C OR input is inverted.

(3) → Allow end-around carry is active.

To Shift

Single operand is gated. Parity is forced to other input OR.

(4) → The needed shift control is activated; it:

Sets shifter for data shift. Gates shifter to AM latch. Blocks final sum gate to AM latch.

The data path is through the normally selected OR-E gate to the shifter.

To Do Connectives: OR-E, AND, OR

Data passes through the adder twice. On the first pass, the connective is performed but incorrect parity may be generated. On the second pass, correct parity is generated.

First Pass

For All Connectives:

(4) → R4 shift control is activated; it:

Sets shifter for R4 shift. Gates shifter to AM latch. Blocks gate of final sum to AM latch.

For OR-E:

No other control, OR-E bit function, is normally gated to the shifter.

For AND:

(5) → Gate AND is selected and gate OR-E is blocked.

For OR:

(6) → Gate AND is selected and gate OR-E is allowed to remain open.

Second Pass

For All Connectives:

(4) → L4 shift control is activated; it:

Sets shifter for L4 shift.
Gates shifter to AM latch.
Blocks gate of final sum to AM latch.

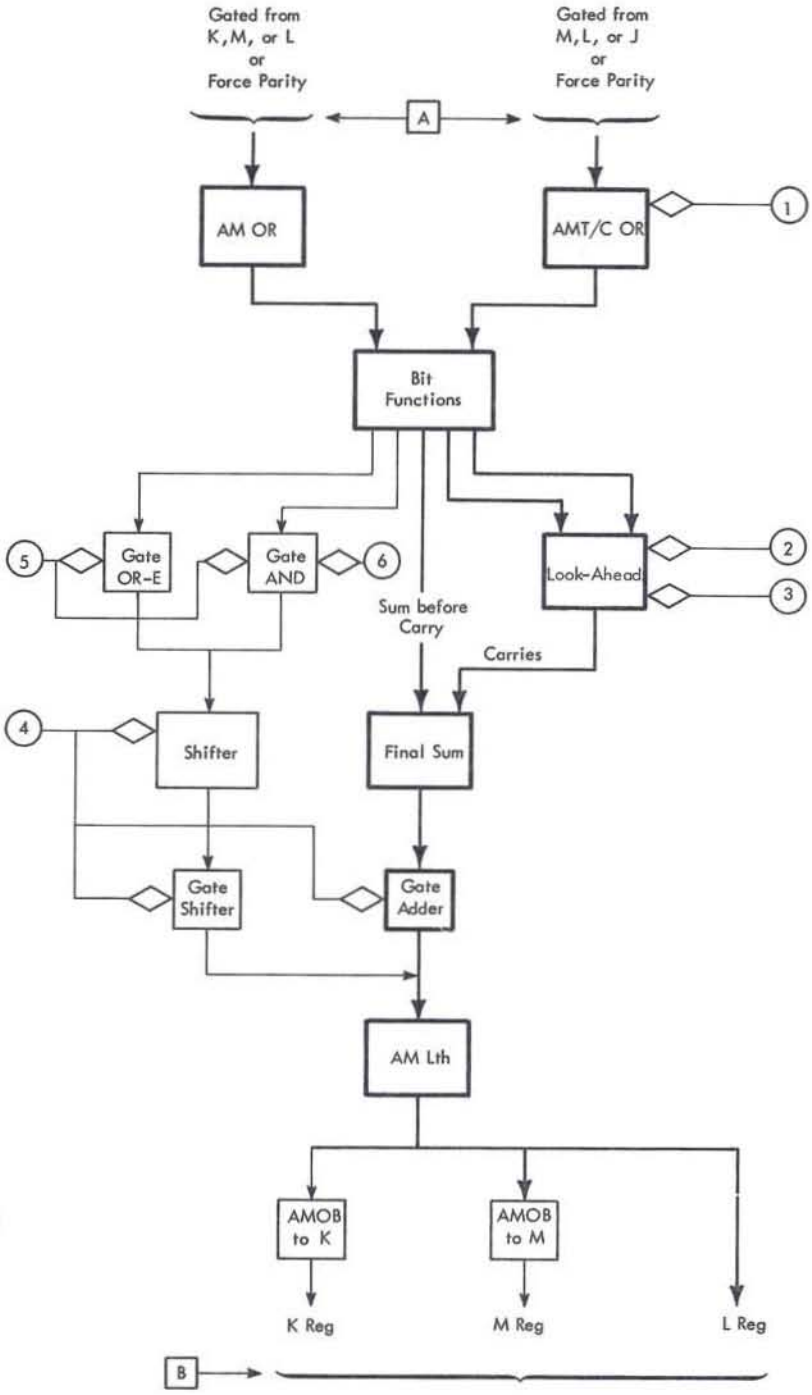


FIGURE 63. MAIN ADDER-SHIFTER: FUNCTIONS, DATA PATHS, AND CONTROL SCHEME

low-order positions. This fact is the time saver in the lookahead circuits.

The sum before carries for any bit position is a function of the input bits which follow these rules:

AM Bit + AMT/C Bit	=	Sum Before Carries (HS)
0 + 0	=	0
0 + 1	=	1
1 + 0	=	1
1 + 1	=	0

The sum before carries is the exclusive OR of the input bits.

Figure 9054 shows the logic used to produce the sum before the carries, called the half sum (HS), for each bit position.

The carry to any bit position is a function of the input bits to all lower order bit positions and is developed by the lookahead circuits using two bit functions from each bit position.

Any single position delivers a carry to the next higher order position according to the following rules:

0 + 0 will never produce a carry
1 + 0, 0 + 1 } will produce a carry if the position receives a carry.
1 + 1 will always produce a carry.

Note that two different predictions concerning a carry from any position can be made.

1. If either input bit is present the position always produces a carry if it receives a carry. The position is said to transmit (T).

2. If both input bits are present the position produces a carry. The position is said to generate (D).

Figure 9054 shows the logic used to produce the transmit and generate functions for each bit position.

The OR rather than the exclusive OR is used to produce the trasmit function. This does not interfere with operation of the lookahead circuits since whenever both bits are present the carry out is determined by the generate function and is not dependent upon the transmit.

The transmit and generate functions for each bit position are delivered to the lookahead circuits. The lookahead circuits deliver a carry to any bit position if two conditions are met:

1. Some lower order bit position generates.

2. All bit positions between the bit position to receive the carry and the generating bit position transmit.

The final sum is generated from the half sum and the carry by the same rules used to product the half sum from the two input bits. That is, the final sum is the exclusive OR of the carry and the half sum.

Lookahead for the Full Adder

To predict the carry into the high-order position of the 64-position adder using the logic thus far described would require 63 AND circuits, any one of which could produce the carry. Further, the circuit testing for the condition in which the low order bit generates and all other bits transmit would require 63 inputs. An adder using such circuits is said to have one level of lookahead. By one level is meant that only bit functions are used to predict the carries to all positions. The Model 75 uses three levels of lookahead and thus avoids the cumbersome circuits described above.

To achieve three levels of lookahead, the adder is divided into groups and sections, and the carries out of groups and sections are predicted by group and section functions.

A group contains four bit positions.

A section contains four groups.

Figure 9055 visualizes this breakdown for the entire adder.

Generate(D) and transmit (T) functions are developed for each group and section. These functions have the same meanings as the bit functions generate and transmit.

GENERATE (D) means that the section or group delivers a carry.

TRANSMIT (T) means that the section or group delivers a carry if it receives a carry in.

Figure 9056 shows how the functions and predicted carries on the three levels of lookahead relate to each other. The figures (5022, 5027) called out in each block show the circuit logic used to develop the function or the carry named in the block.

Figure 9057 shows the levels of logic and therefore the circuit delays involved in the generation of the half sum and the carry into the lookahead circuits. The half sum requires five levels of logic, and the carry requires nine levels. The final sum is developed in 11 levels.

Variations of the Add Operation

As shown on Figure 63, the true add is performed with no internal adder controls. Three other adder operations use the same data paths as the true add but require some control.

Complement Add: Complement addition is used to find the difference between two numbers. For the floating-point add and subtract instructions, one's complement addition is used when the difference is required. Other operations requiring the difference between numbers use two's complement addition.

For two's complement addition one operand is changed to its two's complement and all other adder

operations are for true add. The two's complement is the inversion of the number with a 1 bit added to the low order position. Controls 1 and 2 in Figure 6006 change the operand entering the true/complement input OR to its two's complement.

Control 1: "Sel AMT/C Compl" selects the inversion of the operand for the generation of the bit functions for the entire adder.

Control 2: "Hot 1" adds a one bit to the low-order position by forcing a group carry to group 60 to 63 and also causing the group to generate if all positions transmit.

For one's complement addition, one of the operands is changed to its one's complement and all other adder operation is as for true add. The one's complement is generated by inverting one of the operands and allowing a carry from the high-order position if generated, to be fed to the low-order end.

Control 1 "Sel AMT/C Compl" selects the inversion of the input for the generation of the bit functions.

Control 3 "Allow End Carry" enters the lookahead logic at the section level and allows a carry out of the high-order position to be fed back to the low-order position.

Adder As a Straight Data Path: To use the adder as a data path, the operand to be moved is gated to either input, and all parity bits and no data bits are gated to the other input. The operation is exactly as for the add. The sum is equal to the single input operand.

Parity Generation

The final sum of the two operands is generated by exclusive OR'ing the half-sum of the operands and the carries generated by the operands. The generation of parity for the final sum depends on the relationship that exists among the parity of the half-sum, the parity of the carries and the parity of the final sum. The relationship is:

$HS_p \nabla Carry_p = \overline{Sum_p}$

For example:

Bit Positions	1	2	3	4	Dec-Equiv.	P
A	0	1	0	1	5	1
B	0	1	0	0	4	0
HS	0	0	0	1		0
Carries	1	0	0	0		0
Sum	1	0	0	1	9	1

In practice, the half sum and the carries within each group are available early in the adder cycle. The carries into groups come from lookahead and are not available until later. The parity of the final sum is generated in four stages:

1. The half sum parity on each byte is generated.
2. The carry parity within each group is generated.
3. The parities generated in steps 1 and 2 are combined according to the relationship previously cited to generate the parity of the final sum before group carries.
4. When the group carries are available, they are used to change the parity generated in step 3, dependent on properties of the final sum before the group carries. The parity changes are predicted by examining certain of the half sums within each group and the carry into the group.

Entering into the generation of parity for each byte are three things:

1. The half sum parity of the byte.
2. The parity of the carries within each of the two groups of the byte.
3. Two "Change Parity Group" (CPG) lines generated by examining the carry into the group from lookahead and certain of the half sum within the group.

The output parity for group 08 to 15 is on Systems AM 157.

Input Parity Checking (Byte HS Parity Error)

The checking of input parities depends on a relationship that exists between the input parity of each operand and the half-sum of the operands. The relationship is:

Parity of A ∇ Parity B = $\overline{\text{Parity of HS}}$

For example:

	1	2	3	4	Parity
A	0	1	0	1	1
B	0	1	0	0	0
HS	0	0	0	1	0

$P_A \nabla P_B = \overline{P_{HS}}$

$1 \nabla 0 = \overline{0}$

$1 = 1$

To check input byte parity:

1. The half sum parity is generated for each byte.
2. The exclusive OR of the input parities for each byte is generated.

3. 1 and 2 are compared. If they are the same, an input parity error has occurred. The "byte HS parity error" indicator is set.
The circuit generating byte HS parity error for byte 08 to 15 is on Systems AM 163.

Bit Function Error

All outputs of the main adder-shifter depend upon bit functions generated for each input bit position. The adder contains circuits that detect any single bit function error.

The bit function check is performed in the following steps.

1. The AND and OR bit functions and their complements are generated.
2. From these the half sum and complement half sum are generated, independently of each other, for each bit position.
3. From the half sum and complement half sum generated in 2, the half sum parity for each byte and its complement are generated.

Any single error in steps 1 to 3 are detected by the "byte HS parity check" or the "bit function error" circuits as follows:

1. Single error in 1 causes HS = $\overline{HS}$, or HS and $\overline{HS}$ to be inverse of the proper result.
2. Single error in 2 causes HS = $\overline{HS}$.
3. Single error in 3 causes HS parity = $\overline{HS\ Parity}$.

In 1 and 2, when HS = HS then HS parity = $\overline{HS\ Parity}$ and the "Bit Function Error" circuit detects the error.

In 1, when HS and $\overline{HS}$ are inverse then the HS parity $\neq$ predicted HS parity and the "HS Parity Check" detects the error.

In 3, the "Bit Function Check" circuit detects the error.

The "HS Parity Error," "Bit Function Error", and "Byte Error" circuits for byte 8 to 15 are on Systems AM 163.

Lookahead Check

The lookahead circuits predict the carry into each group. The low-order bit position of any group receiving a group carry always receives a bit carry. In order to check the lookahead circuits, the adder predicts this same carry dependent on the bit functions of, and the carry into, the next lower-order bit. The carry from lookahead and the predicted carry (KC) are compared; if they are unequal an error has occurred.

On Figure 9055, the lookahead check for the carry into bit 51 (low-order bit of group 0, section 3) is added in dotted lines. Note the carry to bit 51 depends on group and section lookahead circuits. The predicted carry for the same bit position (KG_{48-51}) is generated using the carry into bit 52 (CB_{52}) and the bit functions for bit 52 (TB_{52} and DB_{52}). If the predicted carry is not the same as the carry from lookahead the byte error indicator is set. The lookahead check circuits for byte 08 to 15 are on Systems AM 163.

Shifting and Logical Connectives

As shown on Figure 63, shift operations and the performance of the logical connectives use data paths that differ from those of the add operation. Figure 9058 shows how the same bit functions used on the add operation are renamed and moved along different data paths to accomplish operations other than add. The logical exclusive OR is the same function as the half-sum. The logical AND is the same function as the generate.

Either or both of these functions are gated to the shifter. The exclusive OR gate to the shifter is normally conditioned and is used to deliver the single operand to the shifter on shift operations or to deliver the exclusive OR of the two input operands to the shifter when the logical connective exclusive OR is being executed.

To execute the logical connective AND the line "sel log AND" gates the AND function to the shifter and also degates the exclusive OR function so that it does not reach the shifter.

To execute the logical connective OR, both the exclusive OR and the AND functions are gated to the shifter. The "sel log AND" line is conditioned and the "sel log ORE" line is allowed to remain conditioned. These various gatings are represented logically on Figures 63 and 9058.

When the data path is to be through the shifter the output of the shifter is delivered to the AMOB latches, and the final sum from the main adder is blocked from reaching the AMOB latches. This gating is presented on Figures 63 and 9058.

Data Shifting: The relationship of input bits to output bits of the shifter for the various combinations of control lines that may occur are shown on Figure 9059. The control scheme used and the operation of the shift control lines (R32, R8, R4, L4, and L8) the "inhibit bits 28-35" lines and the "inhibit bits 60-63" are shown on Figure 5028. The operation of the "save sign," "propagate sign," and "propogate 16" control lines are shown on Figure 5029. Figure 5030 shows the operation of the "overflow" control line.

Shifter Parity Generation: Parity generation in the shifter depends upon the fact that when a single operand is gated to the adder, the half sum of the operand is equal to the operand. On the logical connectives AND and OR, when two operands use the shifter path correct parity is not generated.

Parity generation for the shifter output takes place in four stages:

1. The adder delivers to the shifter the half sum parity of each four-bit group and of the three-bit left extension (bits 64-66).
2. The shifter combines the half sum group parities for all combinations of groups that may form output bytes. This is done by the byte parity regeneration circuits on Figures 5031 and 5032. The circuits on Figure 5032 also take into account the affect of the save sign control on the bytes that may become bytes zero to eight.
3. The parity shifting circuits, Figure 5033 select the properly combined group parities for each output byte depending upon the shift being executed.
4. When any one of the four controls (prop signs, prop 16, overflow, or inhibit bits 28 to 35) that may force output bits is active the parity shifter circuits adjust the parity accordingly.

Shifter Overflow Detector: When fixed-point numbers are shifted left, it is possible to shift out high-order bits. This condition is detected by the shifter overflow detector; the simplified circuit is shown on Figure 5034.

The "save sign" line active indicates a fixed-point number is being shifted. With the save sign line active and a left shift taking place, all bits being shifted off are compared to input bit 0. If any bit to be shifted off differs from bit 0, the overflow condition is indicated.

Logical Connectives: The adder-shifter's part in the execution of the logical connective instruction is summarized with reference to Figure 9058.

Two cycles are required for the operation. On the first cycle:

1. Both operands are gated to the left end of the adder and parity is forced to the right end.
2. One or both of the bit functions (logic ORE or logic AND) are gated to the shifter depending on the connective to be performed.
3. The shifter is controlled to shift right four. Since the operands are 32 bits long no bits are shifted out.
4. The right four shift control also serves as a gate for gating the shifter to the main adder latch and to disable the gate used by the final sum to reach the main adder latch.
5. Both operands are checked for correct input parity but correct parity has not been generated on the output (for the AND and OR instructions).
6. The output is delivered to the K register.

On the second cycle:

1. The K register having (on AND and OR instructions) incorrect parity is gated to the adder.
2. The logic exclusive OR function is gated to the shifter. This delivers the operand to the shifter as it was received.
3. The left four shift control causes the connective to be delivered to the high-order positions of the main adder latch.
4. The connective now with generated parity is gated to the K register.
5. The left end of the K register is stored in general register specified by R1.

Carry Select Adder

The main adder high-order byte, positions 0-7, is implemented as a carry select adder. Basically, this differs from the seven low-order bytes in the resolution of the sums. The other adder functions--input OR's, bit functions, half sums, sum parity, and checking--are the same.

Sum Generation: Each sum is a function (exclusive-OR) of the associated half-sum and bit carry. The bit carries, in the lower order bytes, are functions of the group carry and the bit functions. The group carry has two states: "1" (carry) or "0" (no carry). For positions 0-7, two sets of bit carries are generated. One set, assume carry (AC), assumes the group carry is in the "1" state; the other set, not assume carry (NAC), assumes the group carry is in the "0" state.

The bit carry sets are then combined with the half-sums, producing two sum sets. One sum set is associated with a group of carry of "1" and is selected as the result sum if the look-ahead group carry result is a "1". The other set, associated with a group carry of "0", is selected if the look-ahead group carry is "0". The carry and sum sets are generated with positions 0-7 united as one group rather than two, as is the case in the lower-order bytes. The entire eight position sum is selected dependent upon the carry into group 4-7 (CG_{4-7}).

Figure 6008 presents a block diagram of the general components of the main adder related to the add function. Figure 6009 is a block diagram of positions 0-7, showing the sum generation path. Comparison of the two shows the differences as previously cited.

Adder Outputs

Figure 6010 is an overall figure of the outputs available from the adder.

CLOCK

Controlled Clock, Running Clock

The source of the machine pulses is an oscillator

whose output is fed through a frequency divider: the divider output is a series of 50-nanosecond pulses that occur at an approximate 200-nanosecond rate. These pulses are gated by the control clock trigger to generate the controlled pulses, and are gated by the running clock trigger to generate the running pulses (Figure 5035).

Various machine signals, through synchronizing triggers, turn the gating triggers on and off, thereby starting and stopping pulses to the central processing unit. The sync triggers synchronize the machine start and stop signals to oscillator time; this ensures that the gating triggers are turned on and off properly relative to the oscillator.

Controlled pulses are started:

1. By the start pushbutton when not in single-cycle mode.
2. At the end of any central processing unit or system reset.
3. At the beginning of the advance portion of a fault location test.
4. By the start pushbutton when in single-cycle mode. In this case a single controlled pulse is emitted by the clock.

Controlled pulses are stopped:

1. By the start pushbutton when in single-cycle mode (one controlled pulse is emitted).
2. By the rate switch being turned to the single-cycle position.
3. At the end of the advance portion of a fault location test.
4. At the beginning of any central processing unit or system reset.
5. By a machine check.

The running pulses are stopped by any system reset, but they are started again when the reset is over.

COUNTERS AND POINTERS

Digit Buffer-Digit Counter

- Digit buffer is a four-position register.
- Digit counter is a four-position register-latch combination with increment/decrement circuits.
- Digit counter steps up or down by one.
- Combined, digit buffer-digit counter contains one data byte with parity.

In System 360 Model 75, the smallest addressable unit of data is the byte which is eight data bits plus one parity bit. However, decimal and certain logical SS instructions manipulate decimal or hexadecimal data one digit, four data bits, at a time. The digit buffer (DB) and digit counter (DC) each provide a data path and temporary storage for a four-bit digit.

Digit Buffer

The digit buffer (Figure 5036) is a four-bit register; the bit positions are numbered zero through three, with position three the low-order position. The digit buffer is used to retain zone bits of the fill-character when executing an Edit instruction, and as temporary storage for the high-order digit of a quotient byte during the execution of the decimal divide instruction. When executing certain other instructions, the digit buffer and the digit counter are combined and used as one unit.

Data inputs to the digit buffer are from the left byte gate, from the AND-OR-Exclusive OR, and from the digit counter. Inputs from the digit counter are used during the execution of the decimal divide instruction to transfer the complemented quotient digit from the digit counter to the digit buffer. Other data inputs are used when the digit buffer and digit counter are combined.

Digit Counter

The digit counter (Figure 5037) is a four-position register-latch combination. The digit counter is used as a temporary storage for a four-bit data digit or as a counter. Increment and decrement circuits and control gates enable the digit counter to step up or down.

The digit counter is used as a counter in decimal multiply and decimal divide, and as a temporary storage register in edit, edit and mark, pack, unpack, and move with offset instructions.

Data inputs to each position of the digit buffer and digit counter are shown in the following listing and in Figure 5036.

	DB				DC			
	0	1	2	3	8	4	2	1
Multiplier Bus					0	1	2	3
AV Latch					4	5	6	7
LBG	0	1	2	3	4	5	6	7
AOE	4	5	6	7	0	1	2	3
DC	8	4	2	1				
Force 9 to DC					1	X	X	1

Selection of data set into the digit buffer-digit counter is controlled by input gates activated by the instruction being executed. The following text provides a general description of input control gates to the digit buffer-digit counter. For details concerning specific data or timing refer to the Theory of Operation Section and the instruction involved.

Gate Multiplier Bus to DC: This control gate is used during the decimal multiply instruction to set the multiplicand digit from the J register into the digit counter.

The first, low-order, multiplicand digit is set into the digit counter from J 60-63 during PF4 time of the set-up sequence. The J register is shifted right four to place the next multiplicand digit into positions 60-63. Thereafter, each time the multiplier is added to the partial product the digit counter is stepped up or down one. When the digit counter steps to zero when stepping down or to ten when stepping up, the multiplier has been added to or subtracted from the partial product the number of times specified by the multiplicand digit. A new multiplicand digit is set into the digit counter and the process is repeated until all multiplicand digits are processed. Each time a new multiplicand digit is set into the digit counter, the gate multiplier bus to digit counter line is active.

Gate AOE to DC-DB: The active status of the gate AOE to DC-DB line gates the data byte from the AND-OR-Exclusive OR output into the digit buffer and the digit counter. The low-order digit of the byte (AOE 4-7) is set into the digit buffer, and the high-order digit (AOE 0-3) is set into the digit counter.

During the execution of the unpack and move instructions, the high-order digit of an Op2 data byte becomes the low-order digit of the succeeding Op 1 data byte. To accomplish this, the data byte from the right byte gate is gated through the AND-OR-exclusive OR into the digit buffer-digit counter, the high-order digit goes into the digit counter. During the next cycle, the digit counter is gated into the low-order input of the decimal adder; the digit in the digit buffer is ignored. Thus, the high-order digit of one byte becomes the low-order digit of the next data byte.

Gate LBG to DB-DC: When the gate left byte gate to digit buffer-digit counter line is active, the data byte from the left byte gate is set into the digit buffer-digit counter. This gating is used during the ED and EDMK instructions to set the fill-character from the K register into the digit buffer-digit counter. Gate left byte gate to digit buffer-digit counter is also used when executing fixed point and floating-point divide instructions to set the high-order eight bits of the normalized divisor into the digit buffer-digit counter for comparison to the dividend.

Gate AV to DB-DC: During execution of the pack instruction, the low-order digit of two unpacked data bytes are combined to form one packed byte. The digit counter is used as a temporary storage for the low-order digit of each packed byte. The gate decimal adder to digit buffer-digit counter line is active during each IS 2 cycle of the pack instruction and sets the low-order decimal adder output digit into the digit counter.

Force 9 to DC: The active status of the force nine to digit counter line sets the digit counter to nine. During the decimal divide instruction, the digit counter is counted up from zero or down from nine to develop the quotient digit.

The digit counter is set to nine and counted down during cycles in which the divisor is added to a negative dividend.

Gate DC to DB: The active status of the gate digit counter to digit buffer line transfers the contents of the digit counter to the digit buffer. During the odd cycles of the decimal divide instruction the high-order digit of the quotient byte is developed in the digit counter. When the quotient digit is complete, it is transferred to the digit buffer. The low-order quotient digit is developed in the digit counter to complete the quotient byte.

DC Stepping: The digit counter can be reset, set, counted up or counted down. Figure 5037 shows the digit counter circuits in simplified logic. The increment/decrement circuits and register latch logic of the digit counter are essentially the same as those of the Y-Z counters. See Y-Z counter stepping for a detailed description of counter stepping. The count up and count down controls to the digit counter are controlled by the decimal multiply and divide instructions, the only instructions in which the digit counter is stepped.

DB Parity: A one-bit digit buffer parity register (Figure 5038) provides a correct parity for the data byte when it is gated from the digit buffer-digit counter to the K register. The digit buffer parity register is set to the input parity when a data byte is set into the digit buffer-digit counter, such as, from the decimal adder, or from the AND-OR-exclusive OR. The four-bit multiplier bus has no parity bit; however, because the multiplier digit set into the digit counter during decimal multiply is used as a count factor, parity is not necessary.

In decimal divide, the quotient byte that is transferred from the digit buffer-digit counter to the K register is developed in the digit counter one digit at a time. The high-order digit of the quotient byte is developed in the digit counter then transferred to the digit buffer. The low-order digit of the quotient byte is then developed in the digit counter to complete the quotient byte. When the quotient byte is complete, it is transferred to the K register with correct parity.

Development of a new quotient byte starts with the digit buffer-digit counter set to zero or nine and the parity register set to one. All digit buffer positions are reset and the digit buffer parity is set to one by the active status of the reset digit buffer control. At the same time the digit counter is released and is set to the status of the digit counter inputs, either zero or nine, depending upon the status of the force nine to digit counter control. Thereafter, the high-order digit of the quotient byte is developed by stepping the digit counter up or down. In each cycle that the digit counter steps, its bit configuration changes. When the number of data bits in the digit counter changes from odd to even or visa versa, the digit buffer parity bit is changed. Correct digit buffer-digit counter parity is thus maintained during each cycle that the digit counter is stepped. When the high-order digit of the quotient byte is complete, it is transferred to the digit buffer with correct parity. The digit counter is then restored to zero or nine and the low-order digit of the quotient byte is developed in the same manner. The digit buffer parity register continues to change status as necessary to maintain correct parity for the quotient byte. When the low-order digit of the quotient byte is complete, it is transferred to the K register with correct parity.

When the digit counter counts up or down, a parity correction latch (Figure 5038) determines the status of the digit buffer parity register. In each count cycle, the parity correction latch and associated circuits monitor the digit counter 1, 2, and 4 outputs and the status of the digit buffer parity register to determine correct parity for the next count cycle.

When the high-order digit of the quotient byte is developed by stepping the digit counter up from zero (Figure 9060), the digit buffer parity register indicates the correct parity of the digit counter during each cycle (Figure 9060, Section A). After the high-order digit of the quotient byte is transferred to the digit buffer and the digit counter is stepped up to develop the low-order digit of the quotient byte, then the digit buffer parity register indicates the correct parity for both digits. If the high-order digit of the quotient byte contains an even number of bits, excluding parity, then the digit buffer parity register is set or reset each count up cycle as shown in Figure 9060, Section A. If the high-order digit of the quotient byte contains an odd number of bits, then as the low-order digit is developed, the digit buffer parity register is set or reset each count up cycle as shown in Figure 9060, Section B.

Figure 9061 shows the same conditions when the digit counter is counted down to develop the quotient digits.

S and T Pointers

- S and T pointers set to starting byte address of operands (H21-23).
- S and T count up or down.
- S and T select operand bytes during processing.
- T pointer selects result byte location in the K register.
- S and T signal word boundaries.

Execution of variable field length instructions, in general, process one data byte at a time from each of the two operands. Both operands are brought from storage into the K and L registers. All bit positions of the K register are inputs to the left byte gate and the right byte gate; all bit positions of the L register are inputs to the right byte gate. Selected data bytes of Op 1 in the K register and of Op 2 in the L register are then gated through the left byte gate and the right byte gate to the variable field length circuits. Because only one data byte at a time is gated through the left byte gate or the right byte gate, the S and T pointers are used to select one of the eight bytes contained in the K or L register. The S pointer selects the byte gated through the right byte gate and the T pointer selects the byte gated through the left byte gate. See Figure 2040.

S and T pointers are, in general, set to the starting byte address of each operand. They are set during the set-up sequence of an SS instruction or during the fixed-point cycle of fixed-sequence variable field length instructions. For SS insturctions, T is set to Op 1 starting byte address, and S is set to Op 2 starting byte address. H register bit positions 21, 22, and 23 are the only data inputs to the S and T pointers. As each operand fetch is initiated, during variable field length set-up sequences, the storage address is computed and set into the storage address register and the H register. H register positions 21-23 are then set into the S or T pointer. The usual sequence is to fetch the first Op 1 word from storage and set the byte address into the T pointer, then fetch the first Op 2 word from storage and set the byte address into the S pointer.

Thereafter, as each data byte is processed during iteration cycles, S and T pointers are stepped up or down and the decoded output is used to gate the next operand byte to be processed. Whether S and T step up or down is determined by the direction the instruction moves through the operands.

Figure 5039 and Figure 5040 are simplified positive logic diagrams of the S and T pointers. Each pointer is a three-position register-latch combination with increment/decrement circuits that enable the pointers to step up or down.

Each S or T register is a polarity hold (PH) circuit with a data input, a release control and a reset control. When the release control is active, the register sets to the status of the data input each central processing unit A clock time. At all other times during the central processing unit cycle, the S and T registers are locked. When locked, the registers retain their status independent of fluctuations that may occur on the data input lines.

Each S or T latch is a polarity hold circuit similar to the register; however, the timing of the release to the latch differs from that of the register. Each S or T latch position sets to the status of the data input and is locked during central processing unit L clock time. Except at L clock time, the latch output follows the data input.

Data input to each S or T register is from the latched output of that position or from the increment/decrement circuit. Data input to each S or T latch position is from the H register or from the register output of the same position. Selection of data gated into the registers and latches is determined by the status of the S and T function controls.

Function Controls

Functional controls gated to S and T pointers throughout the execution of variable field length instructions enable the pointers to set, reset, step up, or step down, or remain static and retain the present count. Because data bytes are not processed during every central processing unit cycle of every variable field length instruction, S and T controls vary from cycle to cycle. The function gated to the S and T pointer is controlled and timed by the instruction in process.

Gate H21-23 to T Lth: The active status of the gate H21-23 to T latch line (Figure 5040) gates H register bits 21-23 to the T latch; in general, the starting byte address of Op 1. Early in the set-up sequence of each SS instruction, fetches are initiated for the first words of Op 1 and Op 2. The storage address of the first Op 1 word is computed and set into the storage address register and the H register. Gate H21-23 to T latch is then active during SU T3 time to gate the starting byte address of Op 1 to the T pointer. The L clock pulse at the end of SU 3 time locks the T latches to the status of the bit lines from the H register, and the A clock at the beginning of SU 4 sets the T registers to the output of the T latches.

Gate H21-23 to T latch is also active at other times when the starting byte of Op 1 is referenced, such as, at the beginning of the change sign pass of the decimal add or subtract instruction, or to provide correct multiplier/partial-product or divisor/dividend alignment during MP or DP instructions.

Gate T Reg to T Lth: The active status of the gate T register to T latch line gates the contents of the T register to the T latch. Because it is the inverted function of gate H21-23 to T latch, gate T register to T latch is active when the Gate H21-23 to T latch is inactive, except SU L9 time of the MP or DP set-up sequence. During SU L9 of the MP or DP set-up sequence, gate T register to T latch is blocked to force the T pointer to zero preparatory to being stepped down to seven.

During a variable field length execution cycle in which a data byte is processed, the decoded output of the T register gates the data byte out of the K register through the left byte gate. The result byte is set into the same byte position in the K register at the beginning of the next cycle. Therefore, the register value is gated and set into the T latch by the gate T register to T latch control; the output of the T latches are then decoded to select the input byte of the K register.

Gate H21-23 to S Lth: The active status of the gate H21-23 to S latch control (Figure 5039) sets the byte address contained in H register positions 21-23 into the S latch. In general, this is the starting byte address of Op 2. Early in the set-up sequence of SS instructions, the storage address of the word that contains the first Op 2 byte is computed and set into the storage address register and the H register. Gate H21-23 to S latch is active during SU T5 and gates the starting byte address of Op 2 into the S pointer. The L clock that follows locks the S latches and the A clock sets the S register to the S latch.

Gate H21-23 to S latch is also active during the first fixed point cycle of fixed sequence variable field length instructions and during PF L1 of TR or TRT instructions.

Gate S Reg to S Lth: The active status of the gate S register to S latch line gates the contents of the S register to the S latch. Gate S register to S latch is active when the gate H21-23 to S latch is inactive except during PF 4 of MP or DP set-up when the S pointer is forced to zero.

Release S and T: Release controls to the S and T registers and latches enable the pointers to set and to step up and down. Release timing to the latches differs from that to the registers.

Release to S and T registers is a gated central processing unit A clock pulse. Gating of the A clock release to the registers is controlled by the instruction in process. In general, however, a release is gated to S or T register when the pointer is set or stepped.

Release to the S or T latch is an ungated L clock pulse. S or T latches are released each central processing unit cycle and set to the register or to the bit position of the H register when the gate H21-23 to S latch or gate H21-23 to T latch is active. Release to the T latch is blocked during PF 1 and PF 2 cycles of TR or TRT instructions by the active status of the hold T latch control (see S and T Stepping).

Reset: Reset controls to the S and T pointer enable the pointers to be reset to zero. Reset control to both pointers is active whenever a central processing unit reset is initiated or when the first fixed-point latch is set to execute CVB or CVD instructions.

When executing first fixed-point or first floating-point divide instructions, the T pointer is reset to zero to gate the high-order divisor byte from the K register through the left byte gate into the digit buffer-digit counter.

A reset is gated to the S pointer during Seq A cycle of MP or DP instructions. At this time, the S pointer is reset to zero then counted down 1 to 7 to start a new pass through the multiplier or divisor.

S and T Stepping

Execution of SS instructions process data bytes one at a time and serially from each operand. Logical SS instructions start with the high-order (most significant) data byte and move through each operand one byte at a time to the low-order byte. Decimal SS instructions start with the low-order data byte and move through each operand in the opposite direction except for MP and DP instructions.

The S and T pointers are used to select and gate the data bytes of each operand through the variable field length circuits; S selects the data bytes gated through the right byte gate, and T selects the data bytes gated through the left byte gate. As each successive data byte is processed, the S and T pointers step up or down. The stepping function of each pointer is controlled by the active status of the count up or count down line (Figure 5039 or Figure 5040).

Increment/decrement circuits associated with each position of the S or T pointer enables the pointer to step up or down. The increment/decrement circuits are inserted between the latch and register of each S or T pointer position. When stepping, the register is set to the incremented or decremented value of the latch, depending upon which count control is active. The latch then sets the new register value at the end of L clock time.

Count S and T Up: The active status of the count S and T up control causes the S and T pointer to step up one each central processing unit cycle. Through the increment/decrement circuits of each pointer position the status of the count up control and the status of the latch gates a set or reset to the register and propagates a carry to the next higher-order position.

Consider the T pointer (Figure 5040) to be gated to count up from zero. Figure DM9062 shows 9 central processing unit cycles in which the pointer is counted up from zero to seven and the register-latch relationships during each cycle. Assume central processing unit cycle 1 to be the first cycle that the count S and T up control is active, and that the T pointer has previously been set or reset to zero.

Through the increment/decrement circuits of T1 position (Figure 5040), the active status for the count S and T up control and the reset status of T1 latch gates the set of T1 register. The L clock pulse at the end of central processing unit cycle 1 and the start of central processing unit cycle 2 (Figure 9062) locks the T1 latch reset. The A clock pulse at the beginning of central processing unit cycle 2 then releases and sets the T1 register.

Because the Gate T register to T latch control is active, the T1 latch sets to the status of the T1 register at the end of L clock time during central processing unit cycle 2. Thus, the T pointer steps from 0 to 1.

As soon as the T1 latch sets, during central processing unit cycle 2, the T1 increment/decrement circuits gates a carry into T2 and blocks the set of T1 register. The T1 register remains set for the remainder of the cycle; however, because it is released only at A clock time.

Through the T2 increment/decrement circuits, the active status of the count S and T up, carry into T2, and the reset status of T2 latch gates the set of T2 register. The following A clock at the beginning of central processing unit cycle 3 sets the T2 register. Because the set of T1 register is blocked, it resets.

At the end of L clock time, the T latches set to the status of the registers. In this case, during

central processing unit cycle 3, T1 register and latch are reset and T2 register and latch are set. The T pointer has stepped from 1 to 2.

During central processing unit cycle 3, the reset status of T1 latch, through the T1 increment/decrement circuits, gates the set of T1 register. Because the carry in to T2 is active only during the time T1 latch is set, T2 register is gated to set to the status of T2 latch. The T latches are locked at the end of central processing unit cycle 3; T1 latch is locked reset and T2 latch is locked set. The A clock sets T1 and T2 registers.

At the beginning of central processing unit cycle 4, the T registers are set to the value of 3, binary sum of 2 and 1. At the end of L clock time, the T latches are set to the register value. The T pointer has stepped from 2 to 3.

During central processing unit cycle 4, when the T1 latch sets, carry in to T2 is active. Because the T2 latch is set, the carry in to T2 is propagated through T2 increment/decrement circuits and becomes a carry in to T4. The reset status of T4 latch gates the set of T4 register, while the set status of T1 and T2 latches block the set of T1 and T2 registers. The A clock, at the start of central processing unit cycle 5 sets T4 register, and because the set of T1 and T2 registers is blocked, they are reset.

At the beginning of central processing unit cycle 5 (Figure 9062), the T register steps from 3 to 4. At the end of L clock time, the T latches set to the registers and through the increment/decrement circuits establish gates for the next incremented value.

Because the pointer steps binary-wise, stepping up from 4 through 7 is a repetition of the logic of stepping 0 through 3, except that the T4 register and latch remain set until a carry is received from the T2 position. When the pointer steps beyond 7, the count reverts to zero and starts over.

The preceding text and figures have described the stepping up of the T pointer. The count up controls and functions of the S pointer (Figure 5039) perform in the same manner.

Count Down: The count S down or count T down controls enable the S or T pointer to step down. Each central processing unit cycle that the count down control is active the pointer steps down one. Because certain variable field length instructions, such as pack or unpack, process data bytes of each operand at a different rate, the step down controls of S and T pointers are independent.

When executing decimal instructions, except MP or DP, the data byte that contains the low-order decimal digit is the first to be processed. S and T pointers are set to the starting byte address of each operand, then stepped down one as each data byte is processed. Because data bytes are not processed every central processing unit cycle of instruction execution, the count S down and/or count T down are active only during the cycles in which S and T pointers are to step.

The stepping logic of the S and T pointers are identical; therefore, the following count down description of the T pointer also applies to the S pointer.

Each cycle that the T pointer (Figure 5038) is stepped down, one is subtracted from the pointer value. Subtraction is accomplished in the pointers by subtracting one from the units position, T1, and inverting the function of the carry in to T2 carry in to T4 to that of borrows from those positions. When the T pointer is stepped up, one is added to the T1 position each count cycle and the carries from T1 are accumulated in T2 and T4 positions. When T is stepped down, one is subtracted from T1 position each count cycle and repeated borrows from T2 and T4 reduce the pointer value.

In binary subtraction, 1 - 1 = 0 and 0 - 1 = 1 with a borrow from the next higher-order non-zero bit position. This principle is used when S and T pointers are stepped down. Through the increment/decrement circuits (Figure 5040) of T1 and T2 positions, the carry in to T2 and Carry into T4 become active to effect borrows from the T2 and T4 positions. As shown in Figure 9063 for each count-down cycle one is subtracted from T1 position. The T1 register and latch change status every count cycle, either from set (1) to reset (0) or from reset (0) to set (1). During the alternate count cycles in which T1 position is reset (0) a borrow from T2 position occurs (Carry in to T2 is active). During count cycles in which both T1 and T2 positions are reset (0) a borrow from T4 occurs (Carry in to T4 is active). The active status of the Carry in to T2 or the Carry in to T4 causes those positions to change status.

Figure 9063 shows nine consecutive central processing unit cycles in which the S or T pointers are stepped down from six through zero to six. Although shown this way to display the register-latch relationships throughout the stepping range of S or T, the pointers are never stepped for more than eight consecutive cycles. Because the decoded output of S and T are used to gate data bytes from each operand and to signal the approach of a word boundary, no more than eight data bytes are processed between word boundaries. When a word boundary is encountered, S or T pointers equals zero when counting down or equals seven when counting up, stepping of the S and T pointer is suspended until a new word is set into the K register or the L register and byte processing resumes. The stepping of either pointer may be suspended during any cycle by making the

count controls inactive. When the count controls are inactive, the pointer retains its value until a new value is set into it or the pointer is stepped or reset.

Hold T Lth: The Hold T latch control is used when executing the TR instruction and both operands are contained in the same storage word--operands overlapped.

Overlapping the table word return with the address calculation for the next fetch (the next byte to be translated) requires two byte addresses. One is the byte address where the translated byte is to be temporarily stored in the K register; the other byte address is that of the next byte to be translated. A hold is used on the T pointer latch to prevent it from changing after the T pointer is advanced. This holds the translated byte address in the T latch to control T IN decoding to the K register, while the T register advances and gates the next byte out of K through the left byte gate to the addressing adder. Figure 9064 shows a block diagram of the T pointer and an example timing chart to show the above conditions.

Y and Z Counter

- They are four-position binary counters.
- Y and Z contain Op 1 and Op 2 lengths for decimal instructions.
- Y and Z are coupled as an eight-position binary counter for SS logical instructions.
- Y and Z are used as a four-position counter; Y or Z steps up or down by one.
- Coupled as an eight position counter; YZ steps up or down by one or by eight.
- Y and Z provide address factors to compute storage addresses for operand fetches.
- Y and Z determine when execution of SS instructions is complete.

The Y and Z counters of the System 360 Model 75 are each four-position binary counters. They are used as operand length counters for the SS instructions, and as registers to retain the immediate data field of certain SI instructions. When used as counters they may be counted up or down.

In general, during the execution of SS decimal instructions, Y and Z are used as two independent counters with a capacity of counting 16 bytes each. When the execution of a decimal instruction starts, Y counter contains the byte length of Op 1 (L_1) and Z counter contains Op 2 byte length (L_2). Y and Z are then counted down to zero, all data bytes of both operands have been processed and the instruction execution ends.

During the execution of SS Logical instructions, Y and Z counters are coupled and used as one eight-position counter with a maximum count capacity of 256 bytes. For the majority of SS Logical Instructions, Y and Z counters are set to the L field of the instruction (IOP 8 - 15) and counted down one as each data byte is processed. The instruction is terminated when YZ steps down to zero.

When coupled as one eight-position counter, YZ is stepped up or down by one or by eight. Y and Z are counted down by eight when executing the MVC instruction in transmit mode.

Y and Z are coupled as an eight-position counter and stepped from zero up during the execution of TR, TRT, ED, and EDMK instructions; when equal to the specified operand length (L), the instruction is terminated.

In decimal divide, the number of quotient bytes to be generated is $L_1 - L_2$. L_2 is set into Y and Y is counted up until it equals L_1 (IOP 8 - 11).

The following table shows the starting and ending conditions of the Y and Z counters for all SS instructions:

Instructions	Start Y	Start Z	Count Y	Count Z	Count Y and Z	End Op
AP, CP, SP, ZAP	L_1	L_2	Dn	Dn		Y & Z = 0
MP	L_1	L_2	Dn	Dn		Y = 0
DP	L_2	L_2	Up	Dn		Y = L_1
MVO, PK, UNPK	L_1	L_2	Dn	Dn		Y = 0
MVN, MVC, MVZ, CLC, NC, OC, XC	L				Dn	YZ = 0
TR, ED, EDMK	0				Up	YZ = L
TRT	0				Up	YZ = L or Nonzero Character

In addition to performing the functions of operand length counters, Y and Z provide factors used, in certain SS instructions, to compute storage addresses when an operand store or fetch operation is started. For example, the decimal add instruction starts with the storage word that contains the low-order (least significant) data byte, and processes each operand one byte at a time until the highest-order operand bytes are processed. Y and Z counters initially contain the length of each operand, Y contains L_1 and Z contains L_2. Because either decimal operand may be contained in as many as three storage words, and because the word containing the low-order decimal digit is the first to be processed, the contents of Y and Z

counters are used to compute the storage addresses needed to fetch the first word of each operand. Y is gated to the addressing adder to compute $B_1 + D_1 + L_1$ (Y) storage address of the first Op 1 word; Z is gated to the addressing adder to compute $B_2 + D_2 + L_2$ (Z) storage address of the first Op 2 word.

During the execution of the EDMK or TRT instructions, the mark sequence places the storage address of a selected data byte into a general purpose register. When executing these two instructions, Y and Z counters are coupled and used as one eight-position counter and counted up. YZ counter provides a convenient method of computing the storage address of the selected data byte.

Fixed sequence variable field length instructions in the S1 instruction format use the Y and Z counters as immediate data registers. During the execution of these instructions, the Y and Z registers are gated to the AND-OR-Exclusive OR where the connective function with a data byte from storage is performed.

Y-Z Counter Logic

The four positions of the Y or Z counters are identified 1, 2, 4, and 8. When used as counters, they step up or down binary-wise. The numeric value of either counter at any specific time is represented by the binary sum of the four positions; when Y and Z are coupled as one counter, the value is represented by the binary sum of all eight positions, with Z1 and low-order position.

Each position of either counter consists of a trigger register and latch. Increment/decrement circuits are inserted between the register and latch to enable the counters to step up or down and to propagate a carry from one position to the next. The circuit arrangement of the trigger and latch is that of the conventional System/360 polarity hold (PH) retention circuit. With but one exception, the Y and Z triggers (registers) change status, either set or reset, at central processing unit A clock time. The Y or Z latches may change status any time during a counting cycle except at L clock time when they are locked.

Except for control gating and coupling, the Y and Z counter operations are identical; therefore, description of counter circuits apply to either counter.

Figure 5041 shows the circuits, in simplified positive logic, of the Z register, latches, and increment/decrement circuits. Z register (trigger) positions 1-8 are set to instruction operation register bits 12-15 or to the Z latches, depending upon the active status of the Gate IOP 12-15 to Z or the Gate Z Lth to Z Reg line.

Y-Z Controls

The count up, count down, and release controls to the Y and Z counters are, in general, activated by the instruction being executed.

Gate IOP: The Gate IOP 8-11 to Y and Gate IOP 12-15 to Z lines are active during E last cycle (ELC) of every instruction and during central processing unit cycles that the E unit is not busy. Gate IOP 8-11 to Y allows IOP register bits 8-11 to set into the Y register. For SS decimal instructions, IOP 8-11 is the L_1 operand length.

Gate IOP 12-15 to Z allows instruction operation register bits 12-15 to set into the Z register. For SS decimal instructions, IOP 12-15 is the L_2 operand length.

For SS logical instructions, Y and Z counters are coupled; instruction operation register bits eight through fifteen then represent Op 1 field length.

When single-byte type RR, RX, and SI instructions are executed, the set of instruction operation 8-15 (IOP 8-15) to Y and Z counters during the E last cycle of the previous instruction enables the I unit and the E unit cycles to overlap.

Gate Latch to Register: The gate Y latch to Y register and the gate Z latch to Z register lines are the inverted functions of the gate IOP 8-11 to Y and the gate IOP 12-15 to Z lines. These lines are active except when instruction operation is gated to Y and Z.

Gate Y Lth to Y Reg or Gate Z Lth to Z Reg enables the setting of the Y and Z registers to the incremented or decremented value of the latches when the counters are stepping or when data is retained.

Count Up/Down: The active status of the count up or count down lines to the Y and Z counters determine whether the counter steps up or down. The instruction being executed determines which of the two lines is active. Y and Z counters are only stepped when an SS instruction is being executed; therefore, the Y and Z count controls are active only at this time.

Figure 5041 shows simplified logic of the count up and count down control of the Y and Z counters.

Release: Conventional System 360 Model 75 circuit logic of polarity hold triggers and latches use a release signal to set data into a trigger or latch register. When the release is active, the trigger or latch sets to the condition of the input, either on or off. When the release line is not active, the trigger or latch is locked to the status of the input at the

time the release became inactive. Thereafter, the trigger or latch retains the same status, impervious to changes to input status, until the release is again active.

Release to the Y-Z triggers and release to the latches are independent and controlled separately. In general, except the initial set during E last cycle, both are controlled by the instruction being executed.

When Y and Z triggers are set to instruction operation, during E last cycle, the release is a central processing unit B clock pulse; at other times, when Y and Z are stepped up or down and the trigger is set to the latch, the release is at central processing unit A clock time.

The latches function differently; when release is active, the latch output follows the input except at central processing unit L clock time, when the latch is locked to the status of its input at the beginning of the L clock pulse. This enables the use of the latch output to set or reset other triggers at central processing unit A clock time.

The Y-Z latches are set to the register value or to the status of the increment/decrement circuit depending upon the active status of the count lines (See Y-Z Stepping).

In general, when executing SS instructions, Y and Z counters are stepped during central processing unit cycles in which data bytes are processed. Therefore, the count up or count down controls are active during all cycles and release gated to the triggers and latches only during those cycles when counting occurs.

During SU 2, SU 4, SF 1, and SF 3 cycles, the contents of the Y or Z counter is used to compute storage addresses. During these cycles, a hold is activated on the Y and Z latch release and the release is blocked to prevent any deviation of line levels from the Y and Z latches to the addressing adder input.

Y-Z Stepping

Y and Z registers are set to the operand length field of instruction operation register positions 8-15 during E last cycle of every instruction. Thereafter, the registers are set to the output of the latches. When counting, the latches contain the incremented or decremented counter value, the value to be set into the register at the beginning of the next count cycle.

To enable each Y or Z counter position to count up or down, increment/decrement circuits are inserted between the register and latch. The count up or count down lines and the increment/decrement circuits control the set and reset of the latches such that the value in the counter latches is the next value set into the register. Figure 9065 is a timing example of the Y or Z counter stepping. Two examples are shown, that of step counter up and step counter down.

Sixteen central processing unit cycles are shown. The count up example shows the trigger-latch relationship as the count progresses from minimum value (0) to maximum value (15). When the counter is stepped beyond 15, it returns to zero and continues stepping. The count down example shows the counter stepped from a value of 14 down through zero to a value of 15.

Particularly note the relationship of the registers and latches. The counter value in the latches always precedes the register value when counting. For the majority of SS instructions, Y and Z counters are stepped down and the instruction terminated when Y and Z equal zero. When counting down, the counters are stepped with the set conditions for the iteration sequencers. Because the operand length is the specified number of bytes minus one, a counter value of all ones indicates the end of operation instead of a zero value (counter is stepped past zero to 15). The latched output of the counter is decoded instead of the register output because the decoded output is used to set and reset control triggers at central processing unit A clock time. Therefore, a counter value of 1110 for decimal instructions or a combined YZ counter value of 1111-1110 for logical instructions indicates all bytes have been processed.

Figure 9065 shows the counting up or down of the Y or Z counter for 16 consecutive central processing unit cycles. Sixteen count cycles are necessary to display the trigger-latch relationships for all combinations throughout the counting range of each counter. Because Y and Z are stepped one as each data byte is processed, no more than eight consecutive stepping cycles occurs before a word boundary is reached. When a word boundary in either operand is encountered, the stepping of Y and Z is temporarily suspended until a new operand word is set into the K or L register; then byte processing and Y and Z counter stepping is resumed. Because Y and Z stepping occurs only when a release is gated to both the trigger and the latch, stepping is suspended by allowing the release to Y or Z to remain active. When the release is inactive, the triggers and latches retain their value.

The increment/decrement circuits control the set or reset of the counter latches, and generate and propagate carries from one position to the next. For each counter position, the set or reset status of the register combined with the active status of the count up or count down controls determine whether the latch for that position is set or a carry is propagated to the next higher position.

Figure 5042 shows a simplified positive logic diagram of Z counter positions one and two with increment/decrement circuits for those positions. The heavy lines indicate the conditions to set the Z1 latch when the counter is counted up one from zero. This status corresponds to central processing unit cycle one of Figure 9065 when stepping the counter up. Because the count Z1 and Z2 up line is active (Figure 5042) and Z1 register is reset, the increment/decrement circuit gates the set of the Z1 latch. At L clock time, the Z1 latch is locked (release is inactive). The set status of the latch then conditions the input to Z1 register (heavy broken line). The Z1 register sets to the status of the latch at A clock time.

Figure 5043 shows the Z counter stepping up from one. The heavy lines are those active at the start of L clock time. The active status of the count up and the count down lines are OR'ed to provide a carry-in to the Z1 counter position. When counting up, the set status of the A1 register and a carry-in blocks the set of the Z1 latch and gates a carry to the Z2 counter position. In the increment/decrement circuit for Z2 counter position, the carry from Z1 combines with the reset status of Z2 register and sets Z2 latch. The set status of the Z2 latch enables the next A clock release to set the Z2 register. The same A clock release resets the Z1 register because the set of the Z1 latch is blocked. Thus, from the end of the L clock pulse at the beginning of central processing unit cycle two (Figure 9065) to the end of the A clock pulse at the beginning of central processing unit cycle three, the Z counter stepped from one to two.

At the end of L clock time during central processing unit cycle three, the Z1 latch is again set. The increment/decrement circuit for Z1 position (Figure 5043) gates the set of the Z1 latch because the count up line is active and Z1 register is reset; no carry is gated to position Z2. Because position Z2 receives no carry, the Z2 register and latch retain their status. Near the end of central processing unit cycle three (Figure 9065), the set status of Z1 and Z2 latches represent a counter value of three, set to the registers with the next A clock release.

The preceding text and figures have described the stepping of the Z counter up from zero to one to two and to three. Stepping from three to four, and so on, occurs in the same manner utilizing counter positions four and eight. Operation of the Y counter is exactly the same.

For each counter position during a count up cycle, if the register is reset and a carry-in is present, the latch is set; if the register is set and a carry-in occurs, the set of the latch is blocked and the carry is gated to the next higher-order counter position. In this manner carries generated on alternate count cycles, when the low-order (Y1 or Z1) register is set, are gated through each counter position progressively to the high-order (Y8 or Z8) position as the count value increases.

The count-down logic of Y and Z counters is similar to the count up except the generation and propagation of carries from the low-order to the high-order counter positions is reversed. When counting down, carries are generated by the low-order (Y1 or Z1) position on alternate count cycles when the register is reset (Figure 9065). A carry received by each counter position causes that position to change status and if the register for that position is set, the carry is gated to the next higher-order position. Thus a carry generated by the low-order position (Y1 or Z1) may be propagated through the increment/decrement circuit of each higher-order position to the Y8 or Z8 position if the registers of the intermediate positions are reset.

Figures 5044 and 5045 are the simplified positive logic diagrams of the Y and Z counters. The count up and count down logic for each counter is exactly the same. Differences do exist; however, in the counter controls that enable Y and Z to be coupled as one eight-position counter, and to step up or down by one or by eight.

Y-Z Counters Coupled

Y and Z counters are used as two independent four-position counters when SS decimal instructions are processed. When SS logical instructions are processed, Y and Z are coupled together and used as a single eight-position counter. When coupled as a single counter, Y is the four high-order positions and Z is the four low-order positions. Combined, Y and Z provide a counting capacity of 256 (0-255), and may be counted up or down by ones or by eight. The move (MVC) instruction is the only one that steps the YZ counter down by eight, and that occurs only when in transmit mode.

Special decoding circuits provide the correct controls to the Y and Z counters for each instruction executed. Figure 5041 shows, in simplified logic, the control decoding for the Y and Z counters. Of particular significance is the carry to Y1 and the split of the count up and count down lines to Z.

The carry to Y1 enables the Y1 counter position to change status each central processing unit cycle that a release is gated to the Y register and latch. When Y is used as an independent counter, the carry to Y1 is active throughout the execution time of the SS instruction and position Y1 changes status every count cycle.

Y is the high-order counter when Y and Z are coupled and counts only the carries from the Z counter. The carry to Y1 is active only during the

cycle that Z - 15 when counting up or Z = 0 when counting down. Because increment/decrement circuits of the Z counter do not provide a carry from the Z8 position, the value in the Z register is decoded to gate the carry to Y1 (Figure 5041).

When executing the move (MVC) instruction, Y and Z are coupled and counted down. If the instruction starts execution in transmit mode, YZ is stepped down by eight each count cycle. If the instruction does not start in transmit mode, then YZ is stepped down by one each count cycle. When YZ is gated to step by eight, the count Z4 down and count Z4 up lines are active. This causes Z4 increment/decrement circuits to gate a carry to Z8 every cycle, and Z8 changes status every cycle that YZ is released. When YZ counts down by eight, the carry to Y1 is active during the alternate stepping cycles that Z8 is reset.

DECODERS

BOP Decoder

- The BOP decoder is implemented in four levels of logic.
- Sequencing, branch, and interrupt signals are included in the decoder outputs.

The decoding is implemented in general, in four levels of logic. A negative AND-INVERT block generates plus outputs which include format decoding and basic instruction class decoding. This is followed by an AND-OR-invert to combine the class decoding into required functions in each of the formats. This output is inverted and another AND-OR-invert, combining the format and associated functions, produces the desired output. Included in the decoder outputs are sequencing, branch, and interrupt signals.

A list of the decoder functions and the instructions which generate these outputs are provided on Figure 9068.

BR1 Field Decoder

- The BOP register R1 field (BR1) contains bits 8-11 of the instruction.
- The decoding is implemented in two levels of logic.
- The BR1 field decoder selects a GPR or FLP register.

The B operation (BOP) register R1 field (BR1) contains bits 8-11 of the instruction. This field is used to select one of the general purpose registers or one of the floating-point registers for out-gating. A manual input is provided to force the decoder to any one of the 16 possible outputs. The select lines are AND'ed with a gate-out control in both of the stacks.

The manual controls include display lines for the general purpose registers and the floating-point registers and four encoded bits. The DSPLY GPR and DSPLY FPR lines are OR'ed and inhibit the BR1 field inputs. The line also gates the manual bits to the decoder.

The decoding is implemented in two levels of logic. The first AND's the low-order bit with the high order bit for both the BR1 and the manual inputs. The two are gated as mentioned and OR'ed. The middle two bits are treated similarly in the first level and produce immediate decoding for the FPR stack (select 0, 2, 4, or 6), since the outside two bits must be zero for floating-point register selection.

The second level of logic is a negative AND-invert to combine the four bits for general purpose register selection with a plus output.

Channel Decoder

- Decodes the channel for I/O instructions.

The channel decoder decodes the addressed channel from bits H13-15 for the Start I/O, Test I/O, Halt I/O, and Test Channel instructions. Outputs of the channel decoder are sent to the channel selector which sends a select line to the proper channel if it is present in the system. The channel decoder also detects illegal channel addresses to generate an early release from the central processing unit. See Figure 6354.

Divide Decoder

- The divide decoder selects the divisor multiple for the divide iteration.

The divide decoder (Figure 5046) serves to determine the proper divisor multiple to be selected for each divide iteration cycle. The main adder 0-7 is implemented as a carry select adder to provide early sums to compare with the divisor, generating the new multiple within the basic clock cycle.

Multiple Resolution

The high-order divisor bits, contained in the digit buffer-digit counter registers are compared with each of the sum sets according to the relationship required by the divide algorithm (Figure 9067). As the new dividend (adder result) is either true or complement, four functions are generated for each divisor multiple:

$$\text{result true} \cdot CG_{4\text{-}7}$$
$$\text{result true} \cdot \overline{CG_{4\text{-}7}}$$
$$\text{result complement} \cdot CG_{4\text{-}7}$$
$$\text{result complement} \cdot \overline{CG_{4\text{-}7}}$$

There are four multiple sets generated. The selection is made dependent upon the add result. If the add result is true (defined by the carry from position 0) and a carry to group 4-7 ($CG_{4\text{-}7}$) occurs, then the true · AC set is selected to the multiple latch. Similarly, the remaining combination of add result and $CG_{4\text{-}7}$ select the other multiple sets.

Divisor Leading Zeros

The divide execution sequencing does not bit normalize the operands. However, the divide decoder adjusts for leading zeros in the divisor by comparing different bit groupings dependent upon these leading zeros. Figure 9068 shows this relationship.

First Cycle Multiple Selection

Prior to the first iteration cycle, the dividend may be positioned in different locations. To determine the multiple for the first iteration cycle, the decoder is able to examine the dividend as it appears in these different locations. This is accomplished by OR'ing the possible first cycle dividend sources, under control of the execution hardware, prior to entering the comparison circuitry.

High-Order Zeros Detector

Positions 0-3 of the two sum sets (AC and NAC) are examined for all zeros and all ones conditions. The correct function is selected by the carry to group 4-7 (CG_{4-7}), producing the two functions:

AM_{0-3} equal zero

AM_{0-3} equal ones

These functions are used by the execution control circuitry in determining instruction sequence.

EOP Decoder

- The EOP decoder decodes the contents of the E operation (EOP) register.

The E operation decoder (EOP decoder) decodes the contents of the E operation (EOP) register. The lines produced by the E operation decoder are prefixed by ED. An example is ED MH (multiply half). The eight-bit operation code field is decoded by four-bit groups into their hexadecimal values, then the two hexadecimal values are combined to produce lines that denote a particular instruction or group of instructions. See Figures 5047 and 9069.

ER1 Field Decoder

- The ER1 decoder is analagous to the decoder for the BR1 field.

The ER1 field decoder is analagous to the decoder for the BR1 field. The decoding is done in two levels of logic which generate 16 select lines for the general purpose registers and four select lines for the floating-point registers. The four encoded bits from manual controls are also used to force the ER1 decoder to one of the 16 outputs. The control line which inhibits the ER1 field and gates the manual lines is, in this case, a store control rather than the display control.

IOP Decoder

- The I Decode section of CPU contains much of the logic for controlling I box operations during I time.

- IOP decoding, GPR out-gating controls, invalid op detection, and parity checking on bits 8-15 of IOP register are included in I decode.

The instruction operation (IOP) register is set at the start of each instruction and remains valid until the start of the next instruction. During this period, the instruction unit performs two basic functions--calculation of the effective address and procurement of operands for execution. The instruction operation decoder performs most of the instruction class decoding for these functions.

The decoding is implemented, in general, in four levels of logic. A negative AND-invert combines two instruction operation bits to obtain format decoding and basic instruction class decoding. A positive AND-OR-invert combines the class decoding.

A list of the decoder outputs and the instructions which generate the outputs are provided on Figure 9068.

LCOP Decoder

- The LCOP decoder provides decoding for the LCOP register.

The last cycle operation (LCOP) decoders decode the outputs of the last cycle operation register. The lines produced by the last cycle operation decoder are prefixed by LD. Example LD FLP M (floating-point multiply). The eight-bit operation code field is decoded by four-bit groups into their hexadecimal values, then the two hexadecimal values are combined to produce lines that denote a particular instruction or group of instructions. See Figures 5047 and 9068.

Multiply Decoder

- The multiply decoder decodes the multiple to be used for fixed-point multiply and floating-point multiply instructions.

- The multiply decoder inputs are J27-31 or J59-63.

- The multiply decoder outputs gates the K, L, or M registers to the main adder during each iteration cycle.

The multiply decoder is used by fixed-point multiply and floating-point multiply instructions to decode the multiple or multiples which are gated from the working registers to the main adder during each iteration cycle. Figure 5048 is the multiply decoder input gating, and a block diagram of the multiply decoder, the multiplier digit latches, multiply decode, and the decoder gates. The chart on Figure 5048 labeled Multiply Multiple Decoder Outputs lists the input bits to the multiplier digit latches, the corresponding multiply decode outputs, and the multiply decode gate output that is activated for each input bit combination.

During each iteration cycle, the multiplier (J register bits 27-31 or 59-63) is set into the multiplier digit latches, Figure 5048. The output of the multiplier digit latches provide the inputs to the multiply decode circuits, and the output of the multiply decode circuits provide the inputs to the multiply decode gates. The multiply decode gates decode which working register (multiple) is gated to the true/complement input of the main adder. At the same time, the adder is conditioned for either a true or a complement add by the functional OR out gate to AM. The K register which contains the X1 multiple on the first iteration cycle is gated to the normal input of the main adder if Mplr Dig 3 L contains a one on the first iteration cycle. The K register is gated to the main adder by the Shift Gate to AM logic. During the following iteration cycles, the K register is always gated to the normal input of the main adder because the partial product is stored in the K register after each iteration cycle. The K register is gated straight or right four to the normal input of the main adder depending on the cycle in progress. If during the first iteration cycle, the X1 multiple is not needed, parity is gated to the normal input of the main adder and the output of the main adder latches (AMOB) is returned to the K register replacing the X1 multiple with a partial product. The left hand series of columns labeled 1st Iteration (X1 multiple) in the chart in Figure 5048 show the decoder outputs during the first iteration cycle. The right hand set of columns labeled Iteration > 1 (not X1 Multiple) show the decoder outputs during the following iteration cycles. If a multiple is decoded as a two cycle multiple, the first cycle gates either the X2 multiple or the X6 multiple to the main adder and the second cycle gates the X8 multiple to the main adder.

GATES AND OR'S

Address OR

The address OR gates either the channel address bus (CAB) bits or the storage address register bits to storage (Figure 5083). The address OR provides a storage address bus to each 2365 storage unit. Each storage address bus consists of 14-bits plus two parity bits, PA and PB.

On Models I75 and J75, bit positions 5-18 of the address OR correspond to bit positions 1-14 of each storage address bus; on the Model H75, bit positions 6-19 of the address OR correspond to bit positions 1-14 of each storage address bus. The PA bit is the parity-bit for storage address bus positions 1-7; the PB bit is the parity-bit for storage address bus positions 8-14.

A parity check circuit checks the validity of all 24 address OR bits. A parity adjust circuit converts the three parity bits associated with the 24-bit addresses to PA and PB parity bits for the 14-bit addresses sent to storage.

Bit positions 0-20 of the address OR are sent to the address compare circuits where each address passing through the address OR is compared against the address set in the system control address keys. An address compare signal is generated whenever a storage address matches the setting of the address keys. This address compare signal is routed to all logic gates for use as a scope sync during maintenance. The address compare signal is also routed to the system console circuits where it may generate a central processing unit halt signal depending on the setting of the stop on address compare switches.

Bit positions 0-5 on the address OR are sent to the invalid address detection circuits. The job of these circuits is to signal "invalid address" whenever a storage location not available on this particular system is addressed. The bits used for detecting an invalid address depend upon the particular main storage configuration.

Key Gate

The key gate gates the four-bit storage protect feature key from the program status word or a four-bit key plus a read-protect bit from general register R1 (Figure 5064). The program status word, bits 8-11, is gated to the key OR on a central processing unit store and a central processing unit fetch operation. General register R1, bits 24-28, is gated to the key

OR on a "set storage protection key" (SSK) instruction.

The bus control unit receives a full byte (and the byte-parity) from each of the two key sources. To generate a parity-bit for the four or five bits sent to storage, the bus control unit does an exclusive OR of the ungated bits in the byte. The result, if a 1 bit, means that an odd number of bits are being removed from the byte, and therefore, the byte-parity bit must be changed. The result of the exclusive OR on the unused bits is exclusive OR'ed with the original byte parity to obtain the adjusted parity-bit. This action inverts the original parity-bit when an odd number of bits are being removed from the byte and leaves the original parity intact when an even number of bits are being removed from the byte. The adjusted parity-bit is sent to storage as the parity for the four or five bits gated to the storage protect feature.

Adjusting byte parity based on bits removed from the byte prevents correcting bad parity. If the byte originally had good parity, good parity is sent to the storage protect feature storage. If the byte had bad parity, a wrong parity-bit is sent to the storage protect feature storage where it will be detected and a parity error is generated.

During interrupts, the protection key is inhibited to store the old program status word. A key of all zeros inhibits key checking within the storage protect feature.

Key OR

The key OR gates either the channel key-bits or the central processing unit key-bits to storage (Figure 5065). The gating lines which control this OR are the same lines that control the mark OR. Notice that each position gates the no bit condition; outputs of the OR are inverted before being sent to the storage units.

Mark OR

The mark OR is an eight-bit plus parity bit OR which is used to gate the mark register or channel marks to storage. The mark OR gates either the central processing unit mark-bits (from the mark register) or the channel mark-bits to storage on store operations (Figure 5069).

Each position gates the no bit condition from either the mark register or from the channel mark bus. Inverters on the outputs of each position change the no bit condition to the bit condition; these inverters also provide the driving power to send the mark bit to the storage units.

VFL Byte Gates-LBG and RBG

- Data input to VFL circuits is through the LBG and RBG.
- Op 1 data is gated from the K register through the LBG one byte at a time.
- Op 2 data is gated from the K or L register through the RBG one byte at a time.

Left Byte Gate (LBG)

The left byte gate (Figure 5049) consists of AND and OR logic that provides a one byte data path from the K register or from the digit buffer-digit counter to the variable field length circuits. All data and parity bits of the K register and digit buffer-digit counter are inputs to the left byte gate. The selection of the register gated through the left byte gate is determined by the set status of one of two gating triggers, "gate T decode out" or "gate DB-DC to LBG."

Gate T Decode Out Trigger: The majority of the variable field length instructions gate Op 1 data bytes from the K register through the left byte gate to the variable field length circuits one byte at a time. One of the eight data bytes in the K register is selected by the T pointer through the T decode out circuits (Figure 5050). T decode out selection is enabled by the on status of the "gate T decode out" trigger. During the execution of variable field length instructions, the T decode out trigger is set on during those cycles in which a data byte from the K register is needed; during other central processing unit cycles, the T decode out trigger may be reset.

Gate DB-DC to LBG Trigger: During certain cycles of the execution of MVO, PACK, or UNPK instructions the data byte contained in the digit buffer-digit counter is gated to the decimal adder through the left byte gate. The set status of the "gate DB-DC to LBG" trigger enables this gating. Therefore, the gate digit buffer-digit counter to left byte gate trigger is set and reset at various times throughout instruction execution time.

The gate T decode out and gate digit buffer-digit counter to left byte gate triggers are mutually exclusive; a set to one becomes a reset to the other (KW 433).

Left Digit Gate: The left digit gate (Figure 5049) provides a data path from the left byte gate to the left input of the decimal adder. The left digit gate is split to provide independent gating of the high order digit and the low order digit.

Right Byte Gate (RBG)

The right byte gate (Figures 5051 and 5052) consists of AND and OR logic that provides a one byte data path from the K or L register to the variable field length circuits. All data and parity bits of the K and L registers are inputs to the right byte gate. The selection of the register and data byte gated through the right byte gate is determined by the set status of one of two gating triggers, "gate K with S" or "gate L with S", and the value (0-7) contained in the S pointer.

In general, variable field length instructions gate Op 2 bytes, one at a time, from the L register through the right byte gate to the decimal adder or the AND-OR-exclusive OR. When data bytes are gated from the L register through the right byte gate, the "gate L with S" trigger is set. When Op 1 and Op 2 are both contained within the same storage word, Op 2 data bytes are then gated from the K register through the right byte gate; in this case, the "gate K with S" trigger is set and "gate L with S" trigger is reset.

Gate L With S Trigger: The on status of the "gate L with S" trigger gates the data byte selected by the T pointer from the L register through the right byte gate. For the majority of the SS instructions, the "gate L with S" trigger is set during the set-up sequence; thereafter, it is reset or set at various times depending upon the instruction and conditions that arise during execution. For example, during the execution of a decimal divide instruction the "gate L with S" trigger is reset at the end of each pass through the divisor, then set again just prior to the start of the next pass. "Gate L with S" trigger is always reset with E last cycle.

Gate K with S Trigger: The on status of the "gate K with S" trigger gates the data byte selected by the S pointer from the K register through the right byte gate. The "gate K with S" trigger is set when the next Op 2 byte to be processed is contained in the K register; this condition occurs when both operands or a portion of both operands are contained in the same storage word (overlapped-see Overlap Control). For certain variable field length instructions, an overlap condition is determined during the set-up sequence and the "gate K with S" trigger is set at the end of set-up or during iteration cycles. "Gate K with S" is set at the end of set-up when the starting byte of both operands are in the same storage word, or during iteration sequence when the byte gating of Op 2 moves into the same storage word from which Op 1 bytes are gated.

The set condition of the "gate L with S" and "gate K with S" triggers is mutually exclusive; therefore, the set of one trigger resets the other. "Gate K with S" trigger is always reset with E last cycle.

Zero Detect--RBG

Correct execution of the AP, SP, ZAP, and TRT instructions require zero detection of the Op 2 byte. The Op 2 byte is zero detected at the output of the right byte gate. As shown in Figure 5051 the zero status of the high order digit and the lower order digit from the right byte gate are AND'ed to set the right byte gate zero detect latch. The right byte gate zero detect latch is set during each central processing unit cycle that both digits of the right byte gate are zero. The on or off status of the right byte gate zero detect latch determines the sequencing of certain phases of the above instructions.

During the execution of a TRT instruction, if a non-zero function byte is encountered (right byte gate zero detect latch off), the mark sequence is entered and the instruction is terminated.

During the execution of variable field length AP, SP or ZAP instructions when all Op 1 bytes have been processed the right byte gate is zero detected. If a non-zero byte is detected, VFL T6 trigger is set to cause an overflow interrupt to occur during the subsequent SF sequence.

Zero Detect - VFL Result

Result bytes from the decimal adder or the AND-OR-exclusive OR are gated to the K register by way of the K in bus. To determine the zero or non-zero status of the result byte, the eight data bit lines of the K in bus are monitored during each central processing unit cycle in which a result byte is gated. AND and OR logic (Figure 5053) performs the zero detection and sets the variable field length result zero detect trigger if the result byte contains one or more data bits; the on status of the trigger indicates a non-zero result, the off status a zero result.

The variable field length result zero detect trigger is reset off the cycle following E last cycle of each instruction; thus, the execution of each variable field length instruction starts with the trigger reset. For those variable field length instructions that require zero detection of the result, the variable field length result zero detect trigger is released during central processing unit cycles that gate a result byte to the K in bus. The first non-zero result byte gated sets the variable field length result zero detect trigger to

the on state. The trigger then remains on for the remainder of the instruction execution. Certain variable field length instructions interrogate the status of the variable field length result zero detect trigger and set controls that determine subsequent sequencing. For example, if a complement add operation is near completion, the status of the variable field length result zero detect trigger and the decimal adder carry trigger determines whether a change sign or recomplement pass follows.

In general, all eight bits of the K in bus are zero detected; however, for those variable field length decimal instructions that gate a sign digit with the first byte, zero decoding of the low order digit (K in 4-7) is suppressed during the cycle that the sign digit is included in the result byte.

REGISTERS AND BUFFERS

AB Registers

- The AB register is the main instruction buffering unit.
- Both the SBO and J register have data paths to the AB registers.
- The one output from the AB registers is to the IOP register.

The AB register, to which the storage bus out (SBO) latch output is connected, forms the main instruction buffering unit. The AB register is two 64-bit (plus eight parity bits) registers which have identical data inputs and are selected alternately to receive the input data. Thus, there is an instruction buffering capacity of two 64-bit words. The decision of which register (A or B) the data are returned to is based upon the address of the requested instruction. Even addresses are returned to the A register and odd addresses are returned to the B register. The setting of the AB register is timed with a Late BR Clk which is adjusted individually for each of the four boards containing the AB registers.

There are two data inputs to the AB register. One from the storage bus out latch and the other from the J register. The gate line which gates the storage bus out data to the AB register is conditioned when the J register gate line is not conditioned, i.e., the J register gate line is the inverse of the storage bus out gate line. The J register input is needed during successful branch operations. It is used to transfer prefetched branch operands, which have been returned to the J register, back to the AB register. See Figure 5054.

There are two additional positions in the AB register which are not part of the data path. These are the A and B invalid tags. These register positions are set when a fetch to the AB register is attempted from a non-existant storage address. The data which is placed in the AB register under these conditions is not from storage but from the maintenance panel keys. This data has no meaning and is used only to prevent parity errors. These register positions may also be set from the J register on branch operations and by scan controls.

AB OR's

There is one output from the AB register. This output forms the input to the instruction operation register, which is a 32-bit (plus four parity bits) register used for first cycle decoding. Since the AB register contains 144 bit positions and the instruction operation register contains 36 bit positions, the instruction operation register cannot accept the full AB register at one time. Also, since instructions may start at any halfword, facilities are provided for gating from the AB register on 16-bit boundaries to the instruction operation register. This means that each instruction operation register position is able to accept any one of eight possible AB register bits. To accomplish this, the input to each instruction operation register position is the result of eight, two-way AND's feeding two four-way OR's, see Figure 5054, which in turn are dot-OR'ed, which is in effect an eight-way OR. There are eight gate select lines which determine the 32 bits to be gated into the instruction operation register. If the instruction is in the RR format and only 16-bits are needed, the second half of the instruction operation is ignored.

BOP Register

- The BOP register contains the op code, and the R1 field.
- The BOP register is set at the beginning of the second I cycle.

The buffer operation (BOP) code register is a 13-bit register containing the eight-bit operation code, the four-bit R1 field and the parity bit for the operation code. The register contains the required information to successfully complete the transition between the instruction (I) box and the execution (E) box instruction execution. It is set at the start of the second cycle of I time and remains valid as long as is necessary, normally until the next set time. The contents are also used for instruction executions within the I unit.

BOP Parity Check

The operation code, bits 0-7 of the instruction, is checked off the buffer operation register. The checking consists of a nine-way exclusive OR which

generates an error signal that is sampled into a trigger with an A clock pulse when the execution unit starts the instruction. This checking station detects an odd number of errors in any one of the following areas:

1. The storage data
2. The AB instruction buffer registers
3. The AB gating to IOP
4. The IOP register
5. The data path from IOP to BOP
6. The BOP register
7. The checking circuitry.

BR 1 Field Incrementer

The buffer operation register R1 incrementer is a four-bit, plus 1 adder. The output is latched and sets back into the BR 1 register on control from the instruction box execution unit. During the Store MPL instruction, the BR 1 field is continually incremented for gating out consecutive registers from the general purpose register stack. When the BR 1 and IR 2 compare circuitry signifies a match, the Store MPL is concluded.

The incrementer consists of one level of parallel carry lookahead logic to propagate carries into each bit position followed by a two level exclusive OR to generate the sum. The output feeds a latch. Overflow carries are ignored because a wrap-around is desired; therefore, an initial input of 15 generates a zero output. As mentioned, the BR 1 register sets on an A clock pulse. The latch control is generated from a not B clock pulse so the latched output brackets the register set. (The latch is released for the duration of the B clock pulse.)

Direct Data Register

- The direct data register is an eight-bit register without parity.
- The direct data register is used during the write direct instruction.

The direct data (DD) register is an eight-bit register without a parity bit. The direct data register is used as a storage device for placing a data byte on the direct data out bus during the execution of the write direct (WRD) instruction.

During the SF 4 cycle of the write direct instruction, a data byte from the left byte gate is gated to the AND-OR-exclusive OR and the direct data register. The direct data register is then released during Seq Lth A time and is set at the following AR clock pulse (Figure DM5055). Because the direct data register is not reset, data set into it by one write direct instruction remains until the next write direct instruction is executed.

EOP Register

- The EOP register contains the Op field to control execution of the instructions.

The E unit operation (EOP) register is a group of nine triggers including a parity trigger. The register contains the instruction operation code of the instruction which the execution unit is executing or about to execute. The register is set from positions 0-7 of the instruction operation register one cycle before the first execution cycle. The output of the register is decoded in the E operation decoder (EOP Decoder) and is used to gate data depending on the instruction.

ER 1 Register

- The register is a four-bit register which is set from BR 1 at the I to E transfer.
- The ER 1 field specifies the GPR or FPR which receives the result of the instruction.

The ER 1 register is a four-bit field and is set from BR 1 when the instruction to execution transfer (I to E transfer) occurs. The ER 1 field specifies the register in the general purpose or floating-point register stack which receives the result of the instruction. A register decoder is used to select the proper register for in-gating. If more than one register is to be used--LD MPL--the ER 1 field is incremented. The setting of the field occurs on a B clock pulse; the BR 1 transfer occurs on an A clock pulse. The ER 1 bits are also compared with the R2 field of the Instruction operation to determine the conclusion of the LD MPL instruction.

ER 1 Field Incrementer

The ER 1 incrementer is used when more than one general purpose register is stored during an E box execution. The output of the ER 1 incrementer is returned to the ER 1 register and is set on a B clock pulse. The field is incremented for a put-away during the following central processing unit cycle.

The implementation of the incrementer is exactly the same as the one for the BR 1 register. It also has a latched output; the latch control is conditioned by an A clock pulse which allows new data to be set into the latch.

ER 1 Compare

The ER 1 field is compared to the R2 field in the instruction operation register. A compare is required to signify the completion of the LD MPL instruction. During this execution, the storage data is consecutively loaded into the general purpose register selected by the ER 1 decoder until the register specified by the R2 field is encountered. The ER 1 field is incremented after each put-away, thus the compare line rises prior to the last cycle control. The compare signal is latched with an L clock pulse.

Exponent Register

- The exponent is an eight-bit latch register.
- The register has two input data sources.
- The register has three data outputs.

The exponent register (ER) is an eight-bit, plus parity, latch register with inputs from the exponent adder output latches (AEOB) and L register positions 8-15. Figure 5056 shows the data flow into and out of the exponent register; the register is set during A clock time by the gating conditions shown in the figure or by the scan word 12 and scan pulse 2 during fault location test mode of operation. The exponent register may contain an operand exponent result sum or a result exponent difference during floating-point instructions or a word count, which is reset to zero during the variable field length set-up sequence--logical instructions. During variable field length logical instructions the exponent register is advanced by one each time a result word is stored. Bits 9-15 of the L register are used during fault location testing (FLT) operations to directly set the exponent register when the gate, scan word 12 and scan pulse 2, is conditioned.

The exponent register output conditions the normal and the true/complement input of the exponent adder, bits 56-63 of the floating-point registers, and condition register bits 34 and 35 of the program status word.

Figure 5056 shows the output gating for the exponent register. The register is gated out during fixed-point, floating-point and variable field length instructions. The exponent register output to the condition register (bit 0) is gated at the condition register, and the exponent register output to the floating-point register is gated at the input to the floating-point register. The outputs to the exponent adder normal and true/complement inputs are gated at the output of the exponent register.

In order to set a bit into the exponent register and obtain an output from the AND circuits supplying inputs to the exponent adder or floating-point registers, the register bit latch must have a positive output indicated by the name above each bit position. To obtain a positive output from the exponent register latch, the input AND's are considered as -OR's and the output OR's are considered as -AI circuits.

Assume the following conditions for exponent register bit 0:

1. -AE bit 0 to ER is negative,
2. -L scan bit 08 is positive,
3. +release exp reg is positive,
4. -release exp reg is negative, and
5. The latch back is positive.

With these conditions, one input for each of the -OR's is conditioned. Since one input to each of the -OR's is conditioned, the two inputs to the -AI are conditioned. With the -AI inputs conditioned, the output of the block is positive.

The positive output from the AOI block is inverted by the invert block which conditions the latch back circuits to the AOI. When the +release exp reg and the -release exp reg inputs reverse their states, the bit is retained in the latch regardless of the input conditions on the -AI bit 0 to ER input or the -L scan bit 08 input to the exponent register bit 0.

With a bit in exponent register bit 0, any gate (+gate ER to AE or +gate ER to AETC) allows the content of the exponent register to be transferred to the normal or the true/complement input of the exponent adder. The outputs to the condition register and the floating-point registers are gated at their respective registers.

Floating-Point Registers

- There are four floating-point registers.
- The registers have two inputs.
- The registers have three outputs.
- The register addresses are 0, 2, 4, and 6.

The four floating-point (FLP) registers each contain 64 data bits plus eight parity bits. Their addresses, specified by the R1 and R2 fields in the instruction format, are 0, 2, 4, and 6. Bits 0-55 are loaded from the K register and bits 56-63 are loaded from the exponent register by floating-point load, load type, or arithmetic instructions.

The floating-point (FLP) register outputs are gated to the registor bus latch (RBL), exponent adder (AE), and sign control. The entire contents of the floating-point register are gated to the register bus

latch; bits 56-63 are gated to the exponent adder, and bit 56 is used in the sign control circuits.

Figure 5057 is the data flow into and out of the floating-point registers, and the gating and selection circuitry necessary to place data in the registers or to transfer data out of the registers. In Figure 5057, a typical bit position is shown for the four floating-point registers with the E operation selection and E cycle timing used to gate data into one of the four registers during execution time of the floating-point instructions. The output gating and register selection is controlled by instruction operation, buffer operation, the FR 2 trigger and the floating-point out (FLOUT) trigger being on. The exponent, bits 56-63, is gated either to the register bus latch or to the exponent adder and is dependent upon the gating shown in Figure 5057. Either bytes 0, 1, 2, and 7 or bytes 0-7 are gated to the floating-point register by the byte gating circuitry depending whether the instruction format is decoded as floating-point short operand or floating-point long operand. The contents of the low order half, bits 24-55, of the floating-point register are not changed during short precision floating-point instructions.

When one or more of the bits are set in Figure 5057, a minus level is obtained at the output of the latch register. In order to have a bit present on the -FLP Bit 0 to RBL line the following conditions must be present on the AND/OR feeding the output AND circuit:

1. Assume the input AND's are negative OR's and the OR is a -AI.
2. Assume all positive gate out register lines are minus except the line labeled +Gate Out Reg 0 Byte 0.
3. Assume a bit in minus Reg 0 Bit 0 Latch, thus giving a minus Reg 0 Bit 0 level.

Therefore, each of the input AND's (-OR's) have a negative input. With the negative input to each AND (-OR) four negative inputs to the OR (-AND) is realized. With all inputs conditioned to the OR, a positive level is obtained at the output. With a positive output and a positive gate out Byte 0 to RBL a negative FLP bit 0 to RBL is obtained. Likewise, if a bit is not present in the minus Reg 0 Bit 0 latch, the AND (-OR) or the OR (-AND) is not conditioned; therefore, a negative level is obtained at the output which indicates the absence of a bit in the addressed floating-point register bit position.

In order to set a bit into floating-point register 0 bit 0, the following conditions must exist:

1. +K Reg 00 to FLP is positive,
2. +Set Reg 0 Byte 0 is positive,
3. -Reset Reg 0 Byte 0 is positive, and
4. the latch back line is negative.

Since the upper AND circuit is conditioned, one input to the OR circuit is also conditioned. The output from the OR circuit is negative, and feeds the invert block. The output of the invert block is positive and places a positive level on the input of the latch back line. The set and reset lines will return to the opposite states when the input gating conditions are removed; however, this will not affect the output of the latch even though the input data to the register is removed. The only way to reset the latch is by again conditioning the set and reset lines without data being present on the line labeled +K Reg 00 to FLP.

General Purpose Registers

- There are 16 general purpose registers.
- The registers may contain operands, indexes, addresses, counts, etc.

There are 16 general purpose registers (GPR), each consists of 36 bits (32 data bits and four parity bits). The general purpose registers may be specified as operand locations for fixed-point arithmetic or they may contain addresses, index quantities, counts, etc. The 16 general purpose registers are selected on the basis of the decoded outputs of various four-bit fields in the instruction.

The general purpose registers are packaged on four boards. Each board contains four complete registers. Board G-B3 contains general purpose registers 0-3, G-B4 contains 4-7, G-C4 contains 8-11, and G-D4 contains 12-15. With this layout the even-odd pairs of registers used in multiply, divide and shifting are contained on one board. Selection of the registers for in and out gating is simplified with this packaging scheme.

To Load a GPR

The general purpose registers are loaded from the K register positions 0-31. Three conditions are necessary to load a general register as shown on Figure 5058. Condition 1, a gate in timing (Early B clock) is always available. Each board containing four registers has its own clock line so that it may be timed individually. Condition 2, a gate control line from the execution unit, is available at all registers when any one is to be loaded. Condition 3, is a register select line from the ER 1 decoder. The line is active when the particular register of the 16 registers is to be loaded.

Specially wired select lines enable the scan controls to gate the data in the K register to general purpose register zero and general purpose register four. This facilitates pattern testing of the addressing adder.

To Gate Out a GPR

As shown on Figure 5058, each general purpose register bit feeds two output gates. These out gates make up two busses; general bus left (GBL) and general bus right (GBR). A single register may be gated to both busses simultaneously. The last level of OR'ing for general bus left is done at one of the addressing adder inputs, and general bus right is similarly OR'ed at a second input to the addressing adder. Gated outputs from the above OR's are sent to the register bus latch in the execution unit. The gated output of general bus left goes to the register bus latch bit positions 0-31 and the gated output of general bus right goes to register bus latch 32-63.

To gate a general purpose register on to a bus, two conditions are necessary: a register select line and a control signal. The register select lines for general bus left are obtained from the decoded outputs of either the B field in the instruction operation register or the R1 field in the Buffer operation register. The register select lines for general bus right are obtained from the decoded outputs of the R2 (X) field of the instruction operation register. A wired in shift of one from the RI field of buffer operation register is also provided for gating odd registers to the general bus right. For example, if R1 = 2, the wired in shift selects general purpose register three for out-gating to the general bus right. For any instruction requiring an even-odd pair of operands from the general purpose registers (e.g. fixed-point multiply, fixed-point divide) the operand at the odd address is obtained by the use of the wired shift. R1 of the buffer operation register contains the address of the even register in such cases. There are four gate out control lines for gating the four sets of select lines (R1, R1 + 1, B and R2 (X)) to the general purpose registers. Each control line is LSA'd to all four general purpose register boards. Each gated select line directly gates out one of the general purpose registers to either general bus left or general bus right. General purpose register one and general purpose register two have a special line to gate them on the general bus left bus. These lines are used by the execution unit during "EDIT AND MARK" and "TRANSLATE AND TEST" instructions.

Checking

There is no checking performed in the general purpose registers. Data is checked at the source and destination points. The general purpose registers are indicated by gating them to the register bus latch. The display general register and load general register manual functions are done using normal select and control paths.

H Register

- The H register is used as an auxiliary address register.
- The H register has inputs from the addressing adder and the incrementer.
- The H register has outputs to the incrementer, compare circuits, channel controls, I and E unit controls, interrupts, etc.

The H register is a 24-bit register used as an auxiliary address register. In branch operations, the H register contains the branch address. In the incrementer, one is added to the contents of the H register and the result is placed in the storage address register for the the branch plus one fetch. If the branch is successful, the H register contents are transferred to the instruction counter register (ICR). During multiple load and store operations, the H register is used in conjunction with the incrementer to update the effective storage operand address. The H register is used on shift instructions and variable field length instructions for transferring operands to the K register. On store instructions, the contents of the H register are compared to both the contents of the instruction counter register and to the incrementer output. If either comparison is satisfied, a store has been made into the instruction counter area, and re-fetching of the AB register contents are required.

The H register, along with the instruction counter register and the incrementer is packaged on three boards. Bits 0-7 and parity are on G-B1, bits 8-15 plus parity are on G-A2, and bits 16-23 plus parity are on G-A1.

Inputs to the H register come from the addressing adder and the incrementer. The adder bits are at the end of an LSA chain and come to the H register via the storage address register boards. The incrementer bits are located on the same board as the corresponding H register bits. Both inputs are gated in at the H register at A time. See Figure 5059.

All 24-bits of the H register are gated to the incrementer. Bits 0-20 feed the program store compare circuits for the comparison to the instruction counter register and the incrementer. Bits 13-23 are selection bits for channel controls. Variable field length and the execution unit receive bits 18-23 for shift counts, byte addresses, etc. Bits 20-23 are received by the instruction unit. Bits 20 and 23 have latched outputs. Bit 23's latched output is used by interrupts and bit 20's latched output is used by the gate select mechanism.

IOP Register

- The IOP register is a 32-bit bipolar register.
- The IOP register has inputs from the AB register, scan, and the addressing adder.
- The IOP register output is decoded by the IOP decoder.

The instruction operation (IOP) register is made up of 32 bipolar data triggers, see Figure 5060. The advantage of the bipolar trigger is that both phases of the output are available simultaneously. However, both phases of the data input lines and gate lines are also required to successfully set the triggers. The main input to the instruction operation register is from the AB register OR's. This input is indicated at the output of the OR's and is gated into the instruction operation register with a "set IOP control" line. This gating line contains a controlled A clock sample. A second input to the instruction operation register is the scan input. The data for the scan input comes from the J register bits 0-35 during fault location test mode of operation. J register bits 0-31 supply inputs to scan inputs 0-31 respectively of the instruction operation register, and J register bits 32-35 are the inputs to the four parity positions of the instruction operation register. These inputs are gated with the scan gate line which is conditioned at scan word two, pulse two during the scan operation as previously mentioned.

A third input is present on positions 8-15 of the instruction operation register. These bits are from the input of the addressing adder positions 24-31, and are gated into the instruction operation register on an execute instruction. These bits are, in effect, OR'ed with the R1 field and the X field of the subject instruction, which in turn changes the designation of the referenced general purpose register. Since the execute bits are OR'ed with the instruction operation register bits only the positive phase of these bits are used. This is because the bits in the instruction operation register may be changed from zero to one by the execute bits but never from one to zero.

Decoding of the instruction operation register is done over several fields: the operation (OP) field (bits 0-7), the R2 or X field (bits 12-15), and the B field (bits 16-19). The D field (bits 20-31) is not decoded but instead is gated to the addressing adder for use in address modification. Two parity bits (P20-23 and P24-31) are used for checking this field in the adder. The R1 field (bits 8-11) is not decoded from the instruction operation register.

All first cycle decoding and some second cycle decoding for the instruction box is decoded from the instruction operation register. A small amount of pre-decoding is done from the output of the AB OR's for use by the instruction counter.

The decoding of the R2 or X field and the B field is identical. Each four bit field is decoded in two levels to form sixteen lines to select one of the 16 general purpose registers. These fields are also tested for an all zero condition.

Parity for bits 20-23 of the D field are generated because the parity accompanying the data to the instruction operation register is for the eight bits 16-23. This parity is generated by generating parity for bits 16-19 (B field) and combining this parity with the eight bit parity with an exclusive OR. The result is the parity for bits 20-23.

GPR Out-Gating Controls

All output gating from the general purpose registers is controlled in the instruction unit. Instructions may call for general purpose register fetches from any one of the following fields--R1, R2 (or X), B, or the implied R1 plus 1. These four-bit fields are decoded and select the proper register when an out-gate control is raised. All fetches are made during one of these sequencing cycles--T1, T2, GROUT (general register out), or under variable field length control.

During the T1 cycle, an address calculation, or in some cases, an add for an operand quantity, is made. In this cycle, the R2 or X field or the B field or both may be requested. For RR format or RX format instructions, the general purpose register specified as the R2 (or X) field is always gated to the adder. For all non-RR format instructions, the B register is gated. If the particular field is all zeros, this implies that no register is required for address calculation. The registers selected by the decoded field are gated through the addressing adder and, when required, the adder sum is set into the storage address register and the H register. One or more registers are always gated to the adder during the T1 cycle whether any are required or not. The format decoding and the zero conditions mentioned are performed directly from the instruction operation register rather than the instruction operation decoder to obtain a speed advantage in the cycle.

The T2 cycle of I time represents the pre-fetch cycle. Operands are obtained from storage or from the general purpose registers and the floating-point registers. These registers are gated to the register bus latch for use by the execution unit. In all cases, two registers are gated out of the general purpose register stack. If the instruction is not

floating-point, then the general purpose register busses are gated to the register bus latch. One of the registers corresponds to the R1 field; the other is either R2 or R1 plus 1. In the RX format and the RM SHFT instructions R1 plus 1 is gated out; in the others, R2 is gated.

Following the two-cycle instruction time, all further general purpose register operand fetches are controlled by the general register out-gate trigger (GROUT). As in the pre-fetch cycle there are always two registers gated out. One of these registers is always R1; the second register is R1 plus 1 except for RR format FX MPY and RS format BR ON Index instructions when R2 is required. General register out-gate trigger is also used to gate the registers to the execution unit.

During variable field length executions the instruction box has the function of addressing storage for the required operands. A variable field length control line is used to maintain the B field input to the adder for calculating these addresses. The register designated by the B field (except general purpose register zero) is constantly added to a 12-bit field in the instruction operation and to a length originating in the execution unit. When needed, the adder output is set into the storage address register and the storage operation is initiated.

Associated with the general purpose register controls is the D field control. This line gates the 12-bit displacement field to the addressing adder. In most applications the same timing as the B field exists. The variable field length control mentioned above always gates this field and T1 does for all non-RR format instructions.

There is one special case which occurs during the Execute instruction. Bits 24-31 of the contents of the general purpose register specified by R1 are OR'ed in instruction operation with bits 8-15 of the subject instruction. To implement this OR'ing, general register out-gate trigger is set to gate out R1 and T1 is set to allow the initial instruction operation set. This is the only case when general register out-gate trigger and a sequencer are both on. To prevent the subject instruction from affecting the gate out of R1 (because T1 is on) a special line is required which blocks the B gate control. This B inhibit is necessary since both B and R1 use the same data path. The R1 register bits are OR'ed into instruction operation and, after another T1 cycle, the instruction processing continues with the T2 cycle.

When the machine is not under program control, it is possible to display any one of the general purpose registers with a special control. This DSPLY GPR line forces the R1 gate out and also forces the gate to the register bus latch. Four register select lines choose one of the 16 general purpose registers for display. A similar scheme is employed for displaying the floating-point registers.

The gate control to register bus latch mentioned above is also located in the instruction decode logic. This line gates the two 32-bit general purpose register busses, located at the adder input, to the register bus latch in the execution unit. The control is raised during the T2 cycle on all non-floating-point instructions, when general register out-gating trigger is set, when the general purpose register bus is displayed, and when the adder input is displayed (also under manual control).

Invalid Operation Detection

Part of the instruction operation decoder is reserved for the invalid operation detection. The output represents all the unassigned operation codes for the machine. If any of these codes appear in the instruction operation register, the output raises and a program interrupt is initiated.

The decoding is four level AND-OR-invert logic as in the instruction operation decoder. All the unassigned codes in each format are generated and, in the final level, they are AND'ed with their respective formats. The output is also used to prevent sampling the adder error line to insure that no machine check results from an invalid operation code.

Addressing Compare

Due to the overlapped instruction-execution operation of the system, it is possible for the instruction unit to require a general purpose register which the execution unit may be loading during its previous instruction. This condition exists during the T1 cycle in the instruction unit when the effective address is computed. The conflict occurs on register specified as the R2 (or X) and the B fields, since these are the ones used for address addition on the first cycle.

Compares between these fields and the R1 field of the buffered operation register are made, and the outputs are gated with control lines associated with instruction unit and execution unit operations. The instruction unit determines which registers, if any, are required during the first cycle. The R2 (or X) field is used for RR format branch and all RX format instructions; the B field is needed for all RX format and RS format instructions. The buffered operation register, during the T1 cycle, contains the operation code of the previous execution instruction. Buffered operation decoding determines whether a putaway into a general purpose register occurs and also whether there is a coupled register (even-odd) putaway as in FX MPY and SHFT DBL. In this

instance, the odd-even field bit does not participate in the compares.

The final output--CMP BLK--prevents the instruction unit from continuing with the instructions until the putaways are complete. The output is latched with an A clock pulse.

A compare has no significance if the field contains all zeros since general purpose register zero cannot participate in the address calculation.

The compare between the R1 field of the buffer operation register and the R2 field of the instruction operation register is also used to control the ending of the store multiple instructions. This compare line also is sent to the execution area of the instruction unit and is latched with an A clock pulse.

The compares are implemented by a bit-for-bit exclusive OR which determines equality, followed by a four-way AND to combine the bits of the field.

IOP Parity Check

Bits 8-15 of the instruction operation register are checked each time a new instruction enters the instruction unit. A nine-way exclusive OR checks the parity of the instruction operation bits and generates an error signal which is sampled into a trigger when a sequencer, T2, is set.

J Register

- The J register receives operands from storage, GPR, and FLP registers.
- The J register has outputs to RBL, AE, AM, and multiplier decoding.
- Quotient insertion logic is used with bits 59-63 of the register.

The J register (Figure 5061) is a 64 data bit plus eight parity bit register. The register receives operands from core storage, the general purpose and floating-point registers via the register bus latch, and from the divide quotient insert logic. The J register has outputs to the main adder, exponent adder, multiplier decoder and the register bus latch register. The output of the J register to the register bus latch is an automatic left four shift while the input from the register bus latch to the J register is either straight or a right eight shift. This arrangement of the left four shift and the right eight shift to the register bus latch register and return allows for shifting the divide instructions quotient left four and shifting the multiplier right four (left four shift to register bus latch and right eight shift to the J register gives a net result of a right four shift) for each iteration cycle.

During divide instructions, the J register is used to assemble the quotient which is transferred to the K register via the main adder for storing into a general purpose register or floating-point register during put-away time. The J register contains the multiplier during multiply instructions. Bits 27-31 or bits 59-63 are used to decode which multiple will be added to the partial product located in the K register.

The J register also has zero detection circuits for detecting zero operands during instruction execution. If a zero operand is detected, specific operations occur to indicate this zero condition. For the operations that occur because of the zero operand conditions, refer to theory of operation section associated with each instruction.

Figure 5061 is layed out with all release (input) controls on the left hand side, the register is contained in the next section (only partial register representation is shown) and the right hand section shows the out-gating conditions.

K Register

- The K register is used as a result and storage register.
- The K register has outputs to the main adder, register bus latch, floating-point registers, and the general purpose registers.

The K register (Figure 5062) is a 64 data bit plus eight parity bit register which is used as a result register during arithmetic operations and as a temporary storage register. It has inputs from the main adder, the variable field length circuitry, and the instruction unit incrementar. Outputs from the K register are to the main adder, the register bus latch, the variable field length circuitry, the general purpose registers, the floating-point registers and to the storage bus in (SBI) for transferring data to core storage.

The four high-order positions of the K register have gates to the main adder from the convert to binary instruction. The result of the main adder is parity checked from the K register.

The K register has zero detection circuits for detecting zero operands during instruction execution. If a zero operand is detected, specific operations occur to indicate this zero condition. For operations that occur because of the zero operand conditions, refer to the theory of operation section associated with each instruction.

Figure 5062 is layed out with all release (input) controls on the left side of the figure. The register

is shown in the next section to the right of the release controls, and the partial register positions is followed by the output gating circuitry associated with out-gating for the K register.

Key Buffer Register

- The key buffer register is five bits plus parity.
- The key buffer is used on the insert key (ISK) instruction.
- The key buffer register buffers the key fetched from SPF.

The key buffer register holds the storage protect feature (SPF) key enroute from the storage protect feature storage to general purpose register R1 on an insert storage key (ISK) instruction (Figure 5063). The key buffer contains five bits plus a parity bit. The five bits consist of a four-bit storage protect key and a read-protect bit. The output of the key buffer register is to the AND-OR-exclusive OR where zeros are added to make a full byte. From the AND-OR-exclusive OR, the byte containing the key is routed into bits 24-31 of the K register. From the K register, the key is set into bits 24-28 of general purpose register R1 along with the added zeros which are set into the remainder of the byte.

L Register

- The L register is used in multiply, divide, and convert instructions.

The L register (Figure 5066) is another 64 data-bit plus 8 parity-bit register. The register contains the X12 multiple of the multiplicand during multiply instructions, the X 3/2 multiple of the divisor during divide, and the operands for the convert instructions. The L register is set from the main adder and has outputs to the main adder and the variable field length section.

LCOP Register

- The LCOP register contains the Op field to control the execution of instructions.

The last cycle operation (LCOP) register is a group of nine triggers including a parity trigger. The register contains the instruction operation code of the instruction which the execution unit is executing.

The last cycle operation register is set from the execution unit operation register at the beginning of the first execution cycle. The output of the register is decoded and used for gating of data pertinent to the instruction.

M Register

- The M register is used primarily to receive operands from the GPR and FLP registers.

The M register (Figure 5067) is a 64 data-bit plus 8 parity-bit register in the execution unit. One of its prime purposes is to receive a pre-fetched operand from the general purpose registers or from the floating-point registers at the beginning of the first execution unit cycle. The operand is received from the general purpose or floating-point registers via the register bus latch.

The M register also has an input from the main adder in addition to the input from the register bus latch. The outputs of the M register are to the main adder and the exponent adder. The M register contains the divisor for divide and the multiplicand for multiply instructions. The outgates to the main adder do bit shifting, generate multiples of the multiplicand, and generate multiples of the divisor. The high order five bits have an output to the divide decoder. The high order three digits of the M register have a zero detect circuit which is used for normalizing floating-point numbers.

Figure 5067 shows the release (input gating) logic on the left side followed by the partial representation of the register bits. On the right side of the figure are the output gating circuits.

Mark Register

- The mark register is eight-bits plus parity bit.
- The register holds the CPU mark bits.
- The register sets either two adjacent bits at a time or one bit at a time by VFL.

The mark register holds the eight central processing unit mark bits used for central processing unit store operations (Figure 5068). The register is set by either of two sets of inputs. For instructions which store halfwords (two bytes) or multiple halfwords (words, double words), the store data follows the address boundary rules. These instructions use a set of four input lines to the mark register. Each of these four lines set two adjacent bits in the mark register. Where a single byte is to be stored, or a variable number of bytes which do not necessarily follow the halfword boundary rules, the eight variable field length input lines are used. Each

input line sets one mark bit; however, only one of these lines is active on any one cycle.

The mark register holds the mark bits for an indefinite period of time. The register is reset only following a central processing unit store operation (or following a test and set operation) which means its contents have just been used. For example, assume that three bytes are to be stored. Preceding the store request, the three appropriate mark bits are set, one at a time. There may be any number of machine cycles from the time that the first mark bit is set until the third mark bit is set. Again, any number of cycles may elapse from the time the third mark bit is set until the store request signal is sent to the bus control unit. Meanwhile, central processing unit fetches can be made and channels can both store and fetch without disturbing the mark register. Only after the central processing unit store operation is started and the mark bits are sent to storage is the mark register reset.

The mark register parity-bit is turned on when the register is reset. This gives correct odd-parity for an empty register. The parity-bit is not changed when the four-line input is used to set mark bits because the bits are always set in multiples of two. When the eight-line input (variable field length) is used, the parity-bit is complemented (changed) each time a mark bit is set.

Notice that the active condition sent to the mark OR is "Mark Bit X = 0" (Figure 5068).

Program Status Word

- The PSW is a 64-bit data register plus 8 parity-bits.
- All bits in the PSW have an input from the J register.

The program status word (PSW) is a 64 data bit plus eight parity bit register, the contents of which are used to control instruction sequencing and to indicate the central processing unit status. The high-order 40 bits (bits 0-39) are used to control or indicate the various central processing unit conditions. The low-order 24 bits (bits 40-63) contain the instruction counter register (ICR).

The 64 program status word bits are packaged on five different boards. Bits 0-23 and 32-39 are on board 01G-B2. Bits 24-31 are located on board 01H-D3. Bits 40-47 instruction counter register 0-7) are on 01G-B1, bits 48-55 (instruction counter register 8-15) are on 01B-A2 and bits 56-63 (instruction counter register 16-23) are located on 01G-A1.

All 64 program status word bits have an input from the J register. These bits are gated at the program status word. Program status word bits 8-31 and 40-63 have gates to the incrementer on their output. Bits 0-7 and 32-39 have un-gated outputs to the incrementer extender. During a "load PSW", the new program status word is sent to the incrementer for parity checking. The program status word passes through the incrementer to the K register on a "store PSW" instruction.

The first byte of the program status word contains the system mask which is used to control channel and external interrupts (Timer, Console). When a mask bit is zero, the corresponding source cannot interrupt the central processing unit. When a one, the source causes an interrupt. Bit 0, the multiplex channel mask bit, is not used in the 2075. Bits 1-6, the selector channel mask bits, have ungated outputs to the channel interrupt controls. Bit seven is the external interrupt mask bit and has an ungated output to the instruction unit controls. See Figure 5070.

Program status word bits 8-11 form the storage protection tag for central processing unit store operations. These bits have an ungated output to the central processing unit tag OR'ing circuits. See Figure 5071.

Program storage word bit 12 is the ASCII mode bit. The sign and zone codes generated for all decimal arithmetic results differ for the extended binary coded decimal interchange code (EBCDIC) and the American Standard code for information interchange (ASCII). When bit 12 is zero, the preferred EBCDIC codes are generated; these are plus, 1100; minus, 1101; and zone, 1111. When bit 12 is one, the preferred ASCII codes are generated; these are plus, 1010; minus, 1011; and zone, 0101 (Figure 5088).

Program status word bit 13 is the machine check mask bit. When this bit is a zero machine check interruptions are not taken. When bit 13 is a one, machine check interruptions occur. This bit has an ungated output to the power distribution unit (PDU) check controls. See Figure 5072.

Program status word bit 14 is the wait status bit. Bit 14 being a one indicates that the central processing unit is in the wait state. Otherwise the central processing unit is in the running state. Bit 14 has an ungated output to interrupt controls and an ungated output to the instruction-execution (IE) controls. See Figure 5072.

Program status word bit 15 is the monitor state bit. When bit 15 is a zero, the central processing unit is in the monitor state. When a one, the central processing unit is in the problem state. This bit has an ungated output to the instruction unit controls. See Figure 5072.

Program status word bits 16-31 are reserved for interrupt codes. They identify the cause of an input/output, program, monitor call or external interruption. Bits 16-20 have no specific code assigned to them and they are reset on any interrupt. Bits 21-23 identify the channel causing the interrupt. They have an input from the channel interrupt controls. This input is gated in at the program status word. See Figure 5073. On channel interruptions, bits 24-31 contain the channel unit address. Other interruptions set an identifying code into bits 24-31. Bits 24-31 have various inputs, all of which are gated in at the program status word.

These inputs are:

1. Channel unit address, bits 24-31.
2. Buffered operation register bits 8-11, bits 24-27.
3. Instruction operation register bits 12-15, bits 28-31.
4. External interrupts, bits 24-31.
5. Execution unit interrupts, bits 28-31.
6. Instruction unit interrupts, bits 28-31.

There is a gated OR in front of bits 24-31 to handle all the various inputs. See Figure 5074.

Program status word bits 32 and 33 preserve the length code of the last instruction. Preceding these bits is a decoder which takes buffered operation register bits 0 and 1 and generates the proper length code for gating into the program status word. See Figure 5075.

Program status word bits 34 and 35 are the condition register. The channel and the execution unit inputs to these positions are received through gated OR's in front of these bits. Bits 34 and 35 also have inputs from the general bus left bits 2 and 3. These bits are used on "Set Program Mask" instructions. See Figure 5076.

Program status word bits 36-39 are the four program mask bits. When the mask bit is a one, the appropriate program exception causes an interrupt. When the mask bit is zero, no interrupt occurs. Bits 4-7 of the general bus left are set into bits 36-39 during a "set program mask" instruction. These bits have ungated outputs to the execution unit interrupts. See Figure 5077.

Program status word bits 40-63 are the instruction counter register bits 0-23. Instruction counter register bits 0-19 and bit 23 have inputs from the incrementer. These bits are gated in at the instruction counter register. Instruction counter register bits 20-22 have inputs from the gate select register which are also gated in at the instruction counter register. Bits 0-20 have ungated outputs to the program store compare circuitry. Bits 20-22 have ungated outputs to the gate select adder. The parity bit for instruction counter register byte 16-23 has inputs from both the gate select register and the incrementer. These inputs are gated in at the instruction counter register. The parity bit for bits 16-23 has an ungated output to the gate select adder. See Figure 5078.

Register Bus Latch

- The RBL is a 68 data bit latch.
- The RBL buffers operands for one cycle.
- The RBL has inputs from GPR, FLP, J, and K registers.
- The RBL has outputs to the J and M registers.

The register bus latch (RBL) is a 68 position, plus eight parity bits, latch. The primary use of the register bus latch is a buffer for operands from the floating-point registers, the general purpose registers, or data from the K register and the J register.

Figure 5079 is the data flow into and out of the register bus latch. When the contents of the J register are gated into the register bus latch, an unconditional left four shift is taken. Figure 5079 shows the J register position zero is gated into register bus latch position -4 and J register position 63 is gated into register bus latch position 59 everytime the J register is transferred to the register bus latch.

The K register and the floating-point registers are gated straight to the register bus latch, and the general purpose registers are gated to register bus latch positions 0-31 or 32-63. The addressed general purpose register is 32 bits in length. Therefore, some instructions place the contents of an addressed general purpose register into the high-order positions (bits 0-31) of the register bus latch, and other instructions place the contents of an addressed general purpose register into the low-order positions (bits 32-63) of the register bus latch, or one general purpose register may be placed into each half of the register bus latch depending upon the instruction being performed.

The output from the register bus latch gates data to the J register and the M register. The transfer from the register bus latch to the J register is either a straight transfer or a right eight shift transfer as shown in Figure 5079. With the left four shift on, the transfer from the J register to the register bus latch and the right eight shift from the register bus latch to the J register, the next digit is placed into position for gating to the multiplier digit latches to allow the next multiplier to be decoded during multiply instructions. The left four

shift on the transfer of the J register to the register bus latch combined with the straight transfer from the register bus latch to the J register during divide operations allows shifting the partial quotient left four bit positions to provide space for the next quotient digit to be gated into bits 59-63 of the J register.

During floating-point operations, the exponent of the floating-point data word is located in bits 56-63 of the floating-point register. During the transfer of the contents of the floating-point register to the register bus latch, the exponent is transferred to positions 56-63 of the register bus latch and the fraction is transferred to positions 0-55 of the register bus latch in a straight transfer. However, during the transfer of the register bus latch register to the J register, during instruction time of the floating-point instruction, the exponent and fraction are shifted right eight bit positions in a closed loop operation. This operation is shown as a RBL to J RT 8 Ring Rel J operation in Figure 5079. At the end of this transfer, the fraction is located in bit positions 8-63 of the J register, and the exponent is located in bits 0-7.

In order to set a bit into any position of the register bus latch, the AND-OR circuit labeled +RBL 00 is considered in the following way: the input AND circuit is considered as a -OR and the output OR is considered as a -AI. This method of looking at the AND/OR circuit for RBL 00 applies to all bit positions of the register bus latch.

For discussion purposes, lets assume a bit is transferred from a floating-point register to the register bus latch. Considering the inputs to the AND circuits (-OR's), the following conditions are present:

1. -J04 to RBL 00 is positive.
2. -K00 to RBL 00 is positive.
3. -FLP bit 0 to RBL is negative.
4. -Gen Reg 00 to RBL 00 is positive.
5. -Lock RBL 0-7 is positive.
6. +Lock RBL 0-7 is negative.
7. The latch back line is positive.

With these conditions present at the input to the +RBL 00 latch, one input of each of the AND circuits (-OR's) is negative.

Next, considering the OR circuit (-AI); both inputs to the OR (-AI) circuit are negative; therefore, a positive output is obtained. The positive output is inverted by the inverter and a negative level is obtained on the latch back line. When the -Lock RBL 0-7 line and the +Lock RBL 0-7 line return to the indicated conditions, the bit transferred from the floating-point register to the register bus latch is locked into the register bus latch. The register bus latch is reset when the lock line conditioning levels are reversed and a bit is not transferred to the register bus latch from another register. This condition exists on the following clock cycle because the -Lock RBL 0-7 line and the +Lock RBL 0-7 line are L running clock pulses; therefore, data are retained in the register bus latch for one clock cycle.

From Figure 5079 it is seen that any condition which gates the contents of the register bus latch to the J register also conditions the line labeled +Rel J. The only other condition which release the J register so that data is placed into it is the J advance pulse which is conditioned when data are returned from storage to the storage bus out register then to the J register.

Return Address Registers

- The return address registers are six position registers used to route data and error indications to their proper destinations.
- The two return address registers (X and Y) are used alternately on 2365 storage selections, and are necessary because of overlapped storage cycles.
- The positions in each register are A, B, J, Channel, Invalid, and Diagnose.
- The X-Y binary trigger controls the input gating to the return address registers.
- The W-Z binary trigger controls the output gating from the return address registers.

The X and Y return address registers are used alternately on 2365 storage selections to route data and error indications to the proper destinations. Two registers, used alternately, are necessary because of overlapped storage cycles. Each register has six positions (no parity).

The bus control unit sets one or more positions in a return address register wherever a select is sent to storage; the set positions subsequently route the storage "advance" pulse to the proper destination. Return address register positions are:

A: Set on even-word instruction buffer fetches; returns double-word to the A register.

B: Set on odd-word instruction buffer fetches; returns double-word to the B register.

J: Set on operand fetches; returns double-word to the J register.

Channel: Set on channel fetches and stores; returns storage advance signal to channel.

Invalid: Set when the bus control unit receives an invalid address (one outside the available storage) on central processing unit fetches on channel fetches or stores. Result: storage is cancelled and storage advance is routed to set A invalid, B invalid, J invalid, or to send the invalid address signal to channel.

Diagnose: Set on diagnose instructions to gate fetched word into the maintenance control word (MCW) register.

The X-Y binary trigger controls input gating to the return address registers (Figure 5080). When a storage selection is made, the "positive select out" signal and a B running clock releases the return address register pointed-to by the X-Y trigger. The "positive select out" signal is delayed to switch X-Y after the register has been locked.

The W-Z binary trigger controls output gating from the return address registers. The "advance" from storage is routed according to the positions set in the return address register pointed-to by W-Z. The W condition of W-Z points-to the X return address register; the Z condition of W-Z points-to the Y return address register. The "advance" from storage is delayed to switch W-Z after the undelayed "advance" has been routed according to the return address register positions set.

The return address register mechanism operates on all storage selections; however, no return address register positions are set on central processing unit store operations.

Storage Address Register (SAR)

- SAR is a 24-bit plus three parity-bit register.
- SAR holds CPU storage addresses.
- SAR is set from AA or incrementer.
- SAR feeds the address OR.

The storage address register is a 24-bit plus three parity-bit register with positions 0-23 (Figure 5081). The central processing unit (instruction unit and execution unit) puts the desired address into the storage address register for all central processing unit storage operations. The storage address register is set from either the addressing adder or the incrementer. The addressing adder sets the storage address register when an effective address is calculated. The incrementer sets the storage address register for instruction counter fetches and for multiple fetch operations. Notice that the addressing adder bits 8-31 set storage address register bits 0-23.

The only output of the storage address register is to the addressing OR which gates either the address in the storage address register or the address on the channel address bus (CAB) to main storage.

Each bit position of the storage address register is a polarity hold device release by a special early L clock pulse. Input bits to the storage address register are not necessarily good at release time. The release is timed to allow the storage address to flow from the addressing adder or the incrementer through the storage address register to the address OR. All bits are good at lock time (end of the release pulse) so that the storage address register holds the storage address until the address is set into a memory address register in storage.

A separate storage address register release (not shown in the figure) is provided for fault location test scan-in. During scan-in, the data input is through the incrementer.

SAR Duplicate

- Duplicate SAR positions 0-6 and 19-20.
- Duplicate SAR is set in parallel with SAR.
- The output of duplicate SAR are good sooner than SAR bits.
- Duplicate SAR outputs are used to direct a select pulse to the proper storage unit.

Nine positions of the storage address register are duplicated in the duplicate storage address register (dup SAR). The positions are 0-6 and 19-20 (Figure 5082). The duplicate storage address register is used by the bus control unit to generate a storage select pulse and is physically closer to the bus control unit selection circuits than the storage address register. By using a separate register for the selection bits, the bus control unit receives these bits prior to the time when they would otherwise be available. This fast decoding of selection bits is necessary because the central processing unit may set the storage address register on the same cycle that a storage request is made; if the requested storage is not busy and there is no interference from channels, the bus control unit selects storage on this same cycle.

Bits 5 and 6 are included in the duplicate storage address register because of the address switching maintenance feature. This feature allows switching of address bits to help isolate core storage failures. Address switching is described in the bus control unit theory of operation section.

The bit positions of the duplicate storage address register are two-way input polarity hold devices identical to the storage address register bit positions (Figure 5081).

Storage Bus In (SBI) Latch Register

- The SBI is 72 bits wide.
- The SBI is set from the K register or channel SBI.
- The SBI feeds each 2365 storage unit.

The storage bus in latch register holds the 64-bit plus eight parity-bit word enroute to storage on store operations (Figure 5084). The storage bus in latch is set from the K register on central processing unit stores and from the channel storage bus in on channel stores. Outputs of the storage bus in are to the 2365 storage units A, B, C, and D (J75). Another storage bus in latch register output bus is routed to the large capacity storage (LCS) units if the system has one or more of these storages.

Bit positions of the storage bus in latch register are two-input polarity hold devices released early in the cycle following a store select cycle.

The inhibit storage bus in trigger (Figure 5084) insures that the storage bus in cannot be changed again on the cycle following a release pulse (SBI data is always good for a minimum of two machine cycles).

Storage Bus Out (SBO) Latch Register

- The SBO is a 72-bit register.
- The SBO buffers fetched storage words.
- The inputs to the SBO are from storage A, B, C, D, and PDU pre-OR.
- The outputs of the SBO are to A, B, J, and channel SBO.

The storage bus out latch register holds the 64-bit plus eight parity-bit word enroute from storage to the destination register on fetch operations (Figure 5085). Five inputs are OR'ed at the storage bus out latch: storage A SBO, storage B SBO, storage C SBO, storage D SBO, and the output of the system console (PDU) pre-OR. The power distribution unit pre-OR output contains either the system console data keys (panel keys) or the storage bus out from optional large capacity storage units.

The storage bus out sets either the J register (operand fetches), the A or B register (instruction counter fetches), or is fed to a channel on the channel storage bus out. The destination of the storage bus out latch data is controlled by the return address circuits which send an "advance" pulse to the appropriate receiving register.

Bit positions within the storage bus out latch are polarity hold devices released at the late LR time following the rise of the "advance" signal from storage.

Shift Counter Register

- The shift counter is an eight position plus parity register.
- The shift counter has inputs from the AEOB, FLT, H register and forced inputs.
- It has outputs to the AE, VFL, and shift counter decoder.

The shift counter (SC) register is an eight bit, plus parity, register used to hold the shift amount for shift instructions, the iteration count for multiply instructions, normalization shift cycle count decoding during add, subtract, and compare and variable field length logical instructions.

Figure 5086 shows a block diagram of the data flow into and out of the shift counter, the input gating for the shift counter, the shift counter, and the out-gating for the shift counter. The shift counter decoder is shown on Figure 5087. The shift counter register inputs are the exponent adder output bus, the L register bits 16-23, H register bits 18-20, and forced inputs to shift counter register positions 1-6. Bits 0-7 of the exponent adder are placed in the shift counter register during instructions requiring the contents of the shift counter to be decremented (example: during shift operations). The input from the L register bit positions 16-23 are used during the fault location test (FLT) mode of operation; these bits are set into the shift counter by the line labeled + Scan Rel Sft Ctr conditioning the AO labeled - Rel SC on Figure 5086. This scan release shift ctr is conditioned by the same scan word 12 and scan pulse 2 that release the exponent register in Figure 5056. The inputs from the H register, bits 18-20, are set into the shift counter register bits 4-6 when a shift instruction is decoded. The shift counter register bits 1-6 are forced for iteration counts during fixed-point instructions, floating-point instructions and on convert instructions.

The output of the shift counter register is gated to the true/complement input of the exponent adder, the shift counter decoder, Figure 5086, and the addressing adder bits 24-27 for variable field length instructions. The contents of the shift counter register are transferred to the true/complement input of the exponent adder and decremented by an amount equal to the value of the shift of the iteration cycle taken. The shift counter, bits 2-7, output provides

inputs to the shift counter decoder which determines the magnitude of the shift during normalization cycles or the remaining iterations to be taken. The shift counter output to the addressing adder is used to set the variable field length address during convert binary and convert decimal instructions.

In order to set a bit in the shift counter register, Figure 5086, the input AND circuits for any bit are considered as -OR's and the output OR circuits are considered as -AI circuits. To set a bit into shift counter register position labeled +SC bit 1 in Figure 5086 from the exponent adder, the following conditions must exist:

1. -AE Bit 1 to SC is negative.
2. -L Scan Bit 17 is positive.
3. -Force SC 1 is positive.
4. +Release Shift Ctr. is positive.
5. -Release Shift Ctr. is negative.
6. The latch back line is positive.

With these conditions present at the inputs to the AND circuits (-OR's), both inputs to the OR (-AI) are negative, thus, conditions being met, the output of the OR (-AI) is positive. The invert block inverts the level and conditions the latch back line to retain the bit in the shift counter register.

When the +Release Shift Ctr and the -Release Shift Ctr lines are returned to the opposite states, the bit is locked in the latch register. The bit is retained in the shift counter register until the +Release Shift Ctr and the -Release Shift Ctr lines are conditioned again.

INDEX

FUNCTIONAL UNITS

SYSTEMS INTRODUCTION

COMMENT SHEET

2075 PROCESSING UNIT —— VOLUME I

FIELD ENGINEERING MANUAL OF INSTRUCTION, FORM 223-2872-1

FROM

NAME _____ OFFICE/DEPT NO. _____

CITY/STATE _____ DATE _____

To make this manual more useful to you, we want your comments: what additional information should be included in the manual; what description or figure could be clarified; what subject requires more explanation; what presentation is particularly helpful to you; and so forth.

FOLD FOLD

CUT ALONG LINE

FOLD FOLD

How do you rate this manual: Excellent _____ Good _____ Fair _____ Poor _____

Suggestion from IBM Employees giving specific solutions intended for award considerations should be submitted through the IBM Suggestion Plan.

NO POSTAGE NECESSARY IF MAILED IN U.S.A.

FOLD ON TWO LINES, STAPLE, AND MAIL

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 81

POUGHKEEPSIE, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
P. O. BOX 390
POUGHKEEPSIE, N. Y. 12602

ATTN: FE MANUALS, DEPARTMENT B95

DO NOT REMOVE FROM 483 MSA

FOLD

FOLD

CUT ALONG LINE

STAPLE

IBM FE Supplement

System/Unit	2075
Re: Form No.	223-2872-1
This Supplement No.	S26-7033
Date	January 1968
Previous Supplement Nos.	None

This supplement revises and updates Volume 1 of the Field Engineering Manual of Instruction on the IBM 2075 Processing Unit, Form 223-2872-1. This supplement incorporates the floating point changes released under Engineering Change EC 705 848E.

Incorporate this supplement by replacing Title page, Preface page, pages 66, 67, 68, and 91 with corresponding pages attached to this notice.

Changes to text are indicated by a vertical bar to the left of the affected material. Revised diagrams are identified by a bullet (•) to the left of the figure caption. (In addition, changes that are not readily apparent are indicated by a vertical bar to the left of the changed area.)

File this cover letter at the back of the publication. It will then serve as a record of the changes received and incorporated.

International Business Machines Corp., Product Publications Dept., Neighborhood Road, Kingston, N.Y. 12401

PRINTED IN U.S.A.