IBM Field Engineering

Theory of Operation

(Manual of Instruction)

System/360 Model 44

Floating Point Feature

IBM® Field Engineering

Theory of Operation

(Manual of Instruction)

System/360 Model 44

Floating Point Feature

PREFACE

This volume describes the Floating Point Feature of the IBM 2044 Processing Unit, the processor for the IBM System/360 Model 44. Chapter 1 of the volume introduces the concepts of floating-point arithmetic and defines its application within the hexadecimal system used in the IBM 2044. Chapter 2 describes the functional parts of the floating point feature and the interaction of the feature with the basic central processing unit. Chapter 3 describes in detail the operation of the floating-point instruction set.

The manual assumes knowledge of the System/360 as described in IBM System/360 Principles of Operation, Form A22-6821.

The volumes that constitute the IBM Field Engineering Theory of Operation manual for the IBM System/360 Model 44, and their form numbers, are:

System/360 Model 44, Introduction and Functional Units, Form Y33-0001: Gives a general introduction to digital computers and computing techniques, defines the relationship of the IBM 2044 to the System/360 and describes the various parts which form the processing unit.

System/360 Model 44, Principles of Operation - Processing Unit, Form Y33-0002: Describes the operation of the instructions for the accelerator and basic machine (other than floating-point instructions), and program and machine interrupts.

System/360 Model 44, Principles of Operation - Channels, Form Y33-0003: Describes the Common Channel area, the Multiplexor Channel 0 and the High Speed Multiplexor Channel.

These volumes are referenced in other volumes by the main element of their titles.

Reference is also made in these volumes to the following associated manuals:

System/360 Model 44, Single Disk Storage Drive, Form Y33-0006: Gives an introduction to the operation of the control unit and describes in detail the functional parts and the operations that may be performed.

Field Engineering Maintenance Manual (FEMM), IBM System/360 Model 44, Form Y33-0007: Contains information for servicing the 2044 Processing Unit.

Field Engineering Maintenance Diagrams (FEMD), IBM System/360 Model 44, Volume 2, Form Y33-0008: Contains maintenance information in the following categories: Data Flow Charts, Flow Charts, Timing Charts, MAP's.

Other related manuals that describe units used in the System/360 Model 44 are:

Field Engineering Manual of Instruction (FEMI), 1052 Adapter, Form 223-2808.

Field Engineering Maintenance Manual (FEMM) Single Disk Storage/Direct Access, Form Y26-3663.

First Edition

This manual makes obsolete Field Engineering Theory of Operation, System/360 Model 44, Form Z33-0005-0.

The manual is written basically to Engineering Change Level 390049 and in some cases anticipates Engineering Change Level 390063. Significant changes or additions to the information in the manual will be covered in subsequent revisions of FE supplements.

This publication was prepared by IBM European Laboratories, Product Publications. A form is provide at the back of this manual for reader's comments. If the form has been removed, comments may be addressed to: IBM Corporation, FE Manuals, Dept. B96, PO Box 390, Poughkeepsie, N.Y. 12602

CONTENTS

CONTENTS (continued)

ILLUSTRATIONS

ABBREVIATIONS

The following abbreviations are used in this manual.

ALS	Arithmetic and Logic Section
CPU	Central Processing Unit
Dec	Decimal
EQ	Equal
EXOR	Exclusive OR
FP	Floating-Point
FPR	Floating-Point Register
GPR	General-Purpose Register
Hex	Hexadecimal
HI	High
HW	Hardware (funnel)
I-Fetch	Instruction-Fetch
LO	Low
MSLS	Medium-Speed Local Storage
Op	Operation
Opnd	Operand
PSW	Program Status Word
SDR	Storage Data Register
T/XC	True/Criss-Cross

PROPERTIES OF NUMBERS

Scientific and engineering applications demand extensive handling of numbers with values that vary from the very large to the very small. To allow for the handling of these numbers, and to obtain results with sufficient precision, registers in computers would need an excessively large capacity.

Since register size is generally limited, and the result of a calculation using very large or very small numbers is usually meaningful only in the significant digits, the floating-point (FP) system of arithmetic is applied to the calculations.

This system of representing numbers is, in effect, a shorthand method which allows the significant digits of a number to be used in a calculation and the magnitude of the number to be held separately from the calculation.

In decimal notation this type of operation is familiar. For example, the timing of computer operations is expressed in milliseconds, microseconds or nanoseconds. This expression produces a number in which extraneous zeros are removed and only relevant significant digits remain. Thus an operation that takes 0.000015 seconds is expressed as 15×10^{-6} seconds (15 microseconds) and an operation that takes a much shorter time, such as 0.000000015 seconds can be expressed, with the same ease, as 15×10^{-9} seconds (15 nanoseconds).

Similarly, large numbers may be expressed in this manner. For example, light travels approximately 6,000,000,000,000 miles in one year that is, 6×10^{12} miles.

In the foregoing examples, the significant digits and the power of ten to which they are raised may be expressed in a smaller number of positions than the original number.

It is usual in floating-point representation to place place the point immediately to the left of the most-significant digit. Thus 15 microseconds becomes $.15 \times 10^{-4}$ seconds and 15 nanoseconds becomes $.15 \times 10^{-7}$ seconds, while 6,000,000,000,000 miles becomes $.6 \times 10^{13}$ miles.

The number of significant digits held determines the accuracy of the calculation and is called the "precision".

SYSTEM/360 FLOATING-POINT NUMBERS

- Floating-point numbers are recorded in the hexadecimal number system.
- The significant digits are recorded as hexadecimal fractions.
- The exponent is the value to which the base 16 is raised.
- The characteristic can be a number from 0 to 127. The characteristic has a value of exponent plus 64, giving a range of exponents of - 64 to + 63.
- Floating-point numbers are recorded in fixed-length format:
  For short precision in a single word.
  For long precision in a double word.
- Bit 0 of either format denotes the sign of the fraction.
- Bits 1 to 7 of either format denote the characteristic.
- Bits 8 to 31 for short precision or bits 8 to 63 for long precision denote the fraction.
- Negative fractions are always carried in true form

The format in which floating-point numbers are held is shown in Figure 1-1.

Chapter 1, Introduction and Functional Units, Form Y33-0001, shows that floating-point numbers are not expressed with a base of 10 as in the decimal system but use, instead, the hexadecimal system with a base of 16. Thus, numbers in System/360 format are expressed as a fraction to the base 16 times a power of 16, whereas decimal representation is a fraction to the base 10 times a power of 10.

The format chosen for System/360 floating-point data allocates seven binary digits to express

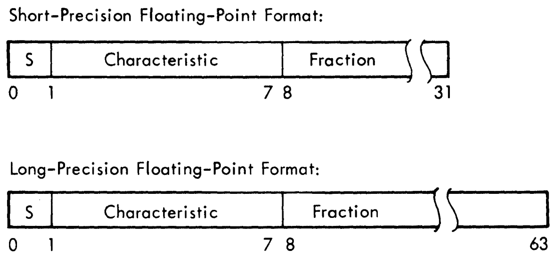

Short-Precision Floating-Point Format:
S
Characteristic
Fraction
0
1
7
8
31
Long-Precision Floating-Point Format:
S
Characteristic
Fraction
0
1
7
8
63

Figure 1-1. Floating-Point Format

the power or exponent. The allocation provides an allowable range of 0 to 127.

Since provision must be made for the representation of either positive or negative exponents, a system of expressing the exponent in "excess 64 arithmetic" is used. This is done by adding + 64 to each exponent and the result or characteristic is expressed in the seven binary digits allocated. This method allows both positive and negative exponents to be expressed as a positive binary number. Thus the characteristic is contained in seven digits allowing a range of 0 through + 64 to + 127, representing an exponent reange of - 64 through 0 to + 63 (as characteristic = exponent = + 64).

The format chosen for the fraction allows either 6 or 14 hexadecimal digits in the fraction and thus the range of numbers is from a fraction × 16^{63} to a fraction × 16^{-64}; the precision of expressing the fraction may range from the short-precision 6 hexadecimal digits to the long-precision 14 hexadecimal digits.

The sign of the floating-point number is held separately from both the fraction and characteristic and, as usual, is a single digit with a 1 representing a minus and a 0 representing a plus. The fraction is always in true form.

The following are sample normalized short floating-point numbers. The last two rows represent the smallest and the largest positive normalized numbers.

1.0	$=+1/16 \times 16^{1}$	=0 1000001 0001 0000 0000 0000 0000 0000
0.5	$=+8/16 \times 16^{0}$	=0 1000000 1000 0000 0000 0000 0000 0000
1/64	$=+4/16 \times 16^{-1}$	=0 0111111 0100 0000 0000 0000 0000 0000
0.0	$=+0 \times 16^{-64}$	=0 0000000 0000 0000 0000 0000 0000 0000
-15.0	$=-15/16 \times 16^{1}$	=1 1000001 1111 0000 0000 0000 0000 0000
5.4×10^{-79}	$=+1/16 \times 16^{-64}$	=0 0000000 0001 0000 0000 0000 0000 0000
7×10^{75}	$=(1-16^{-6}) \times 16^{63}$	=0 1111111 1111 1111 1111 1111 1111 1111

Hexadecimal and Decimal Conversion (Examples)

Refer to the Appendix of the manual for a Powers of 16 table.

Conversion Example 1

Convert the hexadecimal (hex) number 1B2.3C to decimal (dec) notation.

$$\text{Result: } (1 \times 16^{2}) + (11 \times 16^{1}) + (2 \times 16^{0}) + (3 \times 1/16^{1}) + (12 \times 1/16^{2})$$
$$= 256 + 176 + 2 + 0.1875 + 0.0469$$
$$= 434.2344$$

Conversion Example 2

Convert the decimal number 434.2344 to hexadecimal notation.

a. Convert 434 to hexadecimal.

434 ÷ 16	= 27 and a remainder of $\underline{2}$
27 ÷ 16	= 1 and a remainder of $\underline{B}$ (11)
1 ÷ 16	= 0 and a remainder of $\underline{1}$
434 (dec)	= 1B2 (hex)

b. Convert 0.2344 to hexadecimal.

0.2344 x 16	= $\underline{3}$.7504
0.7504 x 16	= $\underline{12}$.0064
0.0064 x 16	= $\underline{0}$.1024
0.2344 (dec)	= 0.3C0 (hex)

c. Combine the results of the above two steps.

434.2344 (dec) = 1B2.3C (hex)

Conversion Example 3

Convert the decimal number 2277.302 to a System/360 short-precision floating-point number.

a. Convert 2277 to hexadecimal.

2277 ÷ 16	= 142 and a remainder of $\underline{5}$
142 ÷ 16	= 8 and a remainder of $\underline{E}$ (14)
8 ÷ 16	= 0 and a remainder of $\underline{8}$
2277 (dec)	= 8E5 (hex)

b. Convert 0.302 to hexadecimal.

0.302 x 16	= $\underline{4}$.832
0.832 x 16	= $\underline{13}$.312
0.312 x 16	= $\underline{4}$.992
0.302 (dec)	= 0.4D5 (hex)

c. Combine results of steps (a) and (b) and express as a fraction multiplied by a power of 16 (exponent).

2277.302 (dec) = 8E5.4D5 (hex)
$= 0.8E54D5 \times 16^{3}$ (hex)

d. Convert the exponent to binary representation and add to 64 to form the characteristic.

Characteristic = 1000011, that is (64 + 3)

e. Assemble sign (bit 0), characteristic (bits 1 to 8) and fraction (bits 9 to 31) to form System/360 short-precision floating-point number.

Result = 0 1000011 1000 1110 0101 0100 1101 0101

Conversion Example 4

Convert to decimal notation the following System/360 short-precision floating-point numbers.

0 1000010 0001 0101 1100 0000 0000 0001

a. Obtain the exponent in decimal form.

Characteristic = 66
Exponent = 66 - 64 = +2

b. Express each unit of the fraction as a decimal number times a power of 16.

Fraction = $(1 \times 16^{-1}) + (5 \times 16^{-2}) + (12 \times 16^{-3}) + (1 \times 16^{-6})$

c. Multiply each unit of the fraction by 16 to the power of the exponent. (Add exponents.)

Result = 16^{2} x Fraction
Result = $(1 \times 16^{1}) + (5 \times 16^{0}) + (12 \times 16^{-1}) + (1 \times 16^{-4})$

d. Calculate result (using if necessary the Powers of 16 table).

Result = 16 + 5 + (12 x 0.0625) + (0.000015)
= 21.750015 (to 6 decimal places)

e. Add sign.

Result = + 21.750015

Conversion Example 5

Convert to decimal notation the following System/360 short-precision floating-point number.

1 0111110 1111 1001 0000 0001 0000 0000

a. Obtain the exponent in decimal form.

Characteristic = 62
Exponent = 62 - 64 = -2

b. Express each unit of the fraction as a decimal number times a power of 16.

Fraction = $(15 \times 16^{-1}) + (9 \times 16^{-2}) + (1 \times 16^{-4})$

c. Multiply each unit of the fraction by 16 to the power of the exponent. (Add exponents.)

Result = 16^{-2} x Fraction
= $(15 \times 16^{-3}) + (9 \times 16^{-4}) + (1 \times 16^{-6})$
= 0.003 662 109 375 + 0.000 137 101 + 0.000 000 059 604
= 0.003 799 498 180
= 3.7995×10^{-3}

d. Add sign.

Result = -3.7995×10^{-3}

Terms Used in System/360 FP Arithmetic

Characteristic: The exponent plus 64, expressed as a seven-bit binary number.

Exponent: The number (or power) by which the base (or radix) 16 is raised.

FP Number: The floating-point number consists of a sign bit, a seven-binary-bit characteristic and either a 6- or 14- hexadecimal-digit fraction.

Fraction: The fraction is either 6 or 14 hexadecimal digits, numbered from 0 to 5 or 0 to 13 respectively. The radix point is in front of the leftmost digit.

Guard Digit: A hexadecimal digit which is preserved in the (hexadecimal) digit-6 position of short-precision results, to take part in the post-normalizing process if it is required by the instruction.

Hexadecimal: A number system using a base (or radix) of 16.

Normalization: The process of shifting left the hexadecimal digits of the fraction portion of an un-normalized number until the leftmost hexadecimal digit becomes significant. The characteristic is reduced by one for each shift left executed. Note that the first three binary digits may be zero, provided that the first hexadecimal digit is significant.

Pre-normalization: The process of normalizing operands prior to performing arithmetic operations.

Post-normalization: The process of normalizing results.

Precision: The accuracy with which the fraction is expressed and a measure of the length of the operands and results. Short-, variable- and long-precision operands contain 6, 8 to 14, and 14 hexadecimal digits respectively.

Radix Point: A general term for use with any number system to describe a function equivalent to that of the decimal point in decimal notation.

Un-normalized Number: A number in which the fraction digits contain one or more hexadecimal zeros prior to the first significant hexadecimal digit.

Variable-Precision: A preset precision (determined by a console switch) which defines the number of hexadecimal digits which are to be used in calculations. The precision switch on System/360 Model 44 allows a precision of 8, 10, 12 or 14 hexadecimal digits, and causes long operands to be truncated to the defined precision on the cycle when low-order digits are fetched.

RULES FOR FLOATING-POINT ARITHMETIC

Multiply

- The exponents are added
- The fractions are multiplied.
- If necessary, the result is normalized.

Example (in decimal notation):

$(0.25 \times 10^4) \times (0.33 \times 10^{-7})$	
Add exponents:	$4+(-7)=-3$
Multiply fractions:	$0.25 \times 0.33 = 0.0825$
Result:	0.0825×10^{-3}
Normalized:	0.825×10^{-4}

This sequence is executed with one instruction.

Divide

- The divisor exponent is subtracted from the dividend exponent.
- The fractions are divided.
- If necessary, the result is normalized.

Example (in decimal notation):

$(0.222 \times 10^{-2}) \div (0.4 \times 10^{-5})$	
Subtract exponents:	$-2-(-5) = +3$
Divide fractions:	$0.222 \div 0.400 = 0.555$
Result:	0.555×10^3

Normalization not necessary (in this example).

Shifting

- Shift Left: for each digit position shifted, the value of the exponent is reduced by one.
- Shift Right: for each digit position shifted, the value of the exponent is increased by one.

Examples (in decimal notation);

a.	Shift-left-three:	$.000073 \times 10^7$
	Result =	$.073000 \times 10^4$
b.	Shift-right-two:	$.560000 \times 10^6$
	Result =	$.005600 \times 10^8$

Add and Subtract

- The number with the smaller exponent is shifted to the right and the value of its exponent increased until the values of both exponents are equal.
- The fractions are added (or subtracted).
- The exponent remains unchanged.
- If specified, the result is normalized.

Example (using decimal notation):

Add 0.00217×10^{10} and 0.8×10^3

1. The number with the smaller exponent is shifted right until exponents become equal.

   0.00217×10^{10}

   $0.00000008 \times 10^{10}$

2. The fractions are added (or subtracted).

   $0.00217008 \times 10^{10}$

3. If specified, the result is normalized.

   0.217008×10^8

FLOATING-POINT EXCEPTIONS

- Normal exceptions apply to FP instructions and data:
  - Operation.
  - Protection.
  - Addressing.
  - Specification (Basic).
- Additional exception conditions are used with the FP feature:
  - Specificiation (floating-point).
  - Exponent overflow.
  - Exponent underflow.
  - Significance.
  - Floating-point divide.

Exceptional instructions, data, or results cause a program interrupt. When the interrupt occurs, the current Program Status Word (PSW) is stored as an old PSW, and a new PSW is obtained. The interrupt code in the old PSW identifies the cause of the interrupt. The following exceptions cause a program interrrupt in floating-point arithmetic.

1. Operation Exception: Occurs when the floating-point feature is not installed, and an attempt is made to execute a floating-point instruction.

The instruction is suppressed and the condition code, data in registers, and storage data are unchanged.

2. Protection Exception: Occurs if the storage protect feature is installed and the storage key of a store-instruction location does not match the protection key in the PSW. The operation is suppressed and the condition code, data in registers, and storage data are unchanged.
3. Addressing Exception: Occurs when an address designates a location outside the available storage for the installed system. The operation is terminated and the resultant data and the condition code, if affected, are unpredictable and should not be used for further computation.
4. Specification Exception: Occurs when a short operand is not located on a 32-bit boundary, a long operand is not located on a 64-bit boundary, or a floating-point register address other than 0, 2, 4 or 6 is specified. The instruction is suppressed, and the condition code, data in registers, and storage data are unchanged.
5. Exponent Overflow Exception: Occurs when the result exponent of an addition, subtraction, multiplication or division overflows (that is, is greater than +63) and the result fraction is not zero. The operation is terminated; the resultant data is unpredictable and should not be used for further computation. The condition code is set to binary 11 for addition or subtraction and remains unchanged for multiplication and division.
6. Exponent Underflow Exception: Occurs when the result exponent of an addition, subtraction, multiplication or division underflows (that is, is less than -64) and the result fraction is not zero. A program interrupt occurs if the exponent underflow mask bit is one. The operation is completed by replacing the result with a true zero. The condition code is set to 00 for addition and subtraction and remains unchanged for multiplication and division. The state of the mask bit does not affect the result.
7. Significance Exception: Occurs when the result fraction of an addition or subtraction is zero. A program interrupt occurs if the significance mask bit is one. The mask bit also affects the result of the operation. When the significance mask bit is zero, the operation is completed without further change to the characteristic of the result. In either case, the condition code is set to 00.
8. Floating-Point Divide Exception: Occurs when a division by a number with zero fraction is attempted. The division is suppressed and the condition code, data in registers, and storage data are unchanged.

INSTRUCTIONS AND INSTRUCTION FORMAT

- RR and RX formats are used for FP instructions.
- The R1 and R2 fields must specify floating-point register 9, 2, 4 or 6.
- For long-precision instructions the actual precision is determined by the setting of the variable-precision switch.

The standard RR and RX formats are used in the floating-point instruction set.

For floating-point instructions, R1 and R2 fields must specify one of the Floating-Point Registers (FPR's), 0, 2, 4 or 6, that are provided with the feature. If either R1 or R2 specifies other than one of these FPR's, a specification exception and a program interrupt occur.

In common with the standard instruction set, results replace the first operand, except for storing operations, when the second operand is replaced.

Figure 1-2 lists the floating-point instructions, their mnemonics, operation (op) codes and formats, and the exceptions that cause a program interrupt.

VARIABLE-PRECISION FEATURE

- Applies to long-precision instructions.
- Precision of 8, 10, 12 or 14 digits can be selected.
- Operands are truncated on operand-fetch.
- Truncated positions are replaced by zeros.
- Results are stored in long-precision format (without truncation) regardless of the setting of the variable-precision switch.
- Results are truncated when they are next fetched.
- Truncation does not occur on console display.

The Model 44 user may specify that long-precision instructions shall be executed with less than 14 digits of precision. Specifying is made by means of a console rotary switch marked 'floating-point precision'. This switch has four positions (14, 12, 10, 8) which refer to the number of hexadecimal digits of the fraction that are to be processed by long-precision instructions.

When the switch is set to 14, the full 14-digit (56-bit) fraction is processed and results of floating-point instructions are identical to those obtained on other System/360 Models with floating-point facili-

Name	Precision	Type	Mnemonic	Op Code	Exceptions
Load	Long *	RR	LDR	28	S
Load	Long *	RX	LD	68	A, S
Load	Short	RR	LER	38	S
Load	Short	RX	LE	78	A, S
Load and Test	Long *	RR, C	LTDR	22	S
Load and Test	Short	RR, C	LTER	32	S
Load Complement	Long *	RR, C	LCDR	23	S
Load Complement	Short	RR, C	LCER	33	S
Load Positive	Long *	RR, C	LPDR	20	S
Load Positive	Short	RR, C	LPER	30	S
Load Negative	Long *	RR, C	LNDR	21	S
Load Negative	Short	RR, C	LNER	31	S
Add Normalized	Long *	RR, C, N	ADR	2A	S, U, E, LS
Add Normalized	Long *	RX, C, N	AD	6A	A, S, U, E, LS
Add Normalized	Short	RR, C, N	AER	3A	S, U, E, LS
Add Normalized	Short	RX, C, N	AE	7A	A, S, U, E, LS
Add Un-normalized	Long *	RR, C	AWR	2E	S, E, LS
Add Un-normalized	Long *	RX, C	AW	6E	A, S, E, LS
Add Un-normalized	Short	RR, C	AVR	3E	S, E, LS
Add Un-normalized	Short	RX, C	AV	7E	A, S, E, LS
Subtract Normalized	Long *	RR, C, N	SDR	2B	S, U, E, LS
Subtract Normalized	Long *	RX, C, N	SD	6B	A, S, U, E, LS
Subtract Normalized	Short	RR, C, N	SER	3B	S, U, E, LS
Subtract Normalized	Short	RX, C, N	SE	7B	A, S, U, E, LS
Subtract Un-normalized	Long *	RR, C	SWR	2F	S, E, LS
Subtract Un-normalized	Long *	RX, C	SW	6F	A, S, E, LS
Subtract Un-normalized	Short	RR, C	SUR	3F	S, E, LS
Subtract Un-normalized	Short	RX, C	SU	7F	A, S, E, LS
Compare	Long *	RR, C	CDR	29	S
Compare	Long *	RX, C	CD	69	A, S
Compare	Short	RR, C	CER	39	S
Compare	Short	RX, C	CE	79	A, S
Halve	Long *	RR	HDR	24	S
Halve	Short	RR	HER	34	S
Multiply	Long *	RR, N	MDR	2C	S, U, E
Multiply	Long *	RX, N	MD	6C	A, S, U, E
Multiply	Short	RR, N	MER	3C	S, U, E
Multiply	Short	RX, N	ME	7C	A, S, U, E
Divide	Long *	RR, N	DDR	2D	S, U, E, FK
Divide	Long *	RX, N	DD	6D	A, S, U, E, FK
Divide	Short	RR, N	DER	3D	S, U, E, FK
Divide	Short	RX, N	DE	7D	A, S, U, E, FK
Store	Long *	RX	STD	60	P, A, S
Store	Short	RX	STE	70	P, A, S

Legend:

A	Addressing exception
C	Condition code is set
E	Exponent-overflow exception
FK	Floating-point divide exception
LS	Significance exception
P	Protection exception
S	Specification exception
U	Exponent-underflow exception
*	Precision determined by FP precision switch
N	Normalized operation

Figure 1-2. Floating-Point Instruction Details

ties. At the 12, 10 and 8 settings, all long-precision instructions of the floating-point set operate on fractions that are truncated to 12, 10 or 8 digits respectively; these results will normally differ in precision, and possibly in significance, from those of other models with floating-point facilities. Short-precision instructions are unaffected by the switch.

Truncation occurs at the ABC funnel at the time that operands are fetched from core storage. Since the least-significant halves of the FPR's are located in extension storage, the least-significant half of all floating-point operands (whether located in main storage or in an FPR) must pass through the Storage Data Register (SDR) and ABC funnel. Truncation applies in this way to all long-precision operands, whether fetched for RR or for RX instructions.

Once truncation has occurred, the operand(s) may be extended with zeros to a full 56-bit fraction, and processed as such; this processing is amplified in the sections of the manual concerning specific instructions. It follows that where shifting has occurred (to align fractions whose exponents differ) non-zero results in the low-order bytes may be placed in the result register, and manual display of FPR's may show a full-length non-zero fraction, regardless of the setting of the variable-precision switch.

In order to fetch the contents of an FPR, one of the 22 long-precision FP instructions must be given. Since all these instructions truncate on fetching, the Central Processing Unit (CPU) program can never access fraction bytes beyond the selected precision. For the programmer, therefore, results may be considered to be stored truncated.

FLOATING-POINT DATA FLOW

- Floating-point registers are used instead of the general-purpose registers for storing floating-point operands.
- FP fraction arithmetic is performed by the basic CPU arithmetic and logic data flow.
- Two registers are added to the basic arithmetic data flow, the AX register and the floating-point scratch register.
- A separate exponent arithmetic data flow is provided for characteristic handling and consists of:
  - Exponent registers A and B
  - Exponent carry look-ahead
  - Exponent funnel
  - Plus 1 and minus 1 carry generators.
- Four FP sequence latches are added for control purposes.

The floating point feature provides additional registers, data flow and control circuitry to enable the floating-point instructions to be executed. These additions cause the console facilities and the checking features of the Model 44 to be extended.

Floating-point operands are not stored in the General-Purpose Registers (GPR's) but are contained in separate FPR's which are provide with the floating point feature. These FPR's are numbered 0, 2, 4 and 6 and can contain long-precision (double-word) floating-point operands. Note that GPR's are used, however, during the I-fetch of RX-type floating-point instructions.

The fraction arithmetic is performed in the basic Arithmetic and Logic Section (ALS) of the CPU but a separate section of data flow is provided for characteristic arithmetic. This section of the data flow (shown in Figure 2-1 and in FEMD Figure 1001) contains the following units:

Exponent funnel	Exponent CLA
Exponent register A	Plus 1 carry generator
Exponent register B	Minus 1 carry generator

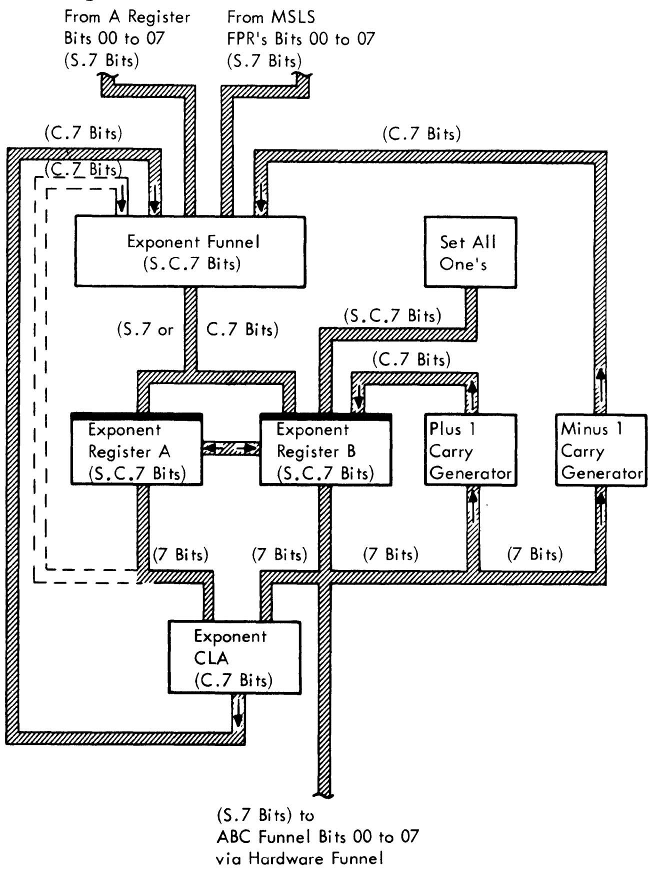
This additional section of the data flow is logically equivalent to the main arithmetic section; the exponent registers A and B and the exponent CLA perform the same logical function on input data as the A and B Registers and the main CLA.

Two additional registers, each of 32 bits, are provided in the main data flow to facilitate the handling of long-precision operands. The registers are the AX register (which can be accessed only by a 32-bit interchange with the A register) and the FP scratch register (which can be loaded from the B register and read out to the Hardware (HW) funnel).

The floating-point instructions in general require lengthy execute times, and four additional FP sequence latches are used to provide adequate controls for the execute phase of these instructions.

During the execute phase of these instructions, the normal rules for single-cycle operation, apply as stated in "Console" in FEMM IBM System/360 Model 44, Form Y33-0007. The additional data flow registers, all FPR's, and the status of the FP sequence latches and their control latches can be displayed from the console. The FPR's and AX register may be stored into, also from the console.

The FP sequence latches are subject to the same type of control check as the basic sequence latches. Other program checks are provided for the exceptions which can occur with floating-point operation; these exceptions include specification (floating-point), significance, FP divide, exponent overflow and exponent underflow.



Legend:	
7 :	Characteristic Bits 1 to 7
C :	Bit Carry
S :	Bit Sign (Bit 0)

Figure 2-1. Exponent Arithmetic Data Flow

CHARACTERISTIC ARITHMETIC

- Characteristic arithmetic is performed in the exponent register area.
- The fraction sign is stored in this area but does not take part in characteristic arithmetic.
- Any carry from characteristic arithmetic is preserved for exponent underflow/overflow detection circuits.
- Arithmetic processes include:
  Characteristic addition
  Characteristic subtraction
  Characteristic incrementing
  Characteristic decrementing

The characteristic arithmetic is performed in the exponent register area (Figure 2-1). The whole characteristic (fraction sign and characteristic) is brought from the main data flow to exponent registers A and B via the exponent funnel.

Characteristic arithmetic is, however, performed only on bits 01 to 07 and any carries are preserved in the exponent registers. Thus, these registers are nine bits wide and consist of bit 'sign', bit 'carry' and bits 01 to 07.

The arithmetic processes which may be performed by the exponent register area are: characteristic addition, subtraction, incrementing and decrementing. The result of this arithmetic, formed in exponent register B and gated to the HW funnel at the appropriate time, consists of the fraction sign (bit 'sign') and the characteristic (bit 01 to 07). Bit 'carry' is used for exception detection and is not gated out as part of the characteristic result.

Thus, the different bus widths as shown in Figure 2-1, contain seven, eight or nine bits, depending on their use. Inputs and outputs of the exponent register area are seven data bits with the fraction sign bit. Inputs to each of the three carry units are seven data bits while the output of these same carry units is eight bits (seven data bits and a carry bit).

Characteristic Addition

- Used for the multiply instruction.

The exponent register area is used for characteristic addition for the multiply operation. For this operation the two characteristics are added and 64 is subtracted from the result (using 'excess 64 arithmetic') to obtain the true result characteristic.

Example

		Exponent
Characteristic 1:	98	+ 34
Characteristic 2:	32 +	- 32 +
Sum of characteristics:	130	
Less 64:	64 -	
Result characteristic:	66	+ 2

Characteristic Subtraction

- Used for the divide instruction.
- Used for exponent matching for add, subtract, and compare instructions.

The exponent register area is used for characteristic subtraction associated with the divide instruction. One characteristic is subtracted from the other and 64 is added to the result ('excess 64 arithmetic') to provide the true result characteristic.

Example

		Exponent
Characteristic 1:	73	+ 9
Characteristic 2:	49 -	- 15 -
Difference of characteristics:	24	
Add 64	64 +	
Result Characteristic:	88	+ 24

For the add, subtract and compare instructions, an arithmetic difference between the two characteristics is needed to determine the number of shifts required for matching the exponents of the two operands; the difference is obtained by subtraction.

Characteristic Incrementing

- Used during shift-right-four operations (digit shifting).
- Used for some characteristic adjustment during the divide instruction.

For digit shifting of a fraction to the right, the corresponding increment of one must be performed on the exponent for each position shifted. This function can be performed on a characteristic in exponent register B by the plus 1 carry generator.

Characteristic Decrementing

- Used during shift-left-four operations (digit shifting).
- Used for some characteristic adjustment during the divide instruction.

For digit shifting of the fraction to the left, the corresponding decrement of one must be performed on the exponent for each position shifted. This function can be performed on a characteristic in exponent register B by the minus 1 carry generator.

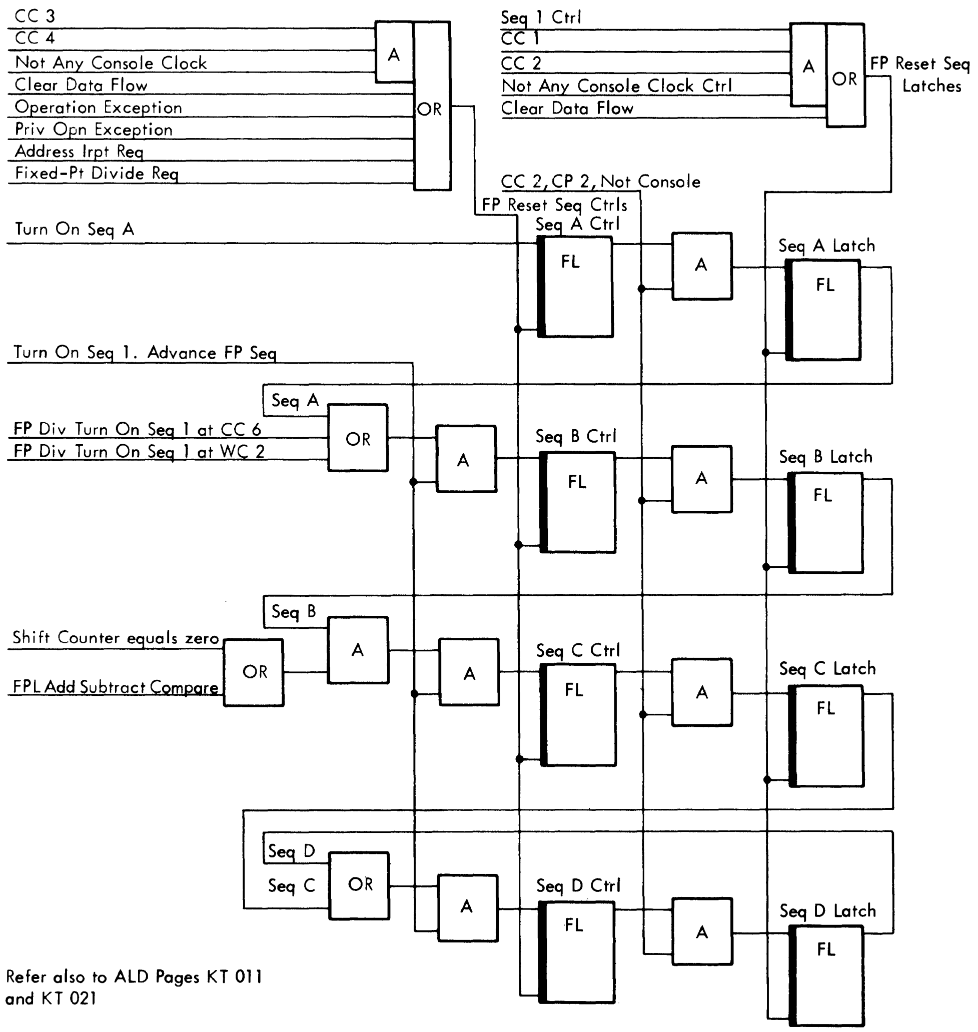
CONTROL COMPONENTS

- Lengthy execute times require additional sequence latches.
- Four additional FP sequence latches (A, B, C, D) are required.
- FP controls are provided by the AND of the FP sequence latches and the basic sequence latches.
- FP sequence latches are subject to control-checking.
- FP sequence latches advance (A to B, B to C, C to D) each time 'sequence 1' is called.
- Basic sequence latches can turn on in any order between each time that 'sequence 1' is called.

Because of the lengthy execution times required for floating-point instructions, four FP sequence latches (A, B, C and D) are added with the floating point feature. The latches augment the basic sequence latches by providing, with them, 20 combinations of controlling sequences for floating-point operations. These combinations are provided by the AND condition of each of the four FP sequence latches with the five basic sequence latches to give sequence controls that are called, for example, 1A, 1B, 2B, 5D.

The FP sequence latches are subject to the same control check as the basic sequence latches. That is, a control check occurs if two of these FP latches are on at CC 2, CP 2 time of any compute clock cycle.

Figure 2-2 shows the FP sequence latches and the FP sequence controls. The first FP sequence combination used in any floating-point instruction is sequence 1A. The FP sequence A latch then remains on until the next sequence 1 is called, at



Refer also to ALD Pages KT 011 and KT 021

Figure 2-2. FP Sequence Controls and Latches

which time the FP sequence latches are advanced (in this case) from A to B.

While the FP sequence latch is on, the basic sequence latches can be used in any combination with it although generally each basic sequence latch is used only once within each group. For example, the sequence progression for the FP short-precision divide instruction is sequence 1A, 5A, 2A, 4A, 1B, 2B. Note that the FP sequence A latch remains on through the basic sequences 5, 2 and 4 and advances to sequence B when sequence 1 is called.

ARITHMETIC COMPONENTS

Floating-Point Registers

- Four floating-point registers, numbered 0, 2, 4 and 6, are provided.
- Each register is 64-bits wide.
- The high-order 32 bits use special triggers in hardware while the low-order 32 bits use extension storage.
- Addressable by the Ra and Rb registers and the console.
- The hardware storage is of the medium-speed local storage type.

Use

Four FPR's are provided with the floating point feature. They are used exclusively to store FP operands and are numbered 0, 2, 4 and 6. Each register has a 64-bit capacity so that long-precision operands can be stored.

The registers are split into two 32-bit sections, the high-order 32 bits (0 to 31) being hardware sections and the low-order 32 bits (32 to 63) being in core-storage locations in extension storage. For short-precision operands, only the hardware section of the register is used, while for long-precision operands both sections are used.

Input

The hardware section of the register is loaded directly from the B register, while the extension storage section is loaded from the B register via the SDR.

Output

Bits 0 to 31: These bits are gated to the corresponding positions of the ABC funnel via the HW funnel. Gating along this bus can select bits 0 to 31 or bits 8 to 31 (the fraction bits) only.

Bits 0 to 7: These bits can be gated to the exponent funnel (exponent arithmetic section only).

Bits 32 to 63: These bits are gated to the ABC funnel through the normal path for data from core storage, but the entry into the ABC funnel is controlled by the setting of the variable-precision switch. Byte 0, bytes 0 and 1, bytes 0, 1 and 2 or bytes 0, 1, 2 and 3 can be selected.

Controls

The part of the registers located in extension storage is controlled by the normal storage addressing circuits with the address formed from the Ra and/or Rb registers.

The part located in hardware is addressed from the same Ra or Rb register and is controlled for write or read by the circuits shown in Figure 2-3.

Read Controls

FPR addresses are obtained from the Ra or Rb register by decoding the FPR number that is defined by the contents of the particular register.

'I-cycle control latched up' latch is used to control which of these two registers is decoded to provide the FPR address data. The latch is set by the I-cycle control and is reset during the instruction execution after operand 2 has been fetched. When this latch is on, the Rb register is decoded to allow operand 2 data to be fetched; when the latch is off, the Ra register is decoded to allow operand 1 data to be fetched and the result to be stored in the operand 1 FPR.

The FPR addresses can also be generated from the console address switches during console store or display operations.

The decoding network output consists of four 'FPA primary read decode xx', lines. Only one line can be active at a time. The common input condition of 'FPA address' of the subsequent AND blocks is permanent during execution of a Floating Point Arithmetic (FPA) instruction or when FPR's are addressed from the console; these AND blocks raise the 'array drive read FPR' line corresponding to the decoded address.

The 'array drive read FPR' line, when active (+), gates the state of all 32 positions of the selected FPR hardware portion to the HW funnel via the last OR blocks shown in Figure 2-3. In the HW and ABC funnels, the data (which is thus available at all times during FP executions other than when the ABC funnel is handling other data) is controlled further by

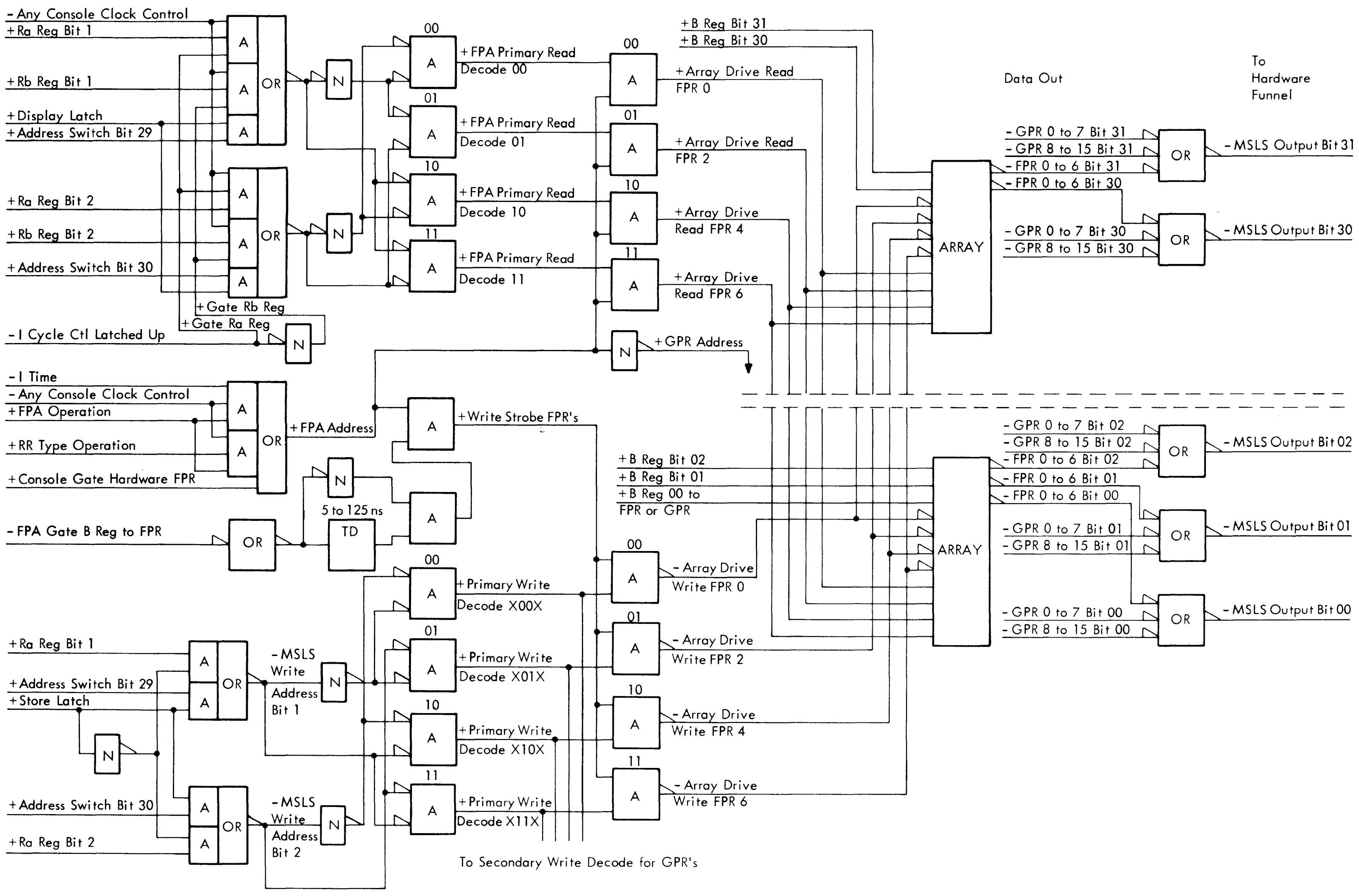


Figure 2-3. FP Medium-Speed Local Storage Read and Write Drives

the different FP operations for admission to the ALS of the data flow.

Write Controls

The FPR address decoding circuits for write are similar to those for read but the Ra register only is used during instruction execution as no operation stores its results in the operand 2 field. The write decoding circuits are common to the FPR's and to the CPR's used with the accelerator feature. However, the output 'primary write decode xxxx' shown in Figure 2-3 is sufficient to address the FPR's. Further decoding to address the GPR's as necessary is achieved by including Ra register bits 0 and 3 in the address analysis. One only of the 'primary write decode' lines shown can be active at a time.

The common condition 'write strobe FPR's is active when 'FPA address' is present (+) with 'FPA gate B register to FPR'. An adjustable time delay has been added, which is set to suppress the write strobe 30 nanoseconds before the B register is reset. Since the B register provides the input data, this suppression prevents possible trouble if the B register is reset while the 'FPA gate B register to FPR' signal is still effective.

NOTE: Except during FP operation execution, the condition 'FPA address' is absent and the condition 'GPR address' (obtained by inversion) is permanently available. When the GPR's are implemented in Medium-Speed Local Storage (MSLS), the same kind of circuits are used for read GPR's and for write GPR's as for the FPR's.

Description

The FPR's are so designed that the high-order 32 bits consist of hardware latches, and the low-order 32 bits are accommodated in a predetermined area of extension storage. A low-order word needs a one-microsecond access time; however, the high-order word has a 250-nanosecond read/write capability, as MSLS modules with a non-destructive read cycle are used (refer to "Accelerator Feature" in Chapter 2, Introduction and Functional Units, Form Y33-0001). Only the hardware (MSLS) section is used for FP short-precision operands. For FP long-precision operands, both hardware and core storage sections are used.

This core storage use allows the amount of hardware to be reduced without affecting the performance of long-precision operations, because the two halves of an operand can be fetched in parallel. The low-order word (in extension storage) is addressed, read out and placed into the SDR while the high-order wor is fetched in parallel from the hardware registers. The high-order word is loaded in the appropriate register (A, B or C), which is then interchanged with the auxiliary register (AX or BX register). The second word is available from the SDR as soon as the handling of the first word is terminated (that is, when A and B registers are free).

Display and Store Facilities

Both parts of each FPR can be displayed separately on the console. The high-order 32 bits (hardware) are displayed by the SDR via the funnels and the B register, while the low-order 32 bits are displayed by the SDR after being read out from the extension storage.

The display is produced by operating the display pushbutton with the storage select rotary switch at the FPR position. The address of the register is defined by the address switches 29, 30 and 31. (Position 31 indicates that the low-order part of the FPR is used.)

For the same setup of address switches, the store operation is performed by operating the store pushbutton. This operation stores the content of the data switches in the addressed section of the FPR selected.

Floating-Point Scratch Register

- Used to store, provisionally, certain partial results during fraction arithmetic.
- Used to develop the floating-point divide quotient.
- Is 32 bits wide.
- Each position is a multi-input trigger.
- Can be displayed on the console.

Use

The scratch register is used to store parts of intermediate fraction results during FP multiply or divide operations.

The FP divide quotient is developed by conditioning the quotient bit entry to position 31, and inserting it on the next shift-left-one operation.

Input

The input to the scratch register is from corresponding positions of the BX register. Also, position 31 receives the quotient bit (during FP divide) and each other position receives the contents of the position to its right on a shift-left-one operation.

Output

All 32 positions of the scratch register can be gated to the corresponding positions of the ALS registers by gating them to the scratch register output bus and routing them through the HW and ABC funnels. By this method, the stored partial results of the fractions are re-introduced into the ALS for further processing. The data paths involved are shown on FEMD Figure 1001.

Positions 28 to 31 are tested for zero for use during FP divide normalization cycles.

Note that when the scratch register content is not gated to its output bus, the state of the output bus lines is the positive 'logical one' condition. This condition is also the normal output of the HW funnel and is used for generating the decrement value for updating the interval timer.

Controls

The scratch register is used and controlled to store the BX register with FP multiply and divide operations only. During the division cycles of FP divide, shift-left-one operations are performed on the scratch register content and, as the developing of the quotient progresses, each FP quotient bit is inserted in the vacated position 31.

Description

Each position of the scratch register is a multi-input trigger as shown in Figure 2-4. Note that position bit 31 receives the quotient bit instead of the next low-order position. The multi-input trigger and the shift signal supply circuits are described in FEMM IBM System/360 Model 44, Form Y33-0007.

Display

The 32 positions of the scratch register are displayed on the console with the CPU roller switch set to position 7.

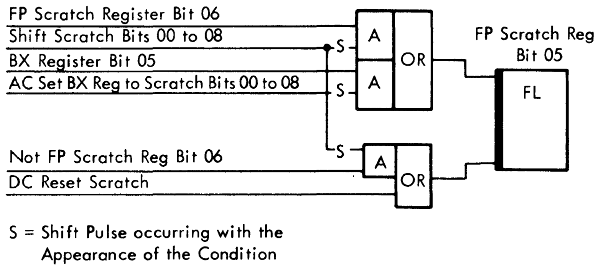

FP Scratch Register Bit 06
Shift Scratch Bits 00 to 08
BX Register Bit 05
AC Set BX Reg to Scratch Bits 00 to 08
Not FP Scratch Reg Bit 06
DC Reset Scratch
S
A
OR
FP Scratch Reg Bit 05
FL
S = Shift Pulse occurring with the Appearance of the Condition

Figure 2-4. Typical Scratch Register Position

Exponent Register A

- Used for exponent arithmetic in the exponent data flow section.
- Nine bits wide: seven exponent bits, one carry bit and one fraction sign bit.
- Controlled by the particular FP instruction.
- The fraction sign bit is not affected by arithmetic operations on the exponent.
- Each position is a multi-input trigger.
- Bit 'sign' and bits 1 to 7 can be displayed on the console.

Use

Exponent register A is used in exponent arithmetic in conjunction with exponent register B to feed exponent CLA. Its sign bit is the fraction sign bit and is used with bit 'sign' of the exponent register B in analysis circuits for correct result sign gating (gate B00 inverted).

Input

Exponent register A receives its input (one characteristic) from the exponent funnel, at the time of operand characteristic gating. The register receives the exponent register B contents by interchanging.

Output

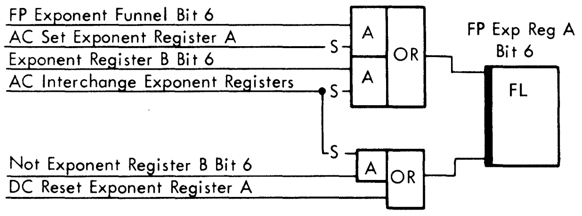
The output is to the corresponding position of the CLA (bits 1 to 7 only), and to exponent register B for interchanging. Position 1 to 3 are also gated to analysis circuits to set the operand 1 exponent high (HI) latch, operand 1 exponent low (LO) latch or exponents equal (EQ) latch. As the B register bit 00 normally contains the fraction sign bit of the result, bit 'sign' is fed to the circuits that control the 'gate B register bit 00 inverted' line.

Control

The input commands are generated from the FP operation decoding circuits and from the FP sequence latches.

Description

A typical exponent register A position is shown in Figure 2-5. The register receives input from the exponent funnel at the time it receives a character-


FP Exponent Funnel Bit 6
AC Set Exponent Register A
Exponent Register B Bit 6
AC Interchange Exponent Registers
Not Exponent Register B Bit 6
DC Reset Exponent Register A
S
A
OR
FP Exp Reg A Bit 6
FL

S = Shift Pulse occurring with the Appearance of the Condition

Note: Exponent register A bit sign has a binary-triggered exponent funnel entry.

Figure 2-5. Typical Exponent Register A Position (Bit 6)

istic from the A register or from the FPR's. The carry bit input is fed not from the A register or the FPR's but from the exponent register B.

Display

Bit 'sign' and bits 1 to 7 only are displayable on the console in CPU roller switch position 2 with the CPU 2 switch operated. The display is indirect and operates similarly to the AX or BX register display, but the interchange made by the CPU 2 switch position is with the exponent register B.

Special Aspects

The positions of the exponent register A are single-triggered, except for bit 'sign' which is binary-triggered. Since exponent register A receives only one characteristic, the Exclusive OR (EXOR) properties of the binary-triggered input have no special effect on the operation.

<u>Exponent Register B</u>

- Used for exponent arithmetic in the exponent data flow section.
- Nine bits wide: seven exponent bits, one carry bit and one fraction sign bit.
- Controlled by the particular FP instruction.
- Performs an Exclusive OR function on data from the exponent funnel.
- The fraction sign bit is not affected by arithmetic operations on the exponent.
- Each position is a multi-input trigger.
- Bit 'sign' and bits 1 to 7 can be displayed on the console.
- Is the path out of the exponent data flow.

Use

Exponent register B is used in characteristic handling in conjunction with exponent register A and the exponent CLA, and the exponent funnel. It is also the only path out of the characteristic handling area for the results and has therefore an interchange capability with exponent register A. Exponent register B can be incremented or decremented with the aid of the plus 1 and minus 1 carry generators when shift-left-four or shift-right-four operations are performed during fraction manipulation.

Input

The inputs from the operand characteristic area, the exponent CLA and the minus 1 carry generator enter exponent register B via the exponent funnel. Inputs are binary-triggered at the register to perform an EXOR function.

A further (binary-triggered) input to exponent register B is from the plus 1 carry generator. This input does not affect bit 'sign' position which receives its input only from the characteristic. Exponent register B can receive data from exponent register A with an 'interchange exponent registers A, B' signal.

The 'clear data flow' input sets all ones in the whole exponent register B. The ones remain if a dc reset exponent register B signal does not occur.

Output

All nine bits can be gated to exponent register A. All bits, except bit 'carry', can be gated to the HW funnel for assembly of the result characteristic.

All seven exponent bits (1 to 7) can be gated to the plus 1 and minus 1 carry generators where an analysis, for exponent register B equals zero or equals one, is made. These seven bits can also be gated to the CLA, to the console for display and to the operand 1 exponent HI, operand 1 exponent LO, or exponents EQ latch circuits.

Bits 1 and 'carry' are used by the underflow and overflow detection circuits and bit 'sign' is used by the complement-add circuits for add, subtract and compare instructions.

Controls

The input command lines are normally generated from the FP operation analysis and from the FP sequence latches as shown in the ALD's. However, 'DC reset exponent register B' is raised only during the FP multiply operation where an addition of the exponents is required.

Description

Figure 2-6 (upper part) shows that exponent register B (like B register in the main CPU data flow) performs an EXOR function to produce an analysis combination of both operands for feeding the CLA. Note that exponent register B feeds the exponent CLA directly. There is no need for an exponent register C analagous to the C register in the main CPU data flow as the use of conditioning inputs from the exponent funnel prevents possible feedback.

Because bit 'sign' represents the sign of the fraction and is not the exponent sign, it is not altered by exponent add or subtract operations (+1, -1 CLA gating), although it is set from the exponent funnel.

Display

The exponent register B bit 'sign' and bits 1 to 7 only can be displayed on the console with the CPU roller rotary switch in position 2.

Exponent CLA

- Carry generator and carry-rippling circuits for exponent arithmetic.
- Boolean expression is Kn = (Exp A. Not Exp B) + (Exp B. Kn-1).
- Output is controlled by the instruction.
- The high-order output is the bit 'carry' and is part of the exponent underflow/overflow detection.

Use

The exponent CLA, which is seven bits wide and is part of the exponent arithmetic area, provides the carries required to complete exponent arithmetic when they are required by the operation.

Input

The inputs to the exponent CLA are fed directly from bits 1 to 7 of both exponent register A and exponent register B.

Output

The output from the exponent CLA is gated to the corresponding position of the exponent funnel. Eight output lines are used, as the bit 'carry' developed by the exponent CLA is gated to the bit 'carry' position of exponent register B for underflow and overflow detection.

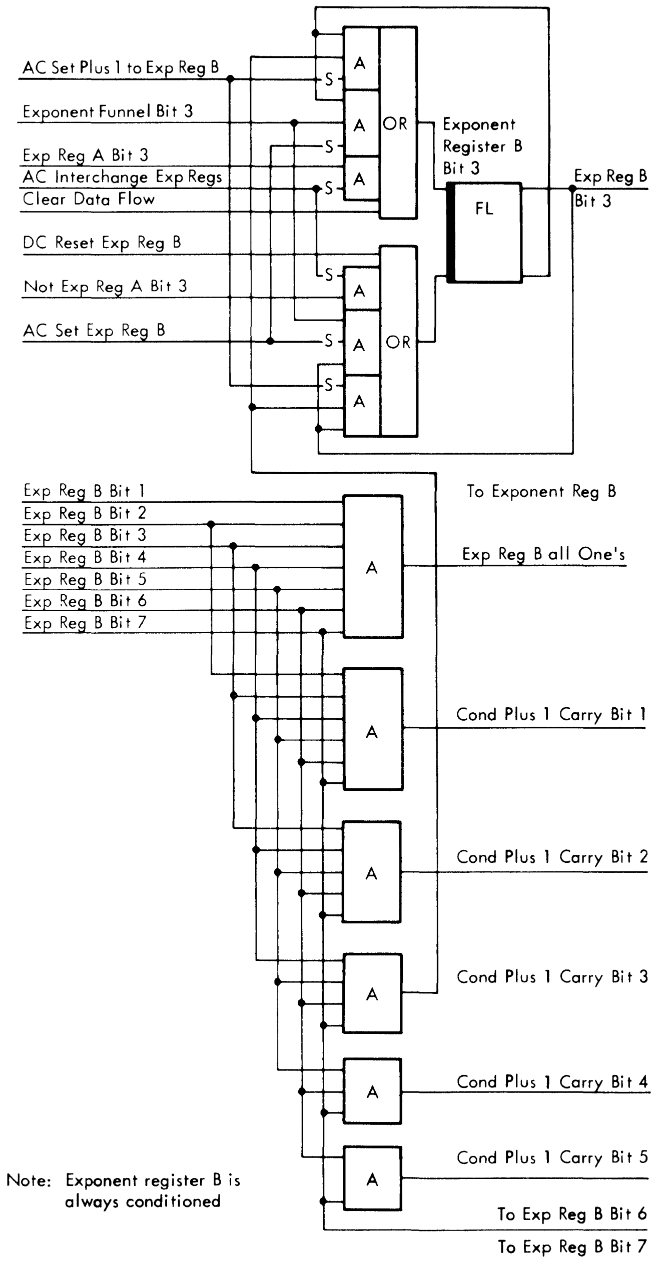


Figure 2-6. Typical Exponent Register B Position and Plus 1 Carry Generator

Controls

There are no controls on the input to the exponent CLA, which is fed directly from the output of exponent registers A and N. The gating of the exponent CLA output through the exponent funnel is controlled and is gated by the line 'condition sum to exponent register'. The timing of this signal differs for each of the various floating-point operations requiring exponent addition or subtraction.

Description

The exponent CLA is a seven-bit wide block of unlatched high-speed logic. This logic is shown in Figure 2-7. The exponent CLA performs a similar function to that of the CLA in the main ALS of the machine.

The carry-in to each position of exponent register B is derived from the exponent register A (OR of the two characteristics) and the exponent register B (EXOR of the two characteristics).

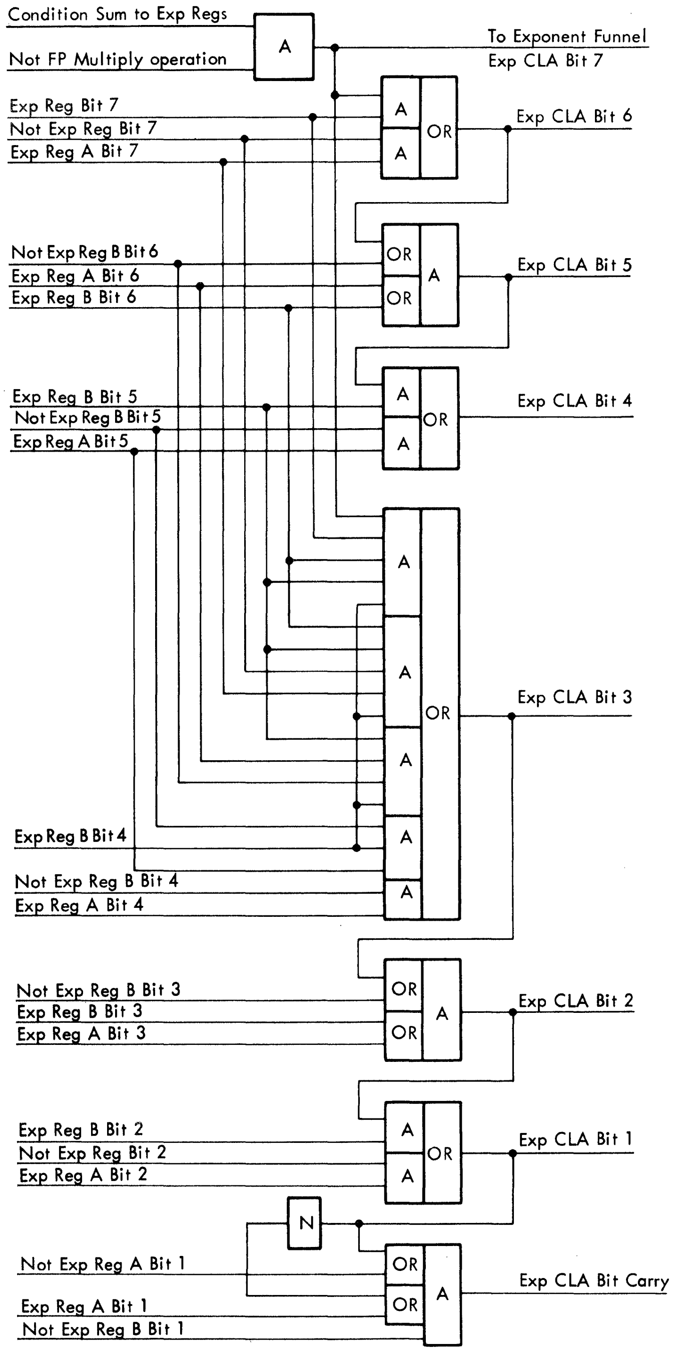


Figure 2-7. Exponent CLA

A carry-in to a position is equivalent to a carry-out of the preceding position which is given by the following:

1. A one-bit in exponent register A and a zero bit in exponent register B. This state is the (1+1) condition for this bit and a carry-out will always result regardless of the carry-in condition for this position.
2. A one-bit in exponent register B and a carry-in condition to that bit. This state is the (1 + 0 + carry-in) condition and represents a carry ripple or propagation through a bit position.

Bit position 7 of both exponent registers A and B is used to generate the carry-in for bit position 6, bit position 6 of both registers is used to generate the bit position 5 carry-in, and so on. The input from bit position 1 generates the eight output line (bit carry).

Bit position carry in the exponent register B is set and reset with the other positions and is used in underflow and overflow detection.

Note that the line 'not FP multiply op' (Figure 2-7) is used to provide the CLA carry-in as 'carry-in to exp CLA' (not shown). This is valid because, for all but multiply operations, characteristic subtraction is performed. The principle of using the CLA output of one bit to feed the next position (as used in the main CLA) is applied.

Plus 1 Carry Generator

- Provides carry-in bits to increase the value of exponent register B.
- Output is direct to exponent register B input.

Use

The plus 1 carry generator is used to increase by one the value of exponent register B during fraction manipulation, such as during the shift-right-four action that occurs with characteristic matching in add, subtract or compare operations.

Input and Output

The input to the plus 1 carry generator is the data bits 0 to 7 of the exponent register B. The output is directly coupled to a conditioning input to exponent register B.

Controls

There are no controls on the input to the plus 1 carry generator from exponent register B. The output is set into exponent register B by the 'AC set plus 1 to exponent register B' signal (Figure 2-6).

Description

The logic of the plus 1 carry generator is shown in Figure 2-6, together with a typical exponent register B position.

The logic used is simple cascading AND circuits that produce the carry-in required to increase the value of the exponent register B by one.

Minus 1 Carry Generator

- Provides carry-in bits to reduce the value of exponent register B.
- Output is normally conditioned through the exponent funnel if no other exponent funnel controls are present.

Use

The minus 1 carry generator is used to reduce the value of exponent register B by one during fraction manipulation such as normalizing.

Input

The input to the minus 1 carry generator is the data bits 0 to 7 of exponent register B.

Output

The output of the minus 1 carry generator is gated to the corresponding position of the exponent funnel. Eight output lines are used, as the bit 'carry' developed by the minus 1 carry generator is gated to the bit 'carry' position of exponent register B for underflow detection.

Controls

There are no controls on the input to the minus 1 carry generator from exponent register B. The gating of the output lines through the exponent funnel is normally conditioned if no other gates are present.

Description

The logic of the minus 1 carry generator is shown in Figure 2-8. As with the plus 1 carry generator, AND circuits are used to produce the carry-in required to reduce the value of exponent register B by one.

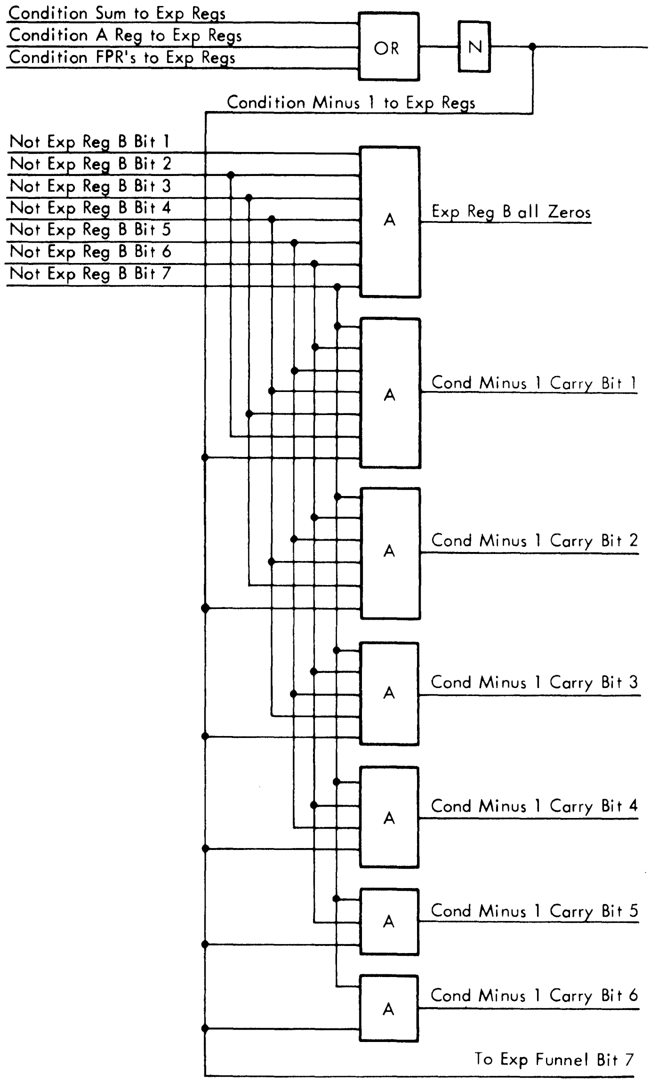


Figure 2-8. Minus 1 Carry Generator

Exponent Funnel

- Collects information to be directed to exponent registers A or B.
- Eight bits wide: seven data bits and sign bit, or seven data bits and carry bit.
- Unlatched logic is used.
- Entry via the exponent funnel is controlled by the instruction.
- Minus 1 carry generator is normally conditioned.

Input

The active inputs to the exponent funnel are:
- A register (bits 00 to 07)
- FPR's (bits 00 to 07)
- Minus 1 carry generator (bits 1 to 7 and bit 'carry')
- Exponent CLA (bits 1 to 7 and bit 'carry')

An inactive input comes from the exponent register A output, but the corresponding gate at the exponent funnel is never conditioned.

Refer to Figure 2-9 for the exponent funnel logic. Note that the data from the exponent CLA bit 1 is inverted for FP multiply and divide operations to fulfil the requirements of excess 64 arithmetic.

Output

The exponent funnel output is available at all times as an input to both exponent register A and exponent register B. Unlike the A and B registers of the ALS, the exponent registers A and B require an 'AC set' pulse to read in.

Controls

The exponent funnel controls are shown in Figure 2-9, the conditioning lines being:
- Condition minus 1 carry bit (normally conditioned)
- Condition sum to exponent registers
- Condition FPR's to exponent registers
- Condition A register to exponent registers

Condition Minus 1 Carry Bit: This conditioning line is always present if no other conditioning line is active. It provides the minus 1 carry bit generator data to the inputs of the exponent funnel bits 1 to 7 and bit 'carry'.

Condition Sum to Exponent Registers: This conditioning line de-conditions the condition minus 1 carry bit line and gates the exponent CLA output through the exponent funnel bit 1 to 7 and bit 'carry'.

Condition FPR's to Exponent Registers: This conditioning line de-conditions the condition minus 1 carry bit line and gates the bits 00 to 07 of the FPR read-out on to the FPR output bus through the exponent funnel bit 'sign' and bits 1 to 7.

Condition A Register to Exponent Registers: This conditioning line de-conditions the condition minus 1 carry bit line and gates bits 00 to 07 of the A register through the exponent funnel bit 'sign' and bits 1 to 7.

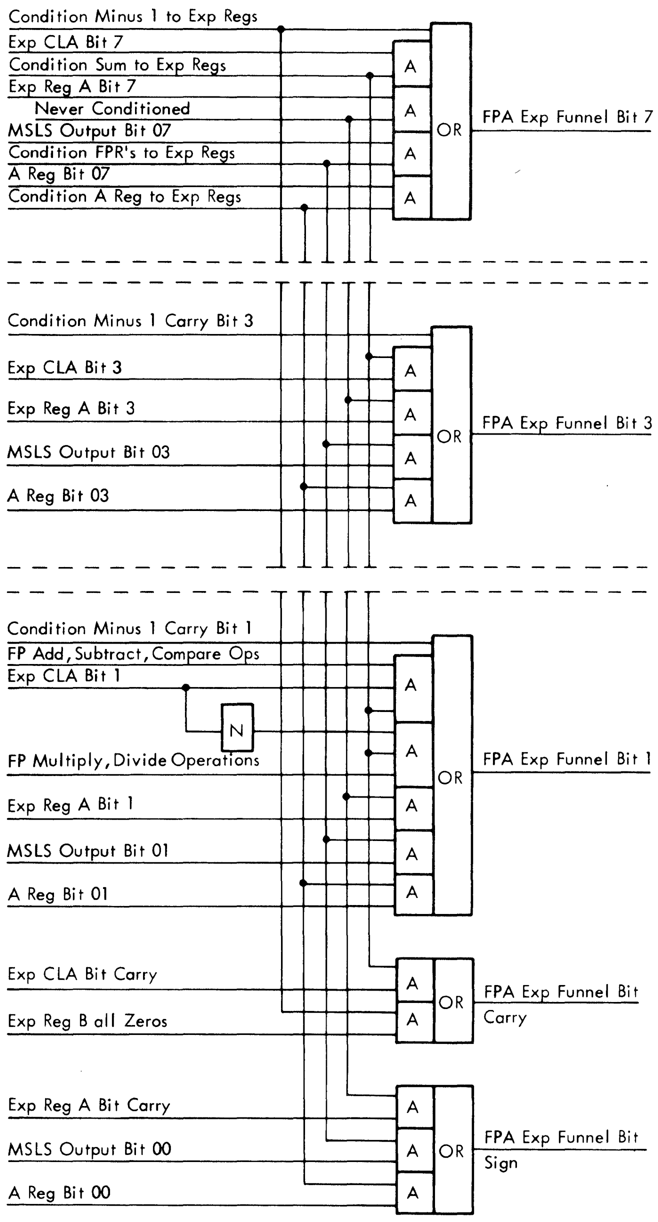


Figure 2-9. Exponent Funnel

Description

The logic for the exponent funnel is shown on the following ALD pages:

Bit 'sign' position	:	Page KT 181
Bits 1 to 7	:	Pages KT 231 to 251

ABC Funnel (Variable-Precision Controls)

- Used for truncating low-order bits of long-precision operands.
- Amount of truncation is defined by switch setting.
- Variable-precision switch controls the 'gate true/criss-cross to ABC funnel' signals.

Use

The FP controls on the 'true/criss-cross to ABC funnel' gating allow the low-order 32 bits of long-precision floating-point operands to be truncated to the precision defined by the variable-precision switch on the console.

Input and Output

All inputs and outputs of the ABC funnel are as described in Chapter 2 of Introduction and Functional Units, Form Y33-0001.

Controls

The setting of the variable-precision switch controls the gating of the true/criss-cross output to the ABC funnel on those cycles where the 32 low-order bits of a long-precision operand are fetched.

These fetch cycles are always an R1, R2 or EA cycle and are defined by the logic in the upper part of Figure 2-10.

The output of the variable-precision switch is then used to control the bit gates that are conditioned on the 'true/criss-cross to ABC funnel' path.

Description

Figure 2-10 shows that the bits gated from the true/criss-cross to the ABC funnel for each position of the variable-precision switch are as in the following table.

Variable-Precision Switch Setting	Bits Gated
14	00 to 31
12	00 to 23
10	00 to 15
8	00 to 07

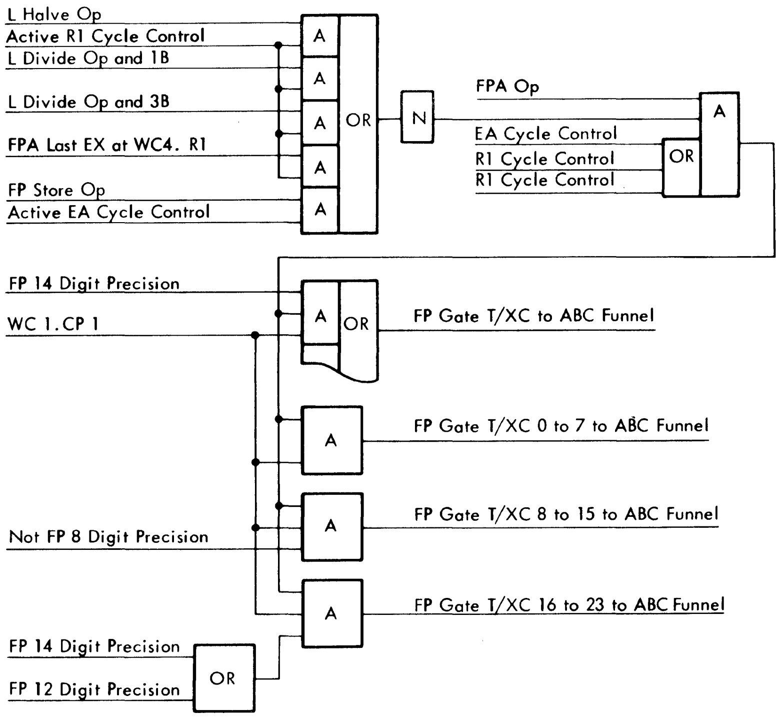


Figure 2-10. FP Variable-Precision True/Criss-Cross Controls

CONSOLE FACILITIES

- FPR's 0, 2, 4 and 6 can be displayed in the data lights and stored into from the data bit switches by using the storage select switch.

- The following FP feature registers can be displayed by the CPU roller:
  AX register (Position 3, CPU 2)
  Exponent register A (Position 2, CPU 2)
  Exponent register B (Position 2)
  FP scratch register (Position 7)

- The AX register can be stored into from the data bit switches using the CPU roller position 3 and the CPU 2 switch.

- The FP sequence control latches and sequence latches are displayed in console panel lights.

- The FP sequence control check is part of the control check circuits which operate the control check lamp.

- The variable-precision switch controls the precision of long-operands to either 8, 10, 12 or 14 digits.

The foregoing statements summarize the console facilities available for the FP feature. For full details of the store-and-display procedure and the associated circuits refer to FEMM IBM System/360 Model 44, Form Y33-0007.

EXPONENT UNDERFLOW AND OVERFLOW DETECTION

- Exponent overflow can occur on FP add, subtract, multiply or divide instructions.

- Exponent underflow can occur on FP add normalized, subtract normalized, multiply or divide instructions.

- Exponent overflow occurs if the result characteristic exceeds 127.

- Exponent overflow occurs if the result characteristic is less than zero.

- Underflow/overflow detection is performed on the result characteristic in exponent register B.

During the various characteristic manipulations an exponent underflow or overflow may occur. If this condition exists at the end of the instruction execution (and the underflow exception is not masked off) the exception latch is set and a program interrupt is requested. Some operations may cause exponent underflow or overflow conditions during instruction execution which are corrected later in the operations. When the conditions occur the result characteristic is correct and the temporary exception condition is ignored.

Exponent overflow conditions take place when the result characteristic is greater than 127. This can occur during the normal execution of FP multiply and divide instructions.

For FP add and subtract instructions an exponent overflow can occur. However, the result characteristic exceeds 127 only if the partial result characteristic (after matching of exponents) is already 127 and a fraction overflow occurs. This means that a shift-right of the fraction by one digit takes place with a corresponding increment (and overflow) of the result exponent.

Exponent underflow conditions occur when the result characteristic is less than zero. This can take place during the normal execution of FP multiply and divide instructions. For FP add and subtract instructions an exponent underflow occurs only with the post-normalization process on add normalized and subtract normalized instructions. During the normalization process the value of the exponent register B is reduced for each digit position shifted. This decrementing can cause the result characteristic to be reduced to less than zero.

In order to detect the various conditions that occur, both the exponent registers A and B have a bit 'carry' position to retain the carry-outs which may occur during exponent arithmetic. This bit 'carry' position is set and reset with the other positions of the exponent registers. The bit 'carry' position also participates in any exponent register interchange.

During an exponent add, plus one or minus one operation, an underflow or overflow causes the exponent register bit 'carry' to be inverted (flipped). Since two consecutive underflows or two consecutive overflows cannot occur, the state of the bit 'carry' position can be used to detect an underflow or overflow condition.

For add, subtract and multiply instructions the initial condition of the result exponent register bit 'carry' is zero, and thus an underflow or overflow condition is detected when the bit 'carry' is a one.

For the divide instruction, the initial condition of the result exponent register bit 'carry' is a one and thus an underflow or overflow condition is detected when the bit 'carry' is a zero.

For add and subtract instructions, the underflow can be distinguished from an overflow by the state of exponent register bit 1. An overflow turns off bit 1 and an underflow turns on bit 1.

For multiply and divide instructions, the same analysis normally occurs. However, the amount of exponent arithmetic that occurs during these instruc-instructions may cause the exponent register B bit 1 to be inverted without a corresponding change in the bit 'carry'. If this condition occurs, the FP MD underflow/overflow inversion latch is set on and the overflow is detected when the exponent register B bit 1 is on. Conversely, for this condition, an underflow is detected when the exponent register B bit 1 is off.

This logic is interpreted in Figure 2-11 and is demonstrated in the following examples.

Examples of Underflow/Overflow Detection

Example 1. Add normalized instruction

Operand 1 (hex)	:	01 00A102
Operand 2 (hex)	:	01 000010
Characteristic 1 (binary)	:	000 0001
Characteristic 2 (binary)	:	000 0001

The characteristics are equal and so the fractions can be added.

Fraction result (hex)	:	00A112
Fraction sign (binary)	:	0
Characteristic result (binary)	:	000 0001

Normalization is specified and so two shift-left-four operations are performed on the fraction result which now becomes:

A11200

The characteristic is correspondingly decremented twice as follows:

	C 1234567	Exponent (dec)
Exponent register B	0 0000001	- 63
Minus 1 carry generator	1 1111111	- 1
	0 0000000	- 64
Minus 1 carry generator	1 1111111	- 1
	1 1111111	- 65 (underflow)

This gives a result characteristic in binary notation of:

111 1111 with bit 'carry' turned on.

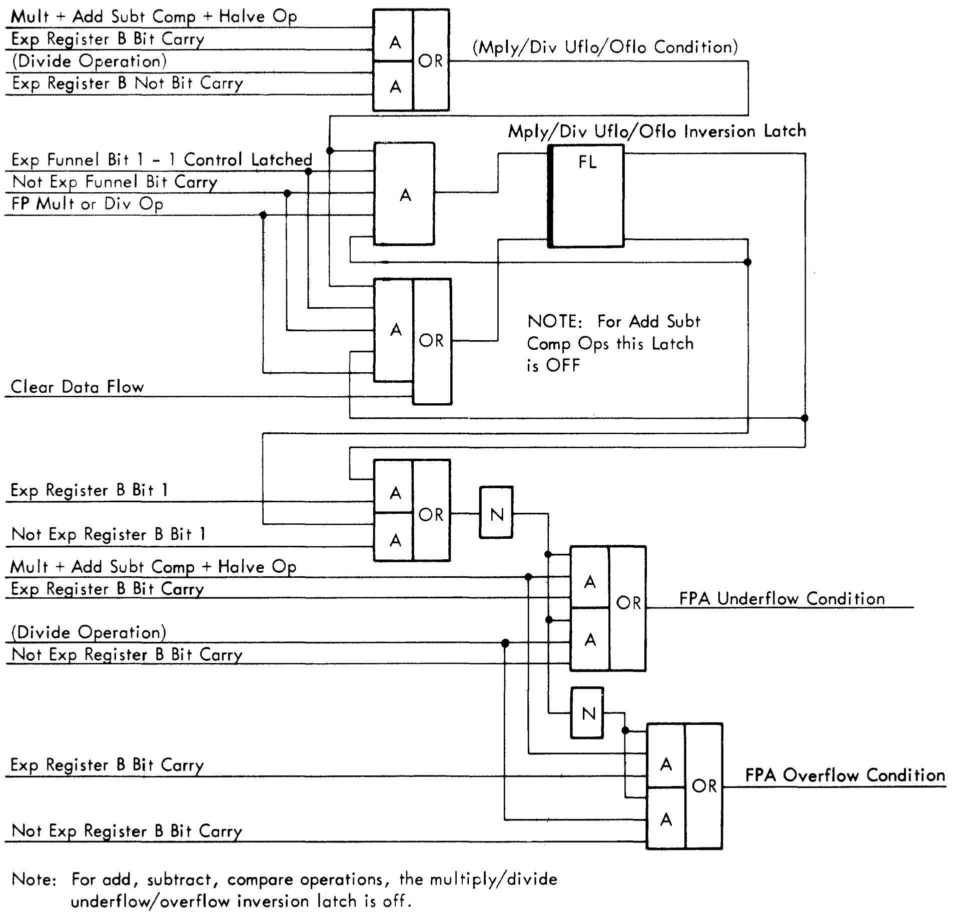

Mult + Add Subt Comp + Halve Op
Exp Register B Bit Carry
(Divide Operation)
Exp Register B Not Bit Carry
A
A
OR
(Mply/Div Uflo/Oflo Condition)
Mply/Div Uflo/Oflo Inversion Latch
Exp Funnel Bit 1 - 1 Control Latched
Not Exp Funnel Bit Carry
FP Mult or Div Op
A
FL
A
OR
NOTE: For Add Subt Comp Ops this Latch is OFF
Clear Data Flow
Exp Register B Bit 1
Not Exp Register B Bit 1
A
A
OR
N
Mult + Add Subt Comp + Halve Op
Exp Register B Bit Carry
(Divide Operation)
Not Exp Register B Bit Carry
A
A
OR
FPA Underflow Condition
N
Exp Register B Bit Carry
Not Exp Register B Bit Carry
A
A
OR
FPA Overflow Condition
Note: For add, subtract, compare operations, the multiply/divide underflow/overflow inversion latch is off.

Figure 2-11. FP Underflow/Overflow Condition Detection

For the add instruction, the exponent register B bit 'carry' turned on indicates an underflow/overflow condition and the exponent register B bit turned on indicates an exponent underflow condition.

Example 2. Add un-normalized instruction

Operand 1 (hex)	:	7EA06B02
Operand 2 (hex)	:	7FF82003
Characteristic 1 (binary)	:	111 1110
Characteristic 2 (binary)	:	111 1111

The characteristics are matched by a right-shift-four of the operand 1 fraction with corresponding increment of the operand 1 characteristic. Operand 1 now becomes:

Operand 1 (hex) : 7F0A06B0 (2)

The fractions can then be added:

Fraction 1 (hex)	:	0A06B0
Fraction 2 (hex)	:	F82003
Result fraction (hex)	:	10226B3

This fraction addition results in a fraction overflow, and a shift-right-four of the fraction with a corresponding increment of the result characteristic is necessary:

Result fraction (hex) : 10226B

The characteristic arithmetic is as follows:

	C	1234567	Exponent (dec)
Exponent register B	0	1111111	+63
Plus 1 carry generator	0	0000001	+ 1
	1	0000000	+64 (overflow)

This gives a result characteristic in binary notation of:

0000000 with bit 'carry' turned on.

For the add instruction, the exponent register B bit 'carry' turned on indicates an underflow/overflow condition and exponent register B bit 1 turned off indicates an exponent overflow condition.

Example 3. Multiply instructions

Operand 1 (hex)	:	81008000
Operand 2 (hex)	:	7A800000

Assuming that the operands are pre-normalized, the operand 1 fraction is given two shift-left-four operations with two corresponding decrements of the operand 1 characteristic. The operand 1 fraction then becomes (hex): 800000

The operand 1 characteristic arithmetic is as follows:

	S	C	1234567	Exponent (dec)
Exponent register B (binary)	1	0	0000001	-63
Minus 1 carry generator		1	1111111	- 1
	1	0	0000000	-64
Minus 1 carry generator		1	1111111	- 1
	1	1	1111111	-65

This stage represents an exponent underflow of the operand 1 characteristic. As this is not the result exponent, an exponent underflow condition is not signalled.

The multiply operation continues with the fractions being multiplied:

Result fraction (hex) : 400000

The characteristics are added using excess 64 arithmetic. This causes the carry-in to the bit 1 position to be inverted.

	S	C	1234567	Exponent (dec)
Exponent register B	1	1	1111111	- 65
Add Operand 2 Characteristic	0	0	1111010	+ 58
	1	0	0111001	- 7

The result sign and characteristic (hex) is thus B9. Note that the temporary underflow during the pre-normalization has been corrected as the exponent register B bit 'carry' of the result is now turned off.

The combined result (hex) is: B9 400 000

FP CHECK CIRCUITS

FP Sequence Control Checking

- A control check on the FP sequence control latches is performed.
- The sequence control check latch is set with more than one FP sequence control latch on at CC 1, CC 2 time.
- The FP sequence control check is performed in parallel with the basic sequence control check.

The control check on the FP sequence control latches is performed by checking (at CC 1, CC 2 time) that not more than one latch (A, B, C or D) is on. The on state of the sequence control check latch is indicated on the console by the control check lamp.

The sequence control check latch is reset by the 'master reset for latches' signal. The reset of this latch and the handling of the machine check caused by the latch are described in Chapter 2 of Introduction and Functional Units, Form Y33-0001, under the heading "Checking".

The logic used for testing the basic and FP sequence control latches is shown in Figure 2-12.

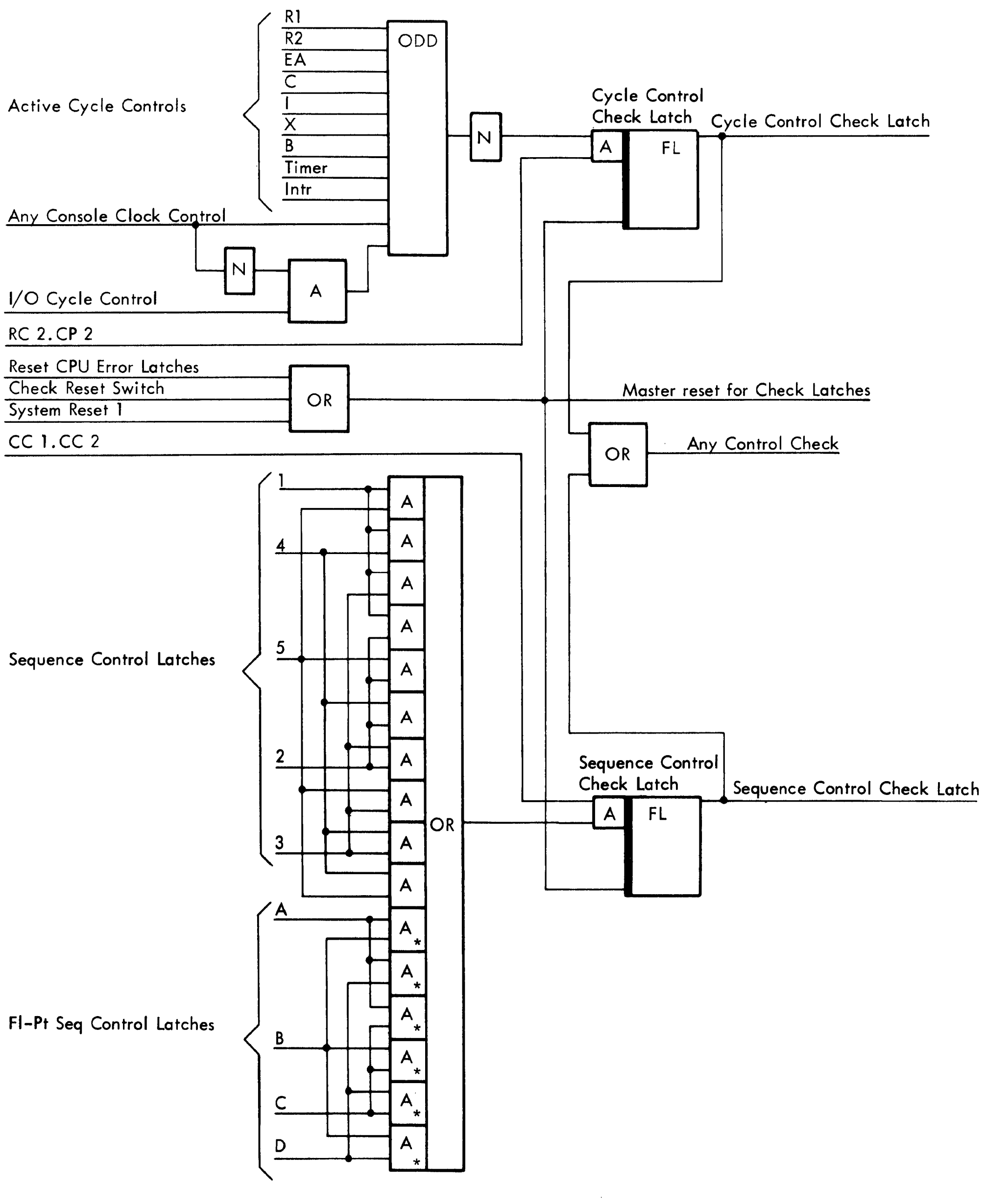
FP Program Exceptions

The types of exceptions that can occur with the floating-point feature are detailed in Chapter 1 of this manual under "Floating-Point Exceptions". The handling of all these types of exceptions is described in "Checking" in Chapter 2 of Introduction and Functional Units, Form Y33-0001, as is the method of detection of all but the FP arithmetic type of exception. Detection of the FP arithmetic type of exception is described in this section.

The significance exception and the underflow exception can occur only if the corresponding mask bit in PSW 1 is a one. All the FP arithmetic exceptions cause a program interrupt to be requested.

Significance Exception

- Can be masked by PSW 2 bit 7 when the bit equals zero.
- Is recognized when a floating-point add or subtract result is zero (zero fraction).
- Is reset by the reset interrupt latches signal.



* Floating Point Feature

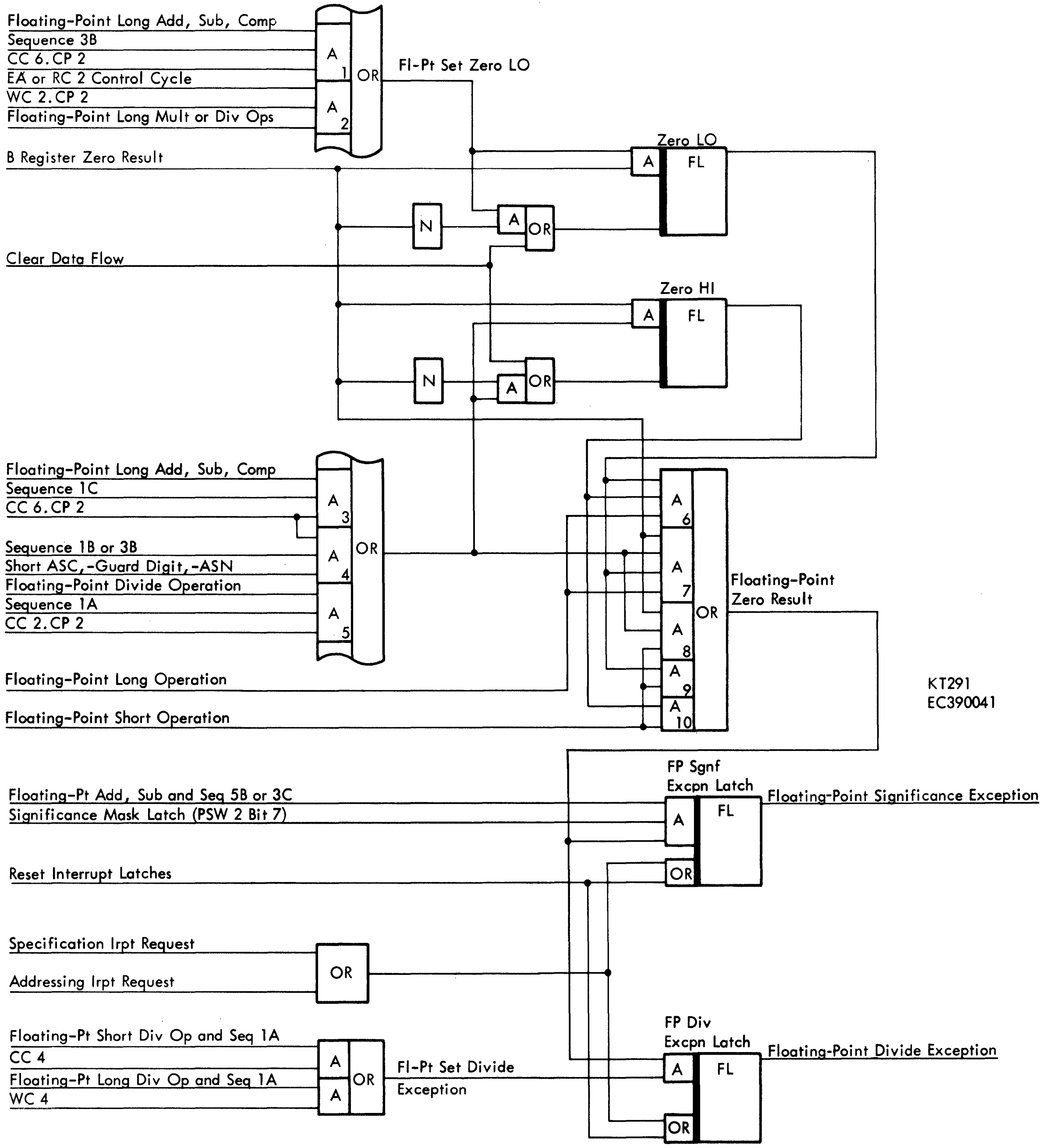
Figure 2-12. Control Checking

When the result fraction of a floating-point add or subtract operation is all zero and the significance mask bit (PSW 2 bit 7) is a one, a significance exception is recognized.

Figure 2-13 shows the significance exception latch and its associated logic. The latch is set at the end of the operation on sequence 5B for a short-precision operation, or on sequence 3C for a long-precision operation. The latch is set if PSW 2 bit 7 is a one and if the floating-point zero result line is active. AND blocks 6 and 7 are included for speed-up purposes.

Floating-point zero result is generated by different AND blocks (blocks 6 to 10) depending on the precision of the operation.

For short-precision operations, a zero HI condition (which means the leftmost part of the fraction is zero) is sufficient to define a floating-point zero result. This condition is given by the on state of the zero HI latch. The zero HI latch is set in this case with 'B register zero result' and 'floating-point set zero HI' signals obtained through AND block 4 with normalized short add, subtract or compare operations (the guard digit equals zero) or with short add



Floating-Point Long Add, Sub, Comp
Sequence 3B
CC 6.CP 2
EA or RC 2 Control Cycle
WC 2.CP 2
Floating-Point Long Mult or Div Ops
A 1
A 2
OR
Fl-Pt Set Zero LO
B Register Zero Result
Zero LO
A
FL
N
A
OR
Clear Data Flow
Zero HI
Floating-Point Long Add, Sub, Comp
Sequence 1C
CC 6.CP 2
Sequence 1B or 3B
Short ASC,-Guard Digit,-ASN
Floating-Point Divide Operation
Sequence 1A
CC 2.CP 2
A 3
A 4
A 5
A 6
A 7
A 8
A 9
A 10
Floating-Point Zero Result
Floating-Point Long Operation
Floating-Point Short Operation
KT291
EC390041
FP Sgnf Excpn Latch
Floating-Point Significance Exception
Floating-Pt Add, Sub and Seq 5B or 3C
Significance Mask Latch (PSW 2 Bit 7)
Reset Interrupt Latches
Specification Irpt Request
Addressing Irpt Request
Floating-Pt Short Div Op and Seq 1A
CC 4
Floating-Pt Long Div Op and Seq 1A
WC 4
Fl-Pt Set Divide Exception
FP Div Excpn Latch
Floating-Point Divide Exception

Figure 2-13. FP Significance and Divide Exceptions

or subtract un-normalized operations. (These conditions are OR'ed to produce the line 'short ASC, -guard digit 0, -ASN operation'.)

For long-precision operations, both a zero HI and a zero LO condition are needed to ensure that the fraction is zero (AND block 6). The B register zero result signal sets the zero HI latch in conjunction with the OR'ed output of AND block 3. In conjunction with the OR'ed output of AND block 1, B register zero result can also set the zero LO latch. Note that when the low part of the fraction is analyzed for zero, the B register result output takes into account the variable-precision switch setting; additional FP circuits produce outputs which force zero detection on the B register bytes 1, 2 and 3, bytes 2 and 3, or byte 3 only for precision switch settings of 8, 10 or 12 respectively, regardless of the real content of the B register.

The zero LO and zero HI latches are shown as FL blocks in Figure 2-13. In practice, these latches are of the same type as the condition code latches which are described in Chapter 2 of Introduction and Functional Units, Form Y33-0001.

The significance exception latch is reset during a system reset or with the reset interrupt latches signal.

FP Divide Exception

- Is recognized if the divisor fraction is zero.
- Is reset by the reset interrupt latches signal.

An FP divide exception is recognized if the fraction of the divisor is zero, as zero gives a result of infinity. Figure 2-13 shows the FP divide exception latch which is set by the floating-point set divide exception line (active at CC 4 and sequence 1A for a short-precision divide operation, and at WC 4 and sequence 1A for a long-precision divide operation) if the floating-point zero result signal is present. In this case floating-point set zero HI is derived from AND block 5.

For a long-precision divide operation, the zero HI and zero LO latches must be activated to raise the line floating-point zero result. The zero HI latch is set from AND block 5 in the same way as for a short-precision divide operation. The zero LO latch is set by the floating-point set zero LO signal (which is derived from AND block 2) and with the B register zero result signal. The same variable-precision considerations are made as in the significance exception for the B register zero result analysis.

The FP divide exception latch is reset during a system reset or by the reset interrupt latches signal.

The handling of an FP divide exception causes the operation to be terminated.

Exponent Overflow Exception

- Is recognized when the result characteristic is greater than 127 after a floating-point add, subtract, multiply or divide operation.
- Is reset by the reset interrupt latches signal.

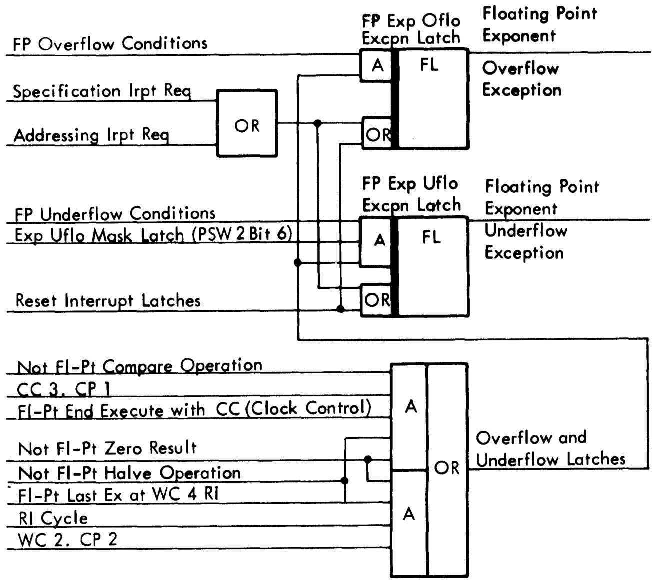
An exponent overflow exception is recognized if the result characteristic in floating-point add, subtract, multiply or divide operations exceeds the characteristic capacity (127). As the characteristic is always carried in excess 64 arithmetic, this value of greater than 127 actually corresponds to an exponent greater than plus 63.

In order to understand the operation of the overflow exception detection, a knowledge of exponent handling is necessary. Refer, therefore, to "Exponent Underflow and Overflow Detection" in this chapter and to the following FEMD Figures:

6319 to 6340	(FP add, subtract, compare)
6345 and 6346	(FP short multiply)
6357 to 6368	(FP long multiply)
6369 to 6374	(FP short divide)
6383 to 6394	(FP long divide)
6395 and 6396	(FP common multiply)
6397 to 6400	(FP common divide)

Figure 2-14 shows the logic circuits that set the exponent overflow exception latch. The floating-point overflow conditions signal also participates in condition code setting for these operations.

The exponent overflow exception latch is reset during a system reset or by the reset interrupt latches signal.



Refer also to ALD page KK 024

Figure 2-14. Exponent Overflow and Underflow Exceptions

The handling of the exponent overflow exception allows the instruction to be completed.

Exponent Underflow Exception

- Can be masked by PSW 2 bit 6 when the bit equals zero.
- Is recognized when the result exponent is less than zero after floating-point add or subtract normalized, multiply or divide operations.
- Is reset by the reset interrupt latches signal.

An exponent underflow exception is generated if the result characteristic in floating-point add or subtract normalized, multiply or divide operations is less than zero. As the characteristic is always carried in excess 64 arithmetic, this value of less than zero corresponds in practice to an exponent of less than minus 64. Since the characteristic is decreased only by left-shifting the fraction, add or subtract unnormalized operations cannot give this exception.

In order to understand fully the way in which this exception detection operates, a knowledge of exponent handling is necessary. Refer, therefore, to "Exponent Underflow and Overflow Detection" in this chapter, and to the following FEMD Figures:

6319 to 6340	(FP add, subtract, compare)
6345 and 6346	(FP short multiply)
6353 and 6354	(FP long multiply)
6357 to 6368	(FP long multiply)
6369 to 6374	(FP short divide)
6383 to 6394	(FP long divide)
6395 and 6396	(FP common multiply)
6397 to 6400	(FP common divide)

LOAD AND STORE INSTRUCTIONS

Load Instructions

- Operand 2 is placed in the first operand location in the form specified by the instruction:

Load:	Unchanged.
Load and test:	Unchanged.
Load complement:	Sign inverted.
Load positive:	Sign made positive.
Load negative:	Sign made negative.

- The condition code is set from the result for load and test, load complement, load positive and load negative as follows:
  - 00: Result fraction is zero.
  - 01: Result fraction less than zero (not load positive).
  - 10: Result fraction greater than zero (not load negative).
  - 11: Not set.
- The 'load' instruction is used for either short-precision or long-precision operands in either RR or RX formats.
- Load and test, load complement, load positive and load negative are RR instructions for either short-precision or long-precision operands.

The timings and sequences applied to the load instructions for short-precision operands and long-precision operands are shown in FEMD Figures 6303 to 6310.

Short Precision

The operand 2 field is read out of storage (RX format, EA cycle) or gated from the FPR defined by the Rb register (RR format) and set into the B register.

The op code and sign are then analyzed and, where necessary, the B register bit 00 is gated inverted to the output bus. Inversion of the B register bit 00 is necessary for the load complement instructions, for load positive instructions when bit 00 is a 1 and for load negative instructions when bit 00 is a 0. The B register is then gated to the operand 1 FPR defined by Ra.

The fraction bits 08 to 31 are analyzed and, if zero, cause the zero HI latch to be set for the load positive, load negative, load complement or load and test instructions.

This zero HI latch is then analyzed in conjunction with the B register gated sign to determine the setting of the condition code for these instructions.

For the 'load' instructions (LE, LER), the condition code is not set.

Long Precision

The operand 2 high-order field is read out of storage (RX format, EA cycle) or gated from the FPR defined by the Rb register (RR format) and set into the B register. The fraction bits 08 to 31 are analyzed in the B register and, if zero, set the zero HI latch for the load complement, load positive, load negative, and load and test instructions. The B and BX registers are then interchanged.

The operand 2 low-order word is fetched and set into the B register. The low-order bits are truncated to the length defined by the variable-precision switch on the SDR to B register transfer. If the low-order bits in the B register are zero, the zero LO latch is set for the load complement, load positive, load negative and load and test instructions.

The B and BX registers are interchanged and the high-order word of the result is gated to the high-order half of the operand 1 FPR defined by the Ra register. The B and BX registers are then re-interchanged and the low-order word is gated from the B register to the SDR, from where it is stored in the low-order half of the operand 1 FPR defined by the Ra register. The op code and B register bit 00 are analyzed and, if necessary, bit 00 is gated to the output bus in inverted form. The sign is inverted on the load complement instruction, on the load negative instruction when B00 is a zero and on the load positive instruction when B00 is a one.

This gated sign bit and the zero HI and zero LO latches are analyzed for the load complement, load negative and load and test instructions to determine the condition code setting.

Store Instructions

- Operand 1 is stored at the operand 2 storage location.
- The instruction format is RX.
- Operands may be either short-precision or long-precision.
- The condition code is not altered by this instruction.

The timing and sequences involved in the 'store' instruction for both short-precision and long-precision operands are shown in FEMD Figures 6311 to 6314.

Short Precision

Operand 1 is to be stored in the operand 2 location. For this operation an EA cycle is initiated and the B and BX registers are interchanged, leaving the B register at a reset value.

The operand 1 high-order bits are gated from the FPR defined by the Ra register and set to the B register. The bits are then gated from the B register to the SDR, from where they are stored into the operand 2 location on the write section of the EA cycle.

Long Precision

The effective address is temporarily stored out of the main data flow by a B to BX register interchange and the A register is reset in preparation for receiving the operand 1 data.

The operand 1 low-order bits are read out of the operand 1 FPR low-order bits in storage on an R1 cycle and are gated from the SDR to the A register, truncating low-order bits to the value defined by the variable-precision switch.

The operand 1 high-order bits are gated to the B register from operand FPR high-order bits. The B and BX registers are then interchanged and the effective address in the B register is used to address the storage during the double EA cycle which is then called.

A re-interchange of the B and BX registers occurs and the high-order bits of operand 1 in the B register are transferred to the SDR and stored in the effective address high-order bit location during the write section of the first phase (EAH) of the double EA cycles.

At the second phase (EAL) of the double EA cycle, the B register is reset and the low-order bits in the A register are gated via the ABC funnel to the B register. The B register is then gated to SDR and the low-order bits are stored in the effective address low-order bit location during the write section of the double EA cycle, second phase.

HALVE INSTRUCTIONS

- Operand 2 is halved by right-shifting the fraction bits one binary position, leaving the characteristic unaltered.
- The result is stored in the operand 1 FPR.
- Instructions are in the RR format and operands may be either long-precision or short-precision.
- The condition code is not altered by this operation.

For details of the sequence and timing of halve instructions, refer to FEMD, Figures 6315 to 6318.

Short Precision

The fraction of operand 2 is prepared for the halving process by gating bits 08 to 31 of operand 2 from the FPR (defined by the Rb register) to the B register via the HW and ABC funnels.

The sign and characteristic field of operand 2 is the sign and characteristic of the result and so this field (bits 00 to 07) of the operand 2 FPR is gated via the exponent funnel to exponent registers A and B. These bits enter exponent register B and are EXOR'ed with the ones that are set into this register by the 'clear data flow' signal during instruction-fetch (I-fetch).

As the result sign and characteristic are assembled with the exponent register B contents, an interchange between exponent registers A and B is then performed.

The fraction bits in the B register are halved by a shift-right-one operation and the result assembled by gating the exponent register B to bits 00 to 07 of the B register via the HW and ABC funnels. The result is then gated from the B register to the operand 1 FPR defined by the Ra register.

Long Precision

The handling of the operand 2 sign and characteristic is performed in the same manner as with short-precision operands. However, storage cycles are needed to fetch and store the low-order bits of the FP operands.

An R2 cycle is called to read out to the SDR the low-order bits of the operand 2 fraction. The high-order bits are gated in parallel to the B register from bits 08 to 31 of the FPR defined by the Rb register. Bits 00 to 07 of this FPR are then gated to exponent registers A and B and the registers are then interchanged; this places the result (operand 2) sign and characteristic in exponent register B.

The B and BX registers are interchanged and the operand 2 fraction (read out to the SDR on the R2 cycle) is gated to the B register. Truncation, as defined by the variable-precision switch, occurs on this transfer.

The B and BX registers are then re-interchanged and the fraction halved by a shift-right-one operation of the B and BX registers. The B and BX registers are then again interchanged and the low-order fraction bits are stored during an R1 cycle into the FPR defined by the Ra register.

The B and BX registers are again re-interchanged and the result high-order bits are assembled by gating exponent register B via the HW and ABC funnels to bits 00 to 07 of the B register.

The result high-order bits in the B register are then stored in the operand 1 FPR high-order bits by gating from the B register to the FPR defined by the Ra register.

Note that for long-precision operands, when the variable-precision switch is set for truncation, the result may contain one unwanted bit. This bit is stored but will be lost during truncation when the operand is next fetched.

ADD AND SUBTRACT INSTRUCTIONS (SHORT-PRECISION OPERANDS)

- Operand 2 is added to or subtracted from operand 1.
- The result, normalized if specified by the op code, is placed in the operand 1 location.
- Both RR and RX formats are used.
- The condition code is set to indicate the form of the results.
    - 00: Result fraction is zero or exponent underflow.
    - 01: Result fraction is less than zero.
    - 10: Result fraction is greater than zero.
    - 11: Result exponent overflow.

The floating-point add and subtract instructions are executed in the following steps:

1. Fetching of operands.
2. Matching of characteristics (exponents) and addition of fractions.
3. Recomplementing of fraction result (where necessary).
4. Correction of fraction overflow and normalization of result (where specified by op code).
5. Storage of result and setting of condition code.

This sequence applies to both long-precision and short-precision operands, but the operations involved in each step depend on the operand length.

Use the following description in conjunction with FEMD Figures 6319 to 6330.

Step 1 - Fetching of Operands

- The operand 2 fraction and characteristic are fetched.
- Operand 1 characteristic only is fetched.
- Operand 2 fraction is placed in bits 08 to 31 of both the A and BX registers.
- The partial difference of the second characteristic minus the first characteristic is formed in exponent register B.
- The partial difference is analyzed to set the operand 1 exponent HI latch, operand 1 exponent LO latch or exponents EQ latch.
- The shift counter is set to seven.

The add and subtract instructions can be in either RR or RX format, giving two distinct types of operand-fetch cycles.

Operand-fetch (RX format)

Operand 2 is located at the word defined by the effective address developed in the B register during I-fetch, and operand 1 is located in the FPR defined by the Ra register.

An EA cycle is taken first and operand 2 is read out from storage to the SDR. The B register is cleared of the effective address by a B to BX register interchange and a reset of the BX register. Operand 2 is then gated from the SDR and set into the A, B and C registers.

The characteristic of operand 1 is gated from the FPR to the exponent register B, and the operand 2 characteristic is gated from the A register and set into both exponent registers A and B.

Exponent register B is set to all ones during I-fetch and this register performs an EXOR function of its contents and input data. Therefore, at this time, exponent register A contains the operand 2 characteristic and sign, and exponent register B contains the one's complement of the EXOR of the characteristics and signs.

This partial sum in exponent register B is analyzed in conjunction with the operand 2 characteristic in exponent register A to set the operand 1 exponent HI latch, operand 1 exponent LO latch or exponents EQ latch. These latches indicate the value of the operand 1 characteristic relative to the operand 2 characteristic; the latches are used to determine whether characteristic matching is required and, if so, which of the operands is the smaller.

The logic used to set the appropriate latch is shown in Figure 3-1. The analysis logic is similar to the exponent CLA logic (described in Chapter 2) as, at the time of this analysis, the exponent register B contains the partial difference of the two characteristics (EXOR of two characteristics EXOR'ed with ones). If this partial difference in exponent register B is all ones, then the two characteristics (and therefore exponents) are equal and the exponents EQ latch is set.

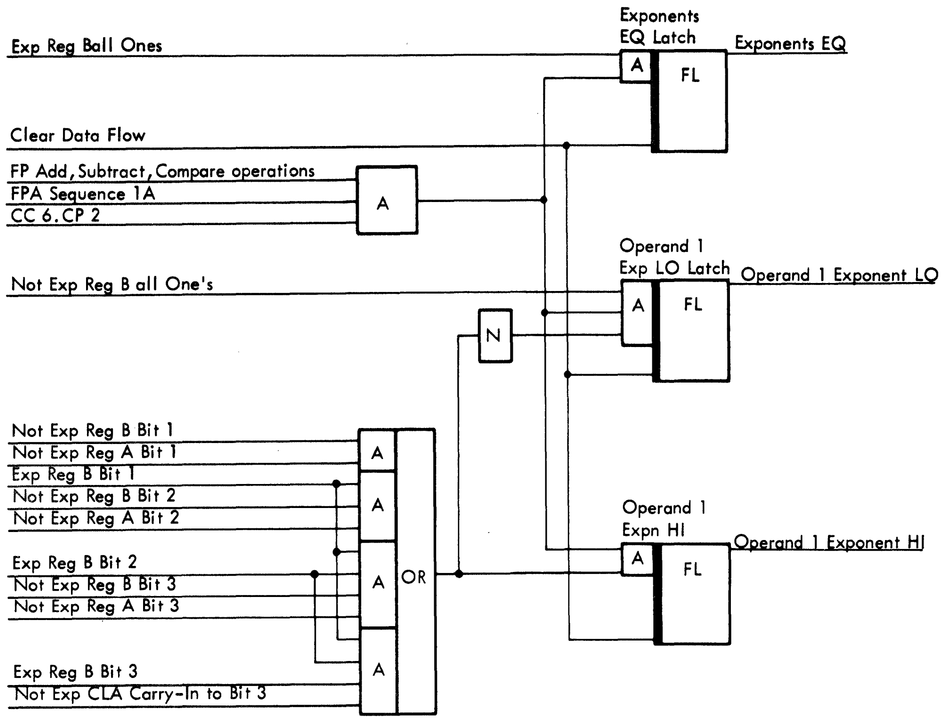


Figure 3-1. Operand 1 Exponent HI, LO and EQ Latches

At the time of analysis the A, B and C registers all contain operand 2 (both fraction and characteristic). At the concluding portion of the analysis cycle, a B to BX register interchange is performed and the operand 2 characteristic portion (bits 00 to 07) of both the A and BX registers is reset. This leaves the operand 2 fraction in bits 08 to 31 of both the A and BX registers. The B register now contains all zeros. The shift counter is set to seven for use in subsequent operations, while the complement add latch is set if complement arithmetic (add operation and operand signs unequal, or subtract operation and operand signs equal) is required. The logic associated with the complement add and recomplement latches is shown in Figure 3-2.

Operand-fetch (RR Format)

Operands 1 and 2 are in FPR's and a storage-fetch cycle is not required. The operand 2 fraction is gated out of the FPR defined by the Ra register and set (via the ABC funnel) to the A, B and C registers. The B and BX registers are then interchanged, leaving the fraction section of operand 2 in bit positions 08 to 31 of both the A and BX registers and all zeros in the B register.

The characteristic section of operand 2 is gated to the exponent funnel and set into exponent registers A and B. The characteristic of operand 1 is then gated from the FPR defined by the Ra register to the exponent funnel and set into exponent register B.

Therefore, as with the RX instructions, exponent register B contains the one's complement of the EXOR of the two characteristics (including their signs). This partial result is again used to set the operand 1 exponent HI latch, operand 1 exponent LO latch or exponents EQ latch.

The shift counter is set to seven for use in subsequent operations.

Step 2 - Matching of Characteristics and Adding of Fractions

- Characteristic 2 minus characteristic 1 is formed in exponent register B.
- Characteristics are already matched for the exponents EQ condition.
- For the operand 1 exponent LO and operand 1 exponent HI cases, the fraction of the smaller operand is shifted right until either the characteristics are equal or seven hexadecimal shifts have been performed.
- The fractions are added or subtracted in the B register. The result fraction is as follows:

Operand 1 exponent HI	:	Operand 2 ± Operand 1
Operand 1 exponent LO Operand EQ	} :	Operand 1 ± Operand 2

- The result characteristic is set in exponent register B.

- The result fraction is tested for zero result to set the zero HI latch.

As a result of the analysis performed during the operand-fetch operation, the comparison of characteristics sets the operand 1 exponent HI latch, operand 1 exponent LO latch or exponents EQ latch.

If either the operand 1 exponent HI latch or operand 1 exponent LO latch is set, the two characteristics must be matched by shifting right the smaller number until either the exponents become equal or the fraction is shifted out of the operand field (B register plus guard digit).

For the exponents EQ condition, the characteristics are already equal and only fraction addition is required.

Operand 1 Exponent HI

The operand 1 exponent HI latch indicates that the value of operand 1 is higher than that of operand 2. Operand 2 must be shifted right, therefore, until the characteristics are matched.

The operand 2 fraction is placed in the B register by a B to BX register interchange and operand 2 is cleared from the A register by an A to AX register interchange.

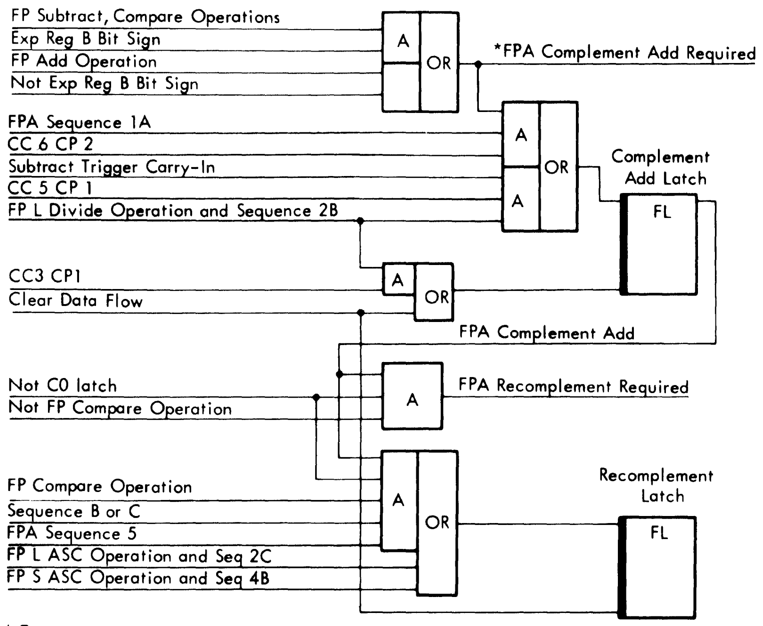
The A register is loaded with the operand 1 fraction by gating bits 08 to 31 of the FPR defined by the Ra register to the ABC funnel. At this time, the A register is conditioned and the B register is deconditioned to allow access to the A register only.

The exponent CLA output is gated to exponent register B via the exponent funnel, forming 'characteristic 2 minus characteristic 1'. In this case, the difference is in complement form since the characteristic of operand 1 is greater than the characteristic of operand 2.

The matching of characteristics begins by repeated shift-right-four operations on the B register, increasing the value of exponent register B and reducing the value of the shift counter until either becomes zero.

If the shift-counter becomes zero, all seven digits of the fraction have been shifted out of the fraction field, indicating that operand 2 was too small to alter operand 1. If exponent register B becomes zero, the characteristics have been matched. The operand 1 characteristic in either case is the characteristic of the result and, after exponent register A has been reset, this characteristic is gated from the operand 1 FPR to exponent register A. It is then placed into exponent register B by an exponent register A to B interchange.

Bits 04 to 31 of the BX register are reset to provide the guard digit for any subsequent normalization. If the complement latch is on, the A register is inverted and common add control is turned on. This action adds or subtracts operand 1 to or from operand 2, the result being contained in the B register. The zero HI latch is set if this result (including the guard digit for normalized operations) is zero.



* Turns on:
Sequence 4B, short-precision add, subtract, compare operations
Sequence 2C, long-precision add, subtract, compare operations

Figure 3-2. FPA Complement and Recomplement Latches

Operand 1 Exponent LO

The operand 1 exponent LO latch indicates that the value of operand 1 is lower than that of operand 2. Operand 1 must be shifted right therefore, until the characteristics are matched.

The exponent CLA is gated to exponent register B via the exponent funnel, forming the result of 'characteristic 2 minus characteristic 1' in the exponent register B. This value indicates the number of shifts that are required to equalize exponents.

The operand 1 fraction is prepared for shifting by gating it from the FPR defined by the Ra register to the B and C registers via the ABC funnel.

The operation of matching the characteristic then begins with repeated shift-right-four operations on the B register and decrementing of both the shift counter and exponent register B on each shift cycle until either becomes zero.

If the shift counter becomes zero, all seven digits of the fraction have been shifted out of the fraction field, indicating that operand 1 was too small to alter operand 2. If exponent register B becomes zero, the characteristics have been equalized. The operand 2 characteristic in either case is the characteristic of the result and is placed in exponent register B by an exponent register A to B interchange.

The BX register bits 04 to 31 are reset, leaving bits 00 to 03 as a guard digit. This digit provides one more significant digit in the low-order position of short operands if normalization is required.

The A register is inverted if the complement latch is on, and the 'common add control' is again switched on. This action adds or subtracts the operand 2 fraction to or from the operand 1 fraction, the result being contained in the B register. The zero HI latch is set if this result fraction is zero. The characteristic result is again formed in the exponent register B.

Exponents EQ

The fraction from operand 1 is set to the ABC funnel and gated to the B and C registers. As the B register was zero and the C register contained operand 1, a 'dc set B register into C register' operation was performed.

Depending on the operation and the operand signs, the A register is inverted if required and the 'common add control' is switched on. This makes active the gates necessary to add or subtract operand 2 to or from operand 1; these gates are: gate A reg to ABC funnel, de-condition C reg, and gate CLA to ABC funnel. The A register is inverted if complement arithmetic is to be performed (complement latch on).

In the exponent arithmetic register area, the exponent CLA output is gated to exponent register B giving, in this condition, a result of zero. The result characteristic is the same as the characteristic of either of the original operands. The exponent register A contains the operand 2 exponent and this is placed into exponent register B by an exponent register interchange.

At the completion of this step in the operation, the two fractions have been added with the fraction result contained in the B register, and the characteristics have been analyzed with the characteristic result being contained in the exponent register B.

The fraction result (including the guard digit for normalized operations) is tested for all zeros. If this condition exists, the zero HI latch is set.

End Conditions

The B register contains the result of the fraction addition or subtraction. In the operand 1 exponent HI case, this is the operand 2 fraction plus or minus the operand 1 fraction, whereas in the operand 1 exponent LO and exponents EQ cases, the result is the operand 1 fraction plus or minus the operand 2 fraction. These contents of the register are used in sign development. In all cases, the characteristic of the result is contained in exponent register B.

If the B register bits 04 to 07 are not equal to zero, a fraction overflow has occurred and the 'shift-right-four required' latch is set. If the B register content is in complement form, the result must be recomplemented prior to storing the result. If the B register is in true form, these recomplement cycles are skipped.

<u>Step 3 - Recomplementing of Fraction Result</u>

- This cycle is required if the fraction result is in complemented form.
- The condition is detected by a no-carry-out from the B register during a complement add operation.
- For normalized operations, the fraction is shifted left one digit to include the guard digit in the recomplementing process.
- Recomplementation inverts B and C registers by using all one's output of the true/criss-cross.
- A shift right is required if the fraction overflows into the exponent field on the recomplement cycle.

If complement arithmetic was performed on the fractions and the result of the fraction arithmetic is in complemented form, a recomplementing cycle is required.

This condition is detected as shown in Figure 3-2 by a no-carry-out from the B register during the complement add operation. The no-carry-out causes the 'FPA recomplement required' signal to be generated which permits sequence 4B to be called, for the recomplementing process. The fact that the cycle has been taken is stored in the recomplement latch for subsequent use in result sign-analysis circuits.

For the recomplement cycle, the A register is reset and a shift-left-four operation of the B and BX register is executed. The shifting occurs so that the guard digit from bits 00 to 03 of the BX register is included in the recomplementing process. The shift-left-four is coupled with a decrement by one of the result characteristics in exponent register B.

The fraction to be recomplemented is now in the B register. The recomplementing process is accomplished by dc setting the B register to the C register, inverting the B and C registers by gating ones from the true/criss-cross into both registers and bringing up the common add control. Note that the subtract trigger will be on and will provide a CLA carry-in for this cycle.

Bits 00 to 07 of the result in the B register are analyzed for a fraction overflow into the characteristic field. If an overflow has occurred, the 'shift-right-four required' latch is set. This latch is also set for all un-normalized operations to shift right the guard digit back into the BX register.

Step 4 - Correcting of Fraction Overflow and Normalizing of Result

- The result fraction overflow is corrected if the 'shift-right-four required' latch is on.
- For all the following conditions, normalizing cycles are required:
  - If the op code specifies normalized results.
  - The high-order digit of the fraction is zero (B register bits 08 to 11).
  - The result fraction is not all zeros.
  - The 'shift-right-four required' latch is off.
- Shift-left-four operations are performed until the high-order digit of the fraction becomes a significant digit.
- The result characteristic is decremented by one on each shift left of the fraction.

When the 'shift-right-four required' latch is on, the fraction is corrected by a shift-right-four of the B and BX registers with a corresponding increment of the exponent register B.

A series of shift-left-four operations of the B and BX registers is performed when normalization has been specified by the op code, the result in the B register is not equal to zero, the 'shift-right-four required' latch is off and the high-order digit of the fraction (bits 08 to 11 of the B register) is zero.

A corresponding decrementing by one of exponent register B is also performed for each shift, and the process is continued until bits 08 to 11 of the B register contain a significant digit.

Step 5 - Storing of Result and Setting of Condition Code

- The characteristic result is transferred from exponent register B to bits 00 to 08 of the B register.
- The B register is stored in the first operand FPR.
- The sign bit is analyzed and, where necessary, corrected by gating the B register bit 00 in inverted form to the output bus.
- The condition code is set.
- FP exception latches are set if an exception which is not masked has occurred.

The fraction section of the result is now in bits 08 to 31 of the B register, and the characteristic of the result is in exponent register B.

To form the complete result, exponent register B is gated via the ABC funnel and set into bits 00 to 07 of the B register. This result is then gated from the B register to the FPR specified by the Ra register.

As the sign bit now in the B register bit 00 was transferred from exponent register B, it is the sign of the operand that was in the A register for the arithmetic operation. This is the operand that was complemented and, as such, is not necessarily the sign of the result. An analysis is required, therefore, to determine if the B register sign bit is the correct result sign.

Analysis of B Register Sign Bit

Four conditions exist under which the sign bit in the B register is incorrect and therefore needs to be gated in inverted form to the B register output bus (Figure 3-3). Note that 'exponent register A bit sign' is the inverted EXOR of the operand sign and is used as a control for generating sign inversion. The sign bit in the B register represents the operand that is complemented (that is, the operand 1 sign for the operand 1 exponent HI case and the operand 2 sign for the exponents EQ and operand 1 exponent

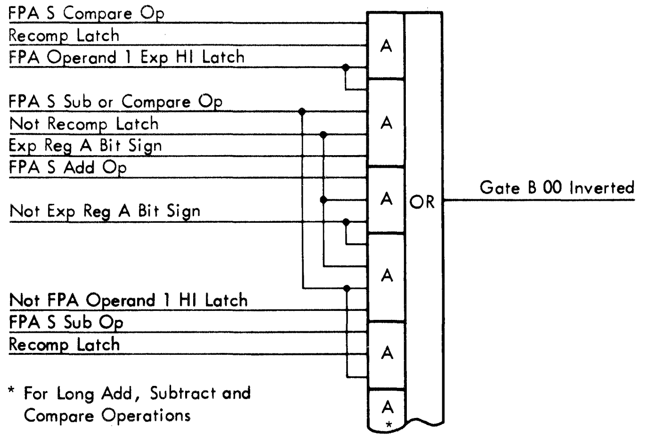


Figure 3-3. Logic for Gate B00 Inverted on Short-Precision Add, Subtract, Compare Operations

LO cases). The sign of the result is the sign of the arithmetic operation of operand 1 plus or minus operand 2 after the characteristics have been matched.

Condition 1: For the add operation when complement arithmetic is performed (add operation with unlike signs). If no recomplement is required, the result sign is the sign of the operand that was not complemented and the inverse of the sign of the operand that was complemented. Thus the B register sign bit 00 is gated in inverted form onto the B register output bus if this condition exists at the end of the operation.

Condition 2: For the subtract operation when complement arithmetic is performed (like signs) and the operand 1 exponent HI latch is on. If no recomplement cycle is required, it means that the sign of the result is the inverse of the operand signs since, in the operand 1 exponent HI case, operand 2 minus operand 1 is formed in the B register; the result of the operation should be operand 1 minus operand 2.

Condition 3: For the subtract operation for the exponents EQ or operand 1 exponent LO cases when a recomplement is required. In both cases, operand 1 minus operand 2 is formed in the B register and, if a recomplement is required, operand 2 must have been greater than operand 1. Thus the result sign must be the inverse of the operand 2 sign in the B register.

Condition 4: For the subtract operation, with either exponents EQ or operand 1 exponent LO conditions, where the original signs were different. As with the third condition, operand 1 minus operand 2 is formed as a result and the sign in the B register is the sign of operand 2. As complement arithmetic is not performed, the result sign must be that of operand 1 and the inverse of operand 2. Thus, for this case, the B register sign bit 00 is again gated in inverted form onto the B register output bus.

Setting of Condition Code

The fraction result in the B register bits 08 to 31 and the gated B register bit 00 are used to set the condition code as follows:

00: Result fraction is zero (zero HI latch on).
01: Result fraction is less than zero (gated B register bit 00 = 1; zero HI latch off).
10: Result fraction is greater than zero (gated B register bit 00 = 0; zero HI latch off).
11: Result exponent overflow.

When the intermediate sum is zero and the significance mask is a one, a significance exception is signalled and a program interrupt is taken. No normalization occurs and the intermediate sum characteristic value remains unchanged. The sign of the zero result is forced to positive.

When the intermediate sum is zero and the significance mask bit is a zero, the program interrupt is masked off and the result is set to a true zero by resetting the B register prior to storing the result.

If the exponent overflows during the subtraction of exponents, an overflow exception is signalled and a program interrupt is taken.

If normalization causes the exponent to underflow, both the characteristic and fraction are made zero and if the corresponding program mask bit is a one, an underflow exception is signalled and a program interrupt is taken.

Examples of Short-Precision Add/Subtract

Example 1

Figure 3-4 shows an example of floating-point addition, including characteristics matching and guard digit preservation. The following operands which give an operand 1 exponent LO condition are used:

First Operand. + 2,890 (dec) = + B4A (hex)

Shown in a non-normalized short-operand format:

$$+ 0.00B4A \times 16^{5}$$

Shown in System/360 binary notation:

Sign	Characteristic	Fraction
0	1000101	0000 0000 1011 0100 1010 0000

Second Operand. + 27,776 (dec) = + 6C80 (hex)

Shown in a non-normalized short-operand format:

$$+ 0.0006C8 \times 16^{7}$$

Shown in System/360 binary notation:

Sign	Characteristic	Fraction
0	1000111	0000 0000 0000 0110 1100 1000

To facilitate the notation of the fraction, and since fraction handling can be associated with the basic

Exponent Arithmetic Binary Digits		Exp Reg B S C 1 - - - - - 7	Exp Reg A S C 1 - - - - - 7
		0 0 0000000	1 1 1111111
	State A		
Opnd 1 Exp to Exp Reg B		↓	(0 - 1000101)
		0 0 0000000	1 1 0111010
Opnd 2 Exp to Exp Reg A and B		0 0 1000111	(0 - 1000111
		0 0 1000111	1 1 1111101
	State B		
Exp CLA Output			(- 0 1111111)
Exp Funnel Output		↓	(- 0 1111111)
		0 0 1000111	1 1 0000010
	State C		
Decrement Exp Reg B	a	0 0 1000111	1 1 0000001
Decrement Exp Reg B	b	0 0 1000111	1 1 0000000
	State D		
Interchange Exp Reg's		1 1 0000000	0 0 1000111
	State E		
Normalization Only.	c	1 1 0000000	0 0 1000110
Decrement Exp Reg B	d	1 1 0000000	0 0 1000101
	e	1 1 0000000	0 0 1000100
	State F		

Fraction Arithmetic	A Reg	AX Reg	C Reg	B Reg	BX Reg
Hex Digits	00 000000	00000000		00 000000	0 0000000
State A					
Opnd 2 to A and B Reg	47 0006C8		*	47 0006C8	0 0000000
Interchange B and BX Reg	47 0006C8			00 000000	4 70006C8
Reset A and BX Reg 00 to 07	00 0006C8			00 000000	0 00006C8
State B					
Gate Opnd 1 08 to 31 to B Reg			*	00 00B4A0	0 0000000
Reset BX Reg 04 to 31				00 00B4A0	0 0000000
State C					
Shift Right Four a			*	00 000B4A	0 0000000
Shift Right Four b			*	00 0000B4	A 0000000
State D					
Reset BX Reg 04 to 31				00 0000B4	A 0000000
Add A Reg to B Reg				(00 0006C8)	A 0000000
Set A Reg + Gate CLA				00 00077C	A 0000000
State E					
Normalization Only. c				00 0077CA	0 0000000
Shift Right Four d				00 077CA0	0 0000000
e				00 77CA00	0 0000000
State F					
Final Contents of B Reg	↓	↓		44 77CA00	0 0000000

* C Register and B Register contents are similar

Figure 3-4. Example of FP Normalized Addition (Opnd 1 Exp LO Condition)

arithmetic explanation, the following notation is adopted, showing the characteristics in binary and the fractions in hexadecimal:

Operand 1	0 1000101	00B4A0
Operand 2	0 1000111	0006C8

Refer to Figure 3-4 (where exponent and fraction arithmetic are shown separately) in conjunction with the following description of the state of registers at various steps during the operation.

State A may be considered as the state of the registers immediately before entering the operands. Exponent register B is set to all ones during I-fetch by 'clear data flow' and a carry-in exponent CLA is provided, since all operations (except multiply) call for a complement add of the characteristics. Note that, for speed requirements, the B register might be reset indirectly by interchanging with a blank BX register. The BX register is then reset at convenience.

In state B, the exponent registers are analyzed to set the operand 1 exponent HI, LO or EQ latches (LO is set in the example). The EXOR of the signs, inverted and represented by the exponent register B bit 'sign', is used to decide the type of operation

Exponent Arithmetic	Exp Reg A S C 1------7	Exp Reg B S C 1------7
Binary Digits	0 0 0000000	1 1 1111111
State A		
Opnd 1 Exp to Exp Reg B		(0 - 1000101)
Opnd 2 Exp to Exp Reg A and B	0 0 1000100	(0 - 1000100)
	0 0 1000100	1 1 1111110
State B		
Exp CLA and Exp Funnel output	0 0 1000100	(- 0 0000001)
		1 1 1111111
Reset Exp Reg A	0 0 0000000	1 1 1111111
State C		
		(0 0 1111111)
Increment Exp Reg B	0 0 0000000	1 1 0000000
State D		
Opnd 1 Exp to Exp Reg A	0 0 1000101	1 1 0000000
Interchange Exp Reg's	1 1 0000000	0 0 1000101
State E		
Decrement Exp Reg B	1 1 0000000	0 0 1000100
State G		

Fraction Arithmetic	A Reg	AX Reg	C Reg	B Reg	BX Reg
Hex Digits	00 000000	00000000		00 000000	0 0000000
State A					
Opnd 2 to A and B Reg	44 23C900		*	44 23C900	0 0000000
Interchange B and BX Reg	44 23C900			00 000000	4 423C900
Reset A and BX Reg 00-07	00 23C900			00 000000	0 023C900
State B					
Int'ch A, AX and B, BX Reg	00 000000	0023C900		00 23C900	0 0000000
Opnd 1 bits 08 to 31 to A Reg	00 04B050			00 23C900	0 0000000
State C					
Invert A Reg (+ Subt Trigger)	FF FB4FAF			00 023C90	0 0000000
Shift Right Four B and BX			*	00 023C90	0 0000000
State D					
Add A Reg to B Reg				(FF FB4FAF)	
+ Carry-In (Subt Trigger)				(00 000001)	
Set A Reg, Gate CLA				FF FD8C40	0 0000000
State E					
Reset A Reg and Shift Left Four B and BX Reg	00 000000		*	FF D8C400	0 0000000
State G					
Invert B, C Reg (Gate T/XC)			*	00 273BFF	
Add A Reg to B Reg				(00 000000)	
+ Carry-In (Subt Trigger)				(00 000001)	
Set A Reg, Gate CLA				00 273C00	
State H					
Final Contents of B Reg				44 273C00	

* C Register and B Register contents are similar

Figure 3-5. Example of FP Complement Add with Recomplement (Opnd 1 Exp HI Condition)

(true add in this case). The bits 00 to 07 of the A and BX registers are also reset to avoid interference with the fraction during processing.

The characteristic difference is obtained in state C. At this time, the fraction of the correct operand is in the B, BX registers for shifting until the characteristics are matched (that is, until the exponent difference in exponent register B is zero). Since the operand 1 exponent LO latch is on in this case, the operand 1 fraction is adjusted and the characteristic result is decreased by one on each shift-right-four operation. For the same reason, the BX register bits 04 to 31 are reset since during shift-right-four operations, these bits loop into the B register leftmost positions (FEMD Figure 1001).

The matching of the characteristics is terminated in state D and the common add signal performs a normal add between the A register and the B register; that is, the A register is EXOR'ed with the B register contents, the C register is de-conditioned to avoid feed-back and the CLA is gated to the B register. Since operand numbering does not necessarily correspond, care must be taken if reference is made to the basic add operation. While the fraction result is formed, the exponent registers are interchanged since the only path out of the exponent data flow is from exponent register B and the correct exponent (operand 2 exponent in this case) is located in exponent register A.

The FP add operation is completed after state E by gating the exponent register B to the B register, where it is combined with the fraction (not shown in the example). In this case the guard digit is lost. However, if the op code specifies a normalized result, the normalizing cycles occur immediately after state E.

Figure 3-4 shows, at the end of the add operation example (following at state E), the three steps that perform the normalization of this specific result. The result exponent is adjusted with the fraction manipulation. At the end of the normalization (state F), the exponent register B is placed in front of the fraction in the B register, in preperation for storing the result.

The 'gate B00 inverted' line is not active in this example and the result in the B register is gated directly to the operand 1 FPR.

Example 2

Figure 3-5 shows an example of floating-point subtraction including the matching of characteristics with operand 1 exponent HI conditions, and recomplementation, with the following operands:

First Operand. + 19,205 (dec) = + 4B05 (hex)

Shown in a non-normalized short-operand format:

$$+ 0.04B05 \times 16^{5}$$

Shown in System/360 binary notation:

Sign	Characteristic	Fraction
0	1000101	0000 0100 1011 0000 0101 0000

Second Operand. + 9,161 (dec) = + 23C9 (hex)

Shown in a normalized short-operand format:

$$+ 0.23C9 \times 16^{4}$$

Shown in System/360 binary notation:

Sign	Characteristic	Fraction
0	1000100	0010 0011 1100 1001 0000 0000

To facilitate the notation of the fraction, and since fraction handling can be associated with the basic arithmetic explanation, the following notation is adopted, showing the characteristics in binary and the fractions in hexadecimal.

Operand 1	0 1000 101	04B050
Operand 2	0 1000 100	23C900

Refer to Figure 3-5 (where a floating-point subtraction is made and exponent and fraction arithmetic are shown separately) in conjunction with the following description of the state of registers at various steps of the operation.

State B is reached in the same way as in the previous example. The analysis that compares the exponents sets the operand 1 exponent HI latch in this case. At the same time, the exponent register B bit 'sign' is inspected. Since the bit is 1, the signs are equal and, as a subtract instruction is being performed, the complement add latch is set.

The exponent arithmetic progresses normally, but since operand 1 characterisitc is the result characteristic, the operand 2 characteristic is cleared from exponent register A. For the same reason, the operand 2 fraction is set again in the B register since it will have to be shifted to match the characteristics. The A register is cleared and operand 1 fraction is set in the A register.

In state C, the exponent register B contains the difference between exponents, while the correct operand is in the B register for shifting.

With state D, the matching of the characteristics is completed and the A register has been inverted as the first step to perform the complement add. The second step (carry-in) is achieved by gating the CLA with the subtract trigger on.

The complement add result is obtained in state E and exponent register B contains the characteristic result. However, there was no carry-out of the CLA during the fraction add (C0 latch off); therefore, the complemented operand (operand 1) is greater than operand 2 and the result is in complement form.

Since the fraction is always stored in true form, a recomplement process is required. Note that if a carry-out of the CLA had occurred, the subtract operation result (being in true form) can be stored after the characteristic has been gated to the B register.

For the recomplementing process the fraction result, in complement form, is shifted left four if it is a normalized operation, to include the guard digit in the recomplementation. If it is an unnormalized operation, the guard digit is lost and no shifting takes place.

In state G, the first step in deriving the two's complement of the B register contents is to invert the contents. The second step is to reset the A register to allow usage of the CLA with a carry-in (subtract trigger). Since the A register has been reset, the floating-point common add signal is used without restrictions.

The result obtained in state H is thus in true form (the two's complement of the complement result). Since no further normalization is required and no fraction overflow has occurred, the exponent register B is combined with the fraction in the B register to form the complete floating-point subtract result. The B register bit 00 is gated out of the B register in true form during the storing process.

COMPARE INSTRUCTIONS (SHORT-PRECISION OPERANDS)

- Operand 1 is compared with operand 2, and the condition code indicates the result of this comparison.
- Comparison is arithmetic, the sign, fraction and characteristic being compared.
- The rules of normalized floating-point subtraction are used to establish the operand difference.
- Instruction format can be either RR or RX.
- The condition code setting is as follows:
  - 00: Operands are equal
  - 01: Operand 1 is low
  - 10: Operand 1 is high
  - 11: Not set

The compare operation follows the rules for floating-point un-normalized subtraction and differs from that operation in only four major areas:
A recomplement cycle is never taken.
The arithmetic result of the subtraction is not stored.
The meaning and setting of the condition code.
Exponent overflow, exponent underflow or significance exceptions cannot cause a program interrupt.

The compare instruction is thus performed in the following steps:

1. Fetching of operands.
2. Matching of characteristics and subtraction of fractions.
3. Analysis of result and setting of condition code.

This sequence applies to both long-precision and short-precision operands, but the operations involved in each step depend on the operand length. Refer to FEMD Figures 6319 to 6330 for the sequences of the short-precision compare instruction.

Step 1 - Fetching of Operands

- The operand 2 fraction and characteristic are fetched.
- The operand 1 characteristic only is fetched.
- The operand 2 fraction is placed in bits 08 to 31 of both A and BX registers.
- The partial difference of the operand 2 characteristic minus the operand 1 characteristic is formed in exponent register B.
- This partial difference is analyzed to set the operand 1 exponent HI, operand 1 exponent LO or exponents EQ latches.
- Dc sets the shift counter to seven.

Fetching of operands is the same for short-precision add, subtract and compare instructions.

Step 2 - Matching of Characteristics and Subtracting of Fractions

- Characteristic 2 minus characteristic 1 is formed in exponent register B.
- For the exponents EQ case, the exponents already match.
- For the operand 1 exponent LO and operand 1 exponent HI cases, the fraction of the smaller operand is shifted right until either the characteristics are equal or seven shifts have been performed.
- The fractions are subtracted and the result is contained in the B register.
- The result characteristic is set into exponent register B.

The operation is the same for add, subtract and compare instructions.

Note that if the result of the compare subtraction is in complement form, the recomplement latch is turned on. However, the signal to start a recomplement cycle is degated by the compare op. The recomplement latch is used once again for determining the correct sign of the result. Refer to Figure 3-1 for this logic.

The B register bits 00 to 07 are all ones if the recomplement latch is turned on. For this condition, therefore, the sign is inverted and the characteristic is in one's complement form when it is transferred from exponent register B.

Step 3 - Analysis of Result and Setting of Condition Code

- The characteristic result (with sign) is transferred from exponent register B to the B register.
- The B register sign bit is checked, and when it is not correct, is gated in inverted form to the B register output bus.
- The B register sign and the zero HI latch are analyzed to set the condition code.

The fraction section of the result is contained in the B register and the characteristic of the result in exponent register B. As with the add and subtract instructions (Step 5), exponent register B is transferred to bits 00 to 07 of the B register. The sign of the B register is the sign of the operand that was originally in the A register, and does not necessarily represent the true sign of the result.

The analysis is similar to that used for the add and subtract instructions (Figure 3-3).

The sign in the B register is the same as for the subtract operation, except when the recomplement latch is on, in which case it is the inverse of subtract.

Thus the logic used in the subtract operation for the second and fourth conditions described previously in "Add and Subtract Instructions" (Step 5), is also activated for the compare operation. For the remaining condition (that is, when the recomplement latch is on) the sign must be inverted on the operand 1 exponent HI condition.

The result fraction in the B register and the sign gated onto the B register output bus are analyzed with the same logic as in "Add and Subtract Instructions" (Step 5) to set the condition code as follows:

00: Operands are equal (zero HI on)
01: Operand 1 is low (gated B register bit 00 = 1, zero HI off)
10: Operand 1 is high (gated B register bit 00 = 0, zero HI off)
11: Not set.

Note that program interrupts for significance exceptions or for exponent underflow or overflow cannot occur on a compare instruction.

ADD AND SUBTRACT INSTRUCTIONS (LONG-PRECISION OPERANDS)

- Operand 2 is added to or subtracted from operand 1.
- The result, normalized if specified by the op code, is placed in the operand 1 location.
- Both RR and RX formats are used.
- The condition code is set to indicate the form of the results.
  00: Result fraction is zero or exponent underflow
  01: Result fraction is less than zero
  10: Result fraction is greater than zero
  11: Result exponent overflow.

The floating-point add and subtract instructions are executed in the following steps:

1. Fetching of operands.
2. Matching of characteristics (exponents) and addition of fractions.
3. Recomplementing of fraction result (where necessary).
4. Correction of fraction overflow and normalization of result (where specified by the op code).
5. Storage of result and setting of condition code.

This sequence applies to both long-precision and short-precision operands, but the operations involved in each step depend on the operand length.

The sequence for long-precision add and subtract instructions are shown in FEMD Figures 6331 to 6340. Only the major objectives of each of the above steps, and any points requiring explanation, are contained in the following description.

The effect of the variable-precision feature on this operation is that all fetched operands have their low-order fraction bits truncated at the ABC funnel to the number of digits defined by the setting of the variable-precision console switch. The truncated positions are replaced by zeros and each operand is then handled as a long-precision operand. That is, 14 digits in the fraction participate in all operations, including characteristic alignment and result normalization cycles.

The result may have significant bits in the digits outside the range of the defined precision. These bits are stored as part of the result and are seen if the result register is displayed in the console lights. However, when this result is fetched as an operand it is truncated at the ABC funnel. Thus, for programming purposes, the result can be considered to be stored in the truncated form.

Step 1 - Fetching of Operands

- The operand 2 fraction and characteristic are fetched.
- The operand 1 characteristic only is fetched.
- Low-order fraction bits are truncated to the precision set by the console switch.
- The operand 2 high-order fraction is placed in bits 08 to 31 of both AX and BX registers.
- Operand 2 low-order fraction bits are placed in the A register and the inverse of these bits in the B register.
- The partial difference of the second characteristic minus the first characteristic is formed in exponent register B.

- This partial difference is analyzed to set the operand 1 exponent HI latch, operand 1 exponent LO latch or exponents EQ latch.

- Dc sets the shift counter to 14.

- 'Complement add' latch is set when complement arithmetic is required.

- Registers are reset under control of the operand 1 exponent HI latch, operand 1 exponent LO latch or exponents EQ latch to prepare for the process of matching characteristics.

The long-precision add and subtract instructions may be in either RR or RX format and the type of operand-fetch cycles depends on the format. Note that a dc set of the BX register occurs which results in the low-order bits of the operand 2 fraction being left in inverted form. This form allows the fraction arithmetic to proceed with only one 32-bit inversion, depending on whether true or complement arithmetic is to be performed.

The shift counter is set to 14 as there are now 14 hexadecimal digits in the fraction, and more than 14 hexadecimal shifts in the alignment process means that the operand being shifted has lost significance.

Exponent register B contains the inverse of the EXOR of the two characteristics and is used to set the operand 1 exponent HI latch, operand 1 exponent LO latch or exponents EQ latch. Depending on which latch is turned on, the registers are prepared for the alignment cycles.

Operand 1 Exponent HI: Operand 2 to be shifted in characteristic matching process.

A and AX registers:	Reset
B register :	Inverse operand 2 fraction low-order bits
BX register :	Operand 2 fraction high-order bits.

Exponents EQ: No alignment process.

Operand 1 Exponent LO: Operand 1 to be shifted in characteristic matching process.

A and AX registers:	Operand 2 fractions bits 8 to 63.
B and BX registers:	'Complement add' latch on; set to ones. 'Complement add' latch off; reset to zeros.

Step 2 - Matching of Characteristics and Adding of Fractions

- Characteristic 2 minus characteristic 1 is formed in exponent register B.

- For the exponents EQ case, the characteristics already match.

- For the operand 1 exponent LO and operand 1 exponent HI cases, the fraction (or inverse of the fraction if the 'complement add' latch is on) of the smaller operand is shifted right until either the characteristics are equal or 14 hexadecimal shifts have been performed.

- The fractions are added or subtracted in two cycles; the carry from the low-order bits is used to set the subtract trigger carry-in for the high-order bits.

- The result in the B and BX registers is as follows:

Operand 1 exponent HI :	Operand 1 ± operand 2.
Exponents EQ, Operand 1 Exponent LO :	Operand 2 ± operand 1.

- The result exponent is set in exponent register B.

- The result fraction is tested to set the zero HI and zero LO latches.

- A fraction overflow into the characteristic field on a true add operation sets the 'shift-right-four required' latch.

- A complement result causes a 'FPA recomplement required' signal (Figure 3-2).

- If the 'FPA recomplement required' signal is on, a recomplement cycle is signalled; if the signal is off, the recomplement cycles are skipped.

For the operand 1 exponent HI condition, the shift-right-four operation for matching characteristics is performed with the operand 2 low-order bits in the B register and the operand 2 high-order bits in the BX register. For this operation, the BX to B shift-right-four path is conditioned. Shifting of bits 00 to 05 of the BX register is de-conditioned.

The foregoing action has the effect of shifting the fraction down the BX register, across into the high-order part of the B register and down the B register. Bits shifted out of the low-order position of the B register are lost. As the BX register bits 00 to 07 at this time will be either all ones or all zeros, the effect of de-conditioning the shift input to bits 00 to 05 of this register is to ensure that ones or zeros respectively will be shifted into the high-order positions of the fraction.

Step 3 - Recomplementing of Fraction Result

- The cycle is taken when the fraction result is in complement form.
- This condition is detected by no-carry-out during the complement add cycles.
- Recomplementing is made in two halves: the fraction result low-order bits, followed by the fraction result high-order bits.
- The 'shift-right-four required' latch is turned on if the result of the recomplement cycle overflows into the characteristic field.

The logic for recomplementing the fraction result is shown in Figure 3-2.

Step 4 - Correction of Fraction Overflow and Normalization of Result

- If the 'shift-right-four required' latch is on, a shift-right-four operation is taken and the exponent register B is incremented by one.
- Normalizing cycles are required if the op code specifies normalization, the high-order digit of the fraction is zero (B register bits 08 to 11) and the fraction is not all zeros.
- Shift-left-four operations (normalizing) are repeated until the high-order digit of the fraction becomes a significant digit.
- The result characteristic in the exponent register B is decremented by one on each shift left of the fraction.
- If an exponent underflow condition is caused by the normalizing process, both the characteristic and fraction of the result are set to zero.

Step 5 - Storage of Results and Setting of Condition Code

- The result characteristic is transferred from exponent register B to bits 00 to 08 of the B register.
- The result in the B and BX registers is stored in the operand 1 FPR.
- The sign bit is analyzed and, where incorrect, is corrected by gating the B register bit 00 in inverted form to the output bus.
- The condition code is set.
- FP exception latches are set if an exception has occurred which is not masked.

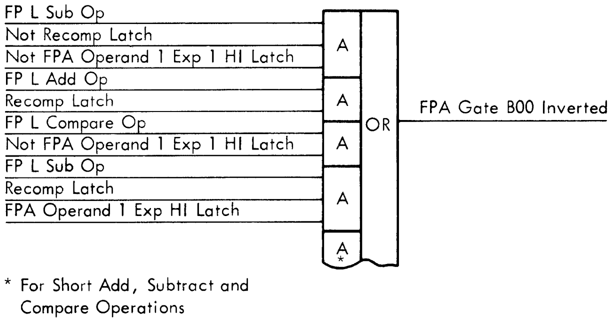
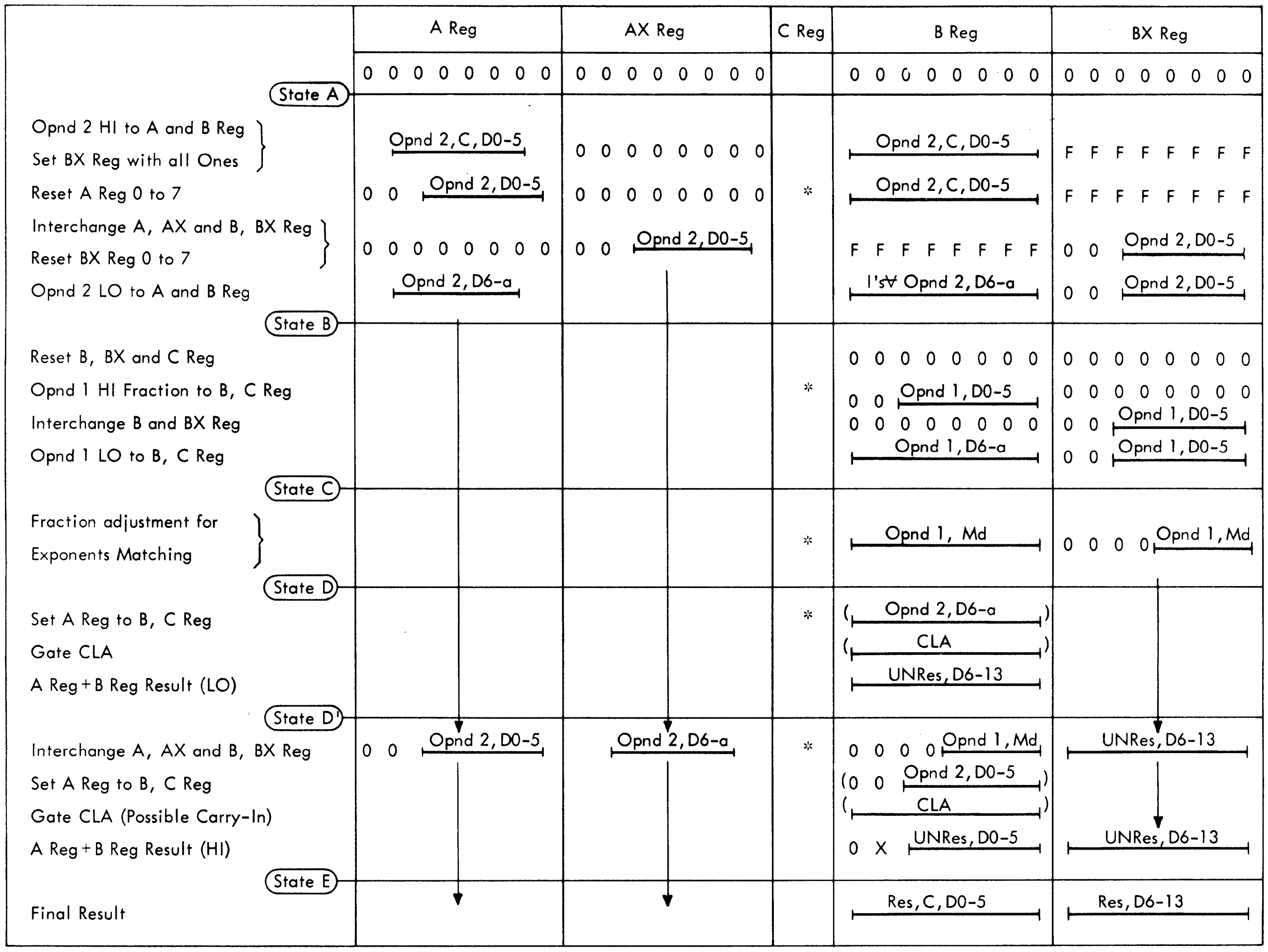

FP L Sub Op
Not Recomp Latch
Not FPA Operand 1 Exp 1 HI Latch
FP L Add Op
Recomp Latch
FP L Compare Op
Not FPA Operand 1 Exp 1 HI Latch
FP L Sub Op
Recomp Latch
FPA Operand 1 Exp HI Latch
A
A
A
A
A*
OR
FPA Gate B00 Inverted
* For Short Add, Subtract and Compare Operations

Figure 3-6. Logic for Gate B00 Inverted on Long-Precision Add, Subtract, Compare Operations

Figure 3-6 shows the logic for gating the B register bit 00 in inverted form for the long-precision add, subtract (and compare) instructions.

The fraction result and the gated B register sign are used to set the condition code as follows:

00:	Result fraction is zero (zero HI and zero LO on) or exponent underflow.
01:	Result fraction is less than zero (either zero HI or zero LO off, and gated B register bit 00 = 1).
10:	Result fraction is greater than zero (either zero HI or zero LO off, and gated B register bit 00 = 0).
11:	Result exponent overflow.



	A Reg	AX Reg	C Reg	B Reg	BX Reg
	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
State A					
Opnd 2 HI to A and B Reg / Set BX Reg with all Ones	Opnd 2,C,D0-5	0 0 0 0 0 0 0 0		Opnd 2,C,D0-5	F F F F F F F F
Reset A Reg 0 to 7	0 0 Opnd 2,D0-5	0 0 0 0 0 0 0 0	*	Opnd 2,C,D0-5	F F F F F F F F
Interchange A, AX and B, BX Reg / Reset BX Reg 0 to 7	0 0 0 0 0 0 0 0	0 0 Opnd 2,D0-5		F F F F F F F F	0 0 Opnd 2,D0-5
Opnd 2 LO to A and B Reg	Opnd 2,D6-a			1's⊻ Opnd 2,D6-a	0 0 Opnd 2,D0-5
State B					
Reset B, BX and C Reg				0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
Opnd 1 HI Fraction to B, C Reg			*	0 0 Opnd 1,D0-5	0 0 0 0 0 0 0 0
Interchange B and BX Reg				0 0 0 0 0 0 0 0	0 0 Opnd 1,D0-5
Opnd 1 LO to B, C Reg				Opnd 1,D6-a	0 0 Opnd 1,D0-5
State C					
Fraction adjustment for Exponents Matching			*	Opnd 1, Md	0 0 0 0 Opnd 1,Md
State D					
Set A Reg to B, C Reg			*	(Opnd 2,D6-a)	
Gate CLA				(CLA)	
A Reg+B Reg Result (LO)				UNRes, D6-13	
State D'					
Interchange A, AX and B, BX Reg	0 0 Opnd 2, D0-5	Opnd 2, D6-a	*	0 0 0 0 Opnd 1,Md	UNRes, D6-13
Set A Reg to B, C Reg				(0 0 Opnd 2, D0-5)	
Gate CLA (Possible Carry-In)				(CLA)	
A Reg+B Reg Result (HI)				0 X UNRes, D0-5	UNRes, D6-13
State E					
Final Result				Res,C,D0-5	Res, D6-13

* C Register and B Register contents are similar

Legend:

Opnd 1	Operand 1	D0-13	Hex Fraction Digits 0 to 13
Opnd 2	Operand 2	a	Precision Switch Specified Number
Res	Result	Md	Matched Fraction
C	Characteristic	UN	Un-normalized
⊻	EXOR'ed		

NOTE: The characteristic handling corresponds to the example in Figure 3-2

Figure 3-7. Example of Long-Precision Add (Opnd 1 Exp LO Condition)

Examples of Long-Precision Add/Subtract

Example 1

Figure 3-7 shows how the fractions are handled for two operations with exponents which give an operand 1 exponent LO condition. The lettered states match those of "Examples of Short-Precision Add/Subtract" (Example 1 and Figure 3-4); the explanation of the characteristic arithmetic is given in that example. The bits 00 to 07 of the operands used in the example are:

Operand 1 45 (hex)
Operand 2 47 (hex)

State A may be considered as the state of the registers immediately before entering the operands.

In state B, the exponent registers are analyzed to set the operand 1 exponent HI, operand 1 exponent LO or exponents EQ latches (operand 1 exponent LO latch is set in this example). A true add operation is also decided at the same time, since both fraction signs are positive.

The operand 2 fraction is in A, AX and B, BX registers. The high-order fraction part is in the AX and BX registers. The low-order parts (truncated to the variable-precision switch specifications) which must be processed first are in the A register, and in inverted form in the B register. These placings allow the requirements of any exponent HI, LO, or EQ latch condition or true-or-complement add operation to be met with the least possible data manipulation.

Since the operand 1 exponent LO latch is set, operand 1 must be shifted until the characteristics are matched. Therefore, operand 2 is unaltered in the A and AX registers, while the B, BX and C registers are reset. In state C, the operand 1 high-order fraction is in BX register and operand 1 low-order part (truncated as were those of operand 2) is in the B register. Both are ready for the shift-right-four steps used for the exponents matching. To allow shifting with the high part following the low-order part, the shift-right-four pulse is suppressed on BX register positions 00 to 05. Thus the B register last positions, which are the fraction low-order positions, are shifted out and lost. A link between positions 28 to 31 of the BX register and positions 00 to 03 of the B register is formed by the shift-right-four pulse acting on the B register bits 00 to 07. This pulse shifts the low digit of the high-order word into the high digit of the low-order word.

In state D, the matching is completed and the add operation starts with the common add signal which gives sequentially:

- Set A register to ABC funnel (to B and C registers)
- De-condition C register
- Gate CLA output to ABC funnel (to B register). This is a normal add operation of A and B registers.

The low-order fraction add result is obtained in state D', and the C0 latch (adder carry-out bit) state is set into the subtract trigger; the trigger is used to store an adder carry-out bit from the low-order sum, and, to provide a carry into the high-order sum by forcing a carry-in to the CLA. Because of the shifts used for matching fractions, the precision of the result is higher than that specified by the variable-precision switch.

The same add operation now takes place between the high-order fraction parts after A, AX and B, BX registers interchange, and with a carry-in provided. where necessary, by the subtract trigger.

The result obtained in state E might have an overflow out of the fraction field (X in Figure 3-7). In this case, a final shift-right-four occurs (accompanied by a corresponding incrementing of the characteristic). The shifting is done without the special provision made during the matching, since the fraction parts are now in the correct order.

In the case of a normalized operation, the eventual high-order fraction hexadecimal zeros are shifted out by shift-left-four pulses. The characteristic is corrected accordingly, the short-precision add operation (normalization) rules applying to the operation.

The final result, which is stored, is obtained in the next step by gating the characteristic (exponent register B) to the B register positions 00 to 07.

Example 2

Figure 3-8 shows an example of a basic floating-point subtract operation, with recomplementation of the result if it is obtained in complement form. The characteristics used and the lettered states correspond to those of the characteristic handling in "Examples of Short-Precision Add/Subtract" (Example 2 and Figure 3-5). That is, the operand 1 characteristic is 45 (hex) and the operand 2 characteristic is 44 (hex).

States A and B are reached in the same way as in the basic floating-point long-precision add operation.

Since the operand 1 exponent HI latch is set in this case, operand 2 must be shifted until the characteristics are matched. Operand 2 is thus left in the B and BX registers and operand 1 is set into A and AX registers. At the same time, a complement add operation is called (as the signs of the two operands are the same). Operand 2 low-order part is kept in inverted form in the B register. After the B and BX registers are interchanged (by gating the ones from the true/criss-cross to the B and C registers) the operand 2 high-order fraction is also inverted.

At state C, operand 2 is thus in B and BX registers (because of the operand 1 exponent HI latch) and is in inverted form (because of the complement add operation).

The same matching occurs as in the basic floating-point long-precision add operation, with looping of the fraction hex bits from the BX register bits 28 to 31 to the B register bits 00 to 03.

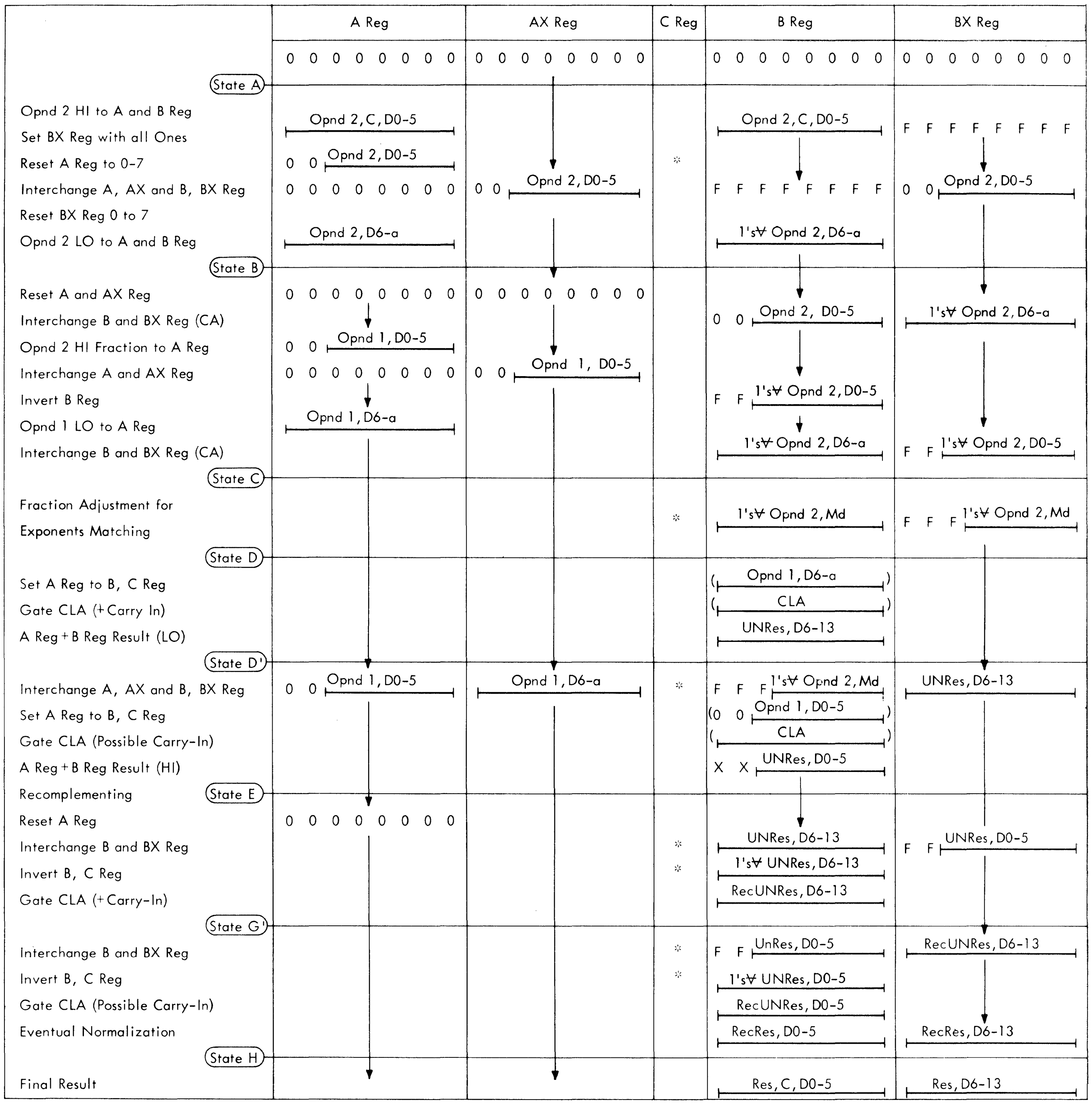
In state D, the 'common add' signal is raised to add in B register the operand 1 low-order fraction (A register) to the two's complement of the operand 2 low-order fraction.

In state D', after interchanging the A, AX and B, BX registers, the same process as in state D occurs between operand HI fractions, the carry-in CLA bit reflecting the condition of the C0 latch after the first addition.

In state E, if the result is in true form (C0 latch on), the operation is terminated as in the basic floating-point long-precision add operation.

If the result is not in true form (C0 latch off), a recomplementing process takes place. The A register is reset, ones are gated to the B and C registers, and the 'common add' signal gates the CLA to the B register. The B and BX registers are interchanged to derive first the two's complement of the low-order fraction, so that an eventual adder carry-out can be inserted as a carry-in to the high-order fraction. The result obtained in state E is the two's complement of the first low-order result.

In state G', the process is repeated with the high-order result fraction (after a B, BX registers inter-



* C Register and B Register contents are similar

Legend:

Opnd 1	Operand 1	D0-13	Hex Fraction Digits 0 to 13
Opnd 2	Operand 2	a	Precision Switch Specified Number
Res	Result	Md	Matched Fraction
C	Characteristic	UN	Un-normalized
∀	EXOR'ed		

NOTE: (CA) means the signal is associated with complement add.

Figure 3-8. Example of Long-Precision Complement Add with Recomplement (Opnd 1 Exp HI Condition)

change) with the subtract trigger (carry-in CLA) set to reflect the state of the C0 latch.

State H is reached with a normalized result if specified; note that, in recomplementing, it is not possible to have a fraction hexadecimal overflow. The operation ends by storing the final result which includes the characteristic gated from exponent register B.

COMPARE INSTRUCTIONS (LONG-PRECISION OPERANDS)

- Operand 1 is compared with operand 2, and the condition code indicates the result of this comparison.
- Comparison is arithmetic, the sign, fraction and characteristic being compared.
- The rules of floating-point subtraction are used to establish the operand difference.
- Instruction format can be either RR or RX.
- The condition code setting is as follows:

00	:	Operands are equal
01	:	Operand 1 is low
10	:	Operand 1 is high
11	:	Not set.

The compare operation follows the rules for floating-point un-normalized subtraction and differs from that operation in only four major areas:

A recomplement cycle is never taken.

The arithmetic result of the subtraction is not stored.

The meaning and setting of the condition code is different.

Exponent overflow, exponent underflow or significance exceptions cannot cause a program interrupt.

The compare instruction is thus performed in the following steps:

1. Fetching of operands.
2. Matching of characteristics and subtraction of fractions.
3. Analysis of result and setting of condition code.

This sequence applies to both long-precision and short-precision operands, but the operations involved in each step depend on the operand length. Only the major objectives of each of the above steps, and any points requiring explanation, are contained in the following description.

Refer to FEMD Figures 6331 to 6340 for the sequences of the long-precision compare instructions.

The effect of the variable-precision feature on this operation is that all fetched operands are truncated in a similar manner to that for the long-precision add and subtract instructions.

Step 1 - Fetching of Operands

- The operand 2 fraction and characteristic are fetched.
- The operand 1 characteristic only is fetched.
- The operand 2 high-order fraction is placed in bits 08 to 31 of both AX and BX registers.
- The operand 2 low-order fraction is placed in the A register and the inverse of these bits in the B register.
- The partial difference of the second characteristic minus the first characteristic is formed in exponent register B.
- This partial difference is analyzed to set the operand 1 exponent HI, operand 1 exponent LO or exponents EQ latches.
- Dc sets the shift counter to 14.
- The complement add latch is set when complement arithmetic is required.
- Registers are reset under control of the operand 1 exponent HI, operand 1 exponent LO or exponents EQ latches to prepare for the process of matching characteristics.
- Low-order fraction bits are truncated to the precision set by the console switch.

The fetching of operands for long-precision add, subtract and compare instructions is similar. The contents of the internal registers are the same as for the add and subtract operations.

Operand 1 Exponent HI: Operand 2 to be shifted in the process of matching characteristics.

A and AX registers: Reset.

B register: Inverse of operand 2 fraction low-order bits.

BX register: Operand 2 fraction high-order bits.

Exponents EQ: No characteristic-matching process.

Operand 1 Exponent LO: Operand 1 to be shifted in the process of matching characteristics.

A and AX registers: Operand 2 fraction bits 8 to 63

B and BX registers:	Complement add latch off; reset to zeros.
	Complement add latch on; set to ones.

Step 2 - Matching of Characteristics and Subtracting of Fractions

- Characteristic 2 minus characteristic 1 is formed in exponent register B.
- For the exponents EQ case, the characteristics already match.
- For the operand 1 exponent LO and operand 1 exponent HI cases, the fraction (or inverse of the fraction if the complement add latch is on) of the smaller operand is shifted right until either the characteristics are equal or 14 hexadecimal shifts have been performed.
- The fractions are subtracted in two cycles; the carry from the low-order bits is used to set the subtract trigger carry-in for the high-order bits.
- The result in the B and BX registers is as follows:
  Operand 1 exponent HI: operand 1 - operand 2.
  Operand 1 exponent LO, exponents EQ: operand 2 - operand 1.
- The result exponent is set in exponent register B.
- The result fraction is tested to set the zero HI, zero LO latches.
- A fraction overflow into the characteristic field on a true add operation sets the 'shift-right-four required' latch.
- A complement result sets the recomplement latch on.
- A recomplement cycle is inhibited on the compare instruction, but the recomplement latch remains on.

Note that if the result of the subtraction of the fractions is in complement form, the recomplement latch is turned on (Figure 3-2). However, the signal to start a recomplement cycle is degated by the compare operation. The recomplement latch is used as in long-precision add/subtract to determine the correct sign of the result.

If the recomplement latch is on, the B register bits 00 to 07 are all ones and the sign of exponent register B and the characteristic are inverted in the B register.

Step 3 - Analysis of Result and Setting of Condition Code

- The characteristic result (with sign) is transferred from exponent register B to B register bits 00 to 07.
- The B register sign bit is checked and, if it is incorrect, is gated in inverted form to the output bus.
- The B register gated sign and the zero HI and zero LO latches are used to set the condition code.

The sign in the B register is not necessarily the correct sign of the result. The logic for the 'gate B00 inverted' signal, for long-precision add, subtract and compare instructions, is shown in Figure 3-6.

The zero HI, zero LO latches and the B register gated sign are analyzed in the same manner as for long-precision add and subtract instructions. The condition code is set as follows:

00	: Operands are equal (zero HI and zero LO on).
01	: Operand 1 is low (either zero HI or zero LO off, and B register gated sign = 1).
10	: Operand 1 is high (either zero HI or zero LO off, and B register gated sign = 0).
11	: Not set.

Note that program interrupts cannot occur on a compare operation for significance exceptions, or for exponent underflow or overflow exceptions.

MULTIPLY INSTRUCTIONS (SHORT-PRECISION OPERANDS)

- The normalized long-precision product of operand 2 (multiplier) and operand 1 (multiplicand) is placed in the operand 1 FPR.
- Multiplication consists of a characteristic addition and a fraction multiplication.
- The sum of the characteristics less 64 is the characteristic of the intermediate product.
- The sign of the product is determined by the rules of arithmetic.
- The intermediate product fraction is 14 digits wide and is held in long-precision format.
- Pre-normalizing of the fractions is not necessary.
- The intermediate product is post-normalized.

- Exponent underflow and exponent overflow exceptions can occur.

- Significance exception is not tested.

The objective of the multiply operation is to take the two operands, multiply their fractions and add their characteristics (exponents).

The addition of the exponents is performed in the exponent register area, in parallel with the multiplication of the six hexadecimal digits of each fraction in the A, B, BX, C registers.

The multiplication of the fractions is performed in a similar manner to fixed-point multiplication (refer to "Fixed-Point Instructions," Principles of Operation - Processing Unit, Form Y33-0002). The fraction multiplication uses the same group-of-ones principle and the multiplication loop rules are also alike. Pre-normalizing of the fractions is not necessary because the full result can be contained in the B and BX registers.

When the fraction multiplication is complete, the fraction is post-normalized, combined with the result exponent and stored in long-precision format.

If the final product characteristic exceeds 127, an exponent overflow exception exists; the operation is therefore terminated and a program interrupt requested. Note that if the intermediate product characteristic exceeds 127, an overflow exception is not signalled should the result characteristic be subsequently reduced to 127 or less on the post-normalization process.

Exponent underflow occurs if the final product characteristic is less than zero. For this condition, both the characteristic and fraction of the product are made zero and an exponent underflow exception is signalled if the corresponding mask bit (PSW 2 bit 6) is a one.

When all 14 result fraction digits are zero, the product is made a true zero by forcing the product sign and characteristic to zero. For this condition, a significance exception is not signalled and the exponent overflow and underflow conditions are ignored. Thus a program interrupt for lost significance is not taken for a multiply operation.

The basic steps involved in the multiplication of short-precision operands can be summarized as follows:

1. Fetching of operands and initialization.
2. Fraction multiply cycles and addition of characteristics (exponents).
3. Post-normalization of result.
4. Storing the result.
5. Test for FP exceptions.

The instructional sequences and timings are shown in FEMD Figures 6341 to 6346.

Step 1 - Fetching of Operands and Initialization

- Operand 2 fraction (multiplier) is set to the A register.

- Operand 1 fraction (multiplicand) is set to the BX register.

- Operand 2 characteristic is set to exponent register A.

- The EXOR of the characteristics of operand 1 and operand 2 is formed in the exponent register B.

- Dc sets the shift counter to 25.

- A multiplier of zero sets the zero HI latch.

- A multiplicand of zero sets the zero LO latch.

The initial setup of the multiplier and multiplicand fraction is similar to the fixed-point multiply operation. The operand 2 fraction (multiplier) is set to bits 08 to 31 of the A register and the operand 1 fraction (multiplicand) is set to bits 08 to 31 of the BX register. Bits 00 to 07 of both the A and BX registers are zero at the end of the setup cycles. If the multiplier is zero, the zero HI latch is set; if the multiplicand is zero, the zero LO latch is set.

The characteristics of the two operands are combined in the exponent register area in preparation for the addition of the exponents.

Exponent register B (which was set to ones during I-fetch) is reset and the characteristics are gated to the exponent registers, so that exponent register B contains the EXOR of the two characteristics and exponent register A contains one of the characteristics in preparation for the characteristic addition process.

The shift counter is set to a value of 25 as the fraction multiplication concerns only 24 bits.

Step 2 - Fraction Multiply Cycles and Addition of Characteristics (Exponents)

- The fractions are multiplied.

- The addition of characteristics is completed.

- The basic rules of fixed-point multiplication apply to the fraction multiplication.

- Characteristic addition is performed using excess 64 arithmetic.

- For a multiplier of zero (zero HI), the multiply cycles are omitted.

The fraction multiplication is treated as two 24-bit positive numbers and the rules of fixed-point multiplication apply. The group-of-ones principle is used, while the type and length of cycle performed depends on the status of the multiplier digits in bits 30, 31 of the BX register and the subtract trigger backup latch. However, since the fraction is always in true form, a positive sign is assumed and no special provision is made for the last cycle on which the shift counter becomes zero. Refer to "FP Common Multiply Cycles" in this chapter for a description of the common multiply loop.

The characteristic addition is completed by gating the exponent CLA output to exponent register B to form the sum of the two characteristics; the exponent CLA high-order position is gated for excess 64 arithmetic so that the sum developed is the arithmetic sum of the two characteristics less 64. This arithmetic sum represents the result exponent plus 64, which is the result characteristic.

If the zero HI latch is on (multiplier of zero which would give a zero result), the multiply cycles and the majority of the characteristic handling are omitted. Refer to FEMD Figures 6344 and 6346.

Step 3 - Post-Normalization of Product

- The result of the fraction multiplication is normalized.
- Normalizing of the product is omitted if either the multiplier or multiplicand is zero.

The result fraction is normalized by repeated shift-left-four operations on the B and BX registers until a significant digit occurs in the high-order position of the fraction (B register bits 08 to 11).

For each shift-left-four operation, exponent register B is decremented by one to preserve the correct value of the result.

Step 4 - Storing the Result

- The result characteristic is combined with the result fraction.
- For exponent underflow, the result is set to zero.
- For a zero result fraction, the result characteristic is set to zero.
- The full double-word product is stored in FPR (Ra).

Normally, the complete result is formed in the B and BX registers by gating exponent register B to bits 00 to 07 of the B register. If the result fraction is zero, however, the gating does not occur and a true zero result is formed in the B register.

If an exponent underflow has occurred, the B register is reset and a result of zero is stored in both halves of the FPR defined by the Ra register.

For normal conditions, the long-format result in the B and BX registers is stored in the FPR defined by the Ra register, as shown in FEMD Figure 6368.

Step 5 - Test for FP Exceptions

- Exponent overflow condition sets the exponent overflow exception latch and a program interrupt is requested.
- Exponent underflow condition sets the exponent underflow exception latch and requests a program interrupt, provided that the mask bit (PSW 2 bit 6) is a one.
- For a multiplier or multiplicand of zero, exponent overflow or underflow conditions are ignored.
- A significance exception is not tested.

For conditions other than a zero multiplier or a zero multiplicand, the exponent overflow and exponent underflow conditions are tested. For an exponent overflow, the exponent overflow exception latch is set; this latch then generates a program-interrupt request. For an exponent underflow, the mask bit (PSW 2 bit 6) must be a one to allow the exponent underflow exception latch to set; this latch, when set, requests a program interrupt.

INTRODUCTION TO MULTIPLY INSTRUCTIONS (LONG-PRECISION OPERANDS)

The multiplication of the two operands follows the rules that exponents are added and fractions are multiplied.

However, execution of fraction multiplication is complicated by the length of each fraction, which can vary from 32 to 56 bits (8 to 14 hexadecimal digits) dependent on the setting of the variable-precision switch. The normal multiplication process handles up to 32 bits and special provision has been made to develop products from fractions greater than 32 bits. Regardless of the fraction length, the maximum length of the product is 56 bits (14 hexadecimal digits).

The manner in which the machine handles these long-precision fractions depends upon the precision chosen by the variable-precision switch.

Principles of Long-Precision Multiply

- Multiplication is performed in four steps for precisions of 10, 12 and 14.
- Multiplication is performed in one step for a precision of 8.
- Fixed-point multiplication rules apply to each of the multiply steps.

For the multiply operation, each operand is truncated to the desired precision during fetching, prenormalized, then shifted-left eight bit positions so that the fraction is aligned to the left-hand end of the registers.

The fractions are then split into high-order and low-order sections. The high-order section always contains 32 bits (8 hexadecimal digits); the low-order section contains 0, 8, 16 or 24 bits (0, 2, 4 or 6 hexadecimal digits) according to the setting (8, 10, 12 or 14 respectively) of the variable-precision switch.

The multiplication is then performed in four subroutines when the precision is either 10, 12 or 14. For a precision of 8, the first three subroutines are skipped and only the fourth subroutine is executed. These four subroutines are shown in Figure 3-9, where all alphabetic figures represent 32 binary bits.

Subroutine 1: The low-order bits are multiplied, with the B register set to zero, thus forming a product, of which the low-order half (J) is discarded. The high-order half (H) is temporarily retained in the B register for subroutine 2.

Subroutine 2: The first cross-multiplication is performed and the product is developed on top of the

Sub-routine	Description	Operation Summary	Operation
	Operand 2	High-order = D	
		Low-order = E	D E
	Operand 1	High-order = F	x
		Low-order = G	F G
1	Form Low-order product.	(G x E) =	H J
	Truncate unwanted bits.		H -
2	Form first cross-multiply product		
	on result of subroutine 1.	H + (G x D) =	K L
	Store K for subroutine 3.	L + (E x F) =	- L
3	Form second cross-multiply product		
	on result of subroutine 2.	L + (E x F) =	M N
	Truncate unwanted bits.		M -
	Add K (from subroutine 2) to M.	(K + M) =	P -
4	Form high-order product		
	on result of subroutine 3.	P + (D x F) =	Q R
	Result Product		Q R

Figure 3-9. Principles of Long-Precision Multiply

information (H) left in the B register. The high-order half (K) of the developed product is temporarily stored for later use and the low-order half (L) is left in the B register for subroutine 3.

Subroutine 3: The second cross-multiplication is performed and the product is developed on top of the information (L) left in the B register. The low-order half (N) of this developed product is discarded and the information (K) temporarily stored in subroutine 2, is added to the high-order half (M) of the product. The sum (P) is left in the B register for subroutine 4. If there is a high-order carry on this add process, the fact is stored in a latch for subroutine 4.

Subroutine 4: The high-order multiplication is performed and the product (Q # R) is developed on top of the information (P) left in the B register. On the last multiply cycle, a one is added to the product if a high-order carry was present in subroutine 3.

The result in the B and BX registers represents the fraction result of the complete multiply operation which is combined with the result exponent to form the complete result.

EXECUTION OF LONG-PRECISION MULTIPLY (PRECISION = 10, 12, 14)

The execution of the long-precision FP multiply instruction involves the four multiply subroutines (described previously) and data manipulation preparatory to these subroutines.

The instruction is performed in the following basic steps:

1. Operand fetch and pre-normalization.
2. Low-order product.
3. First cross-product.
4. Second cross-product.
5. High-order product.
6. Post-normalization, exception test and result store.

Figure 3-10 shows the contents of each of the main registers at the single-cycle points. Each cycle is defined by the controlling cycle latch or sequence latch(es). The legend used to define the register contents is shown, as are the constants used to define values which are dependent on the setting of the variable-precision switch.

Note that manipulations may occur within a cycle which cause the contents of a register to alter during the effective part of the cycle, then to be further altered, reset or replaced as necessary. For cycles where multiplication takes place (sequences 2B, 5B, 1D, 2D), only the contents at the end of the set of multiply cycles are shown.

Expressions are shown at the bottom of Figure 3-10 to indicate how the four products (P1 to P4)

Cycle	A Reg	AX Reg	B Reg	BX Reg	Scratch Reg	Exp Reg A	Exp Reg B	Shift Counter
RR Format only								
Seq 1A	Opnd 2,C,D0-5	-	-	-	-	-	Opnd 2,C	-
R2 Cycle	Opnd 2,C,D0-5	-	Opnd 2,C,D0-5	Opnd 2,D6-a	-	-	Opnd 2,C	-
RX Format only								
EA,EA,1A	Opnd 2,C,D0-5	-	Opnd 2,C,D0-5	Opnd 2,D6-a	-	-	Opnd 2,C	-
5A	-	-	Opnd 2N,D0-7	Opnd 2NL,D8-a	-	-	Opnd 2,CN	-
2A	Opnd 2NL,D8-a	-	Opnd 2N,D0-7	Opnd 2NL,D8-a	Opnd 2NL,D8-a	-	Opnd 2,CN	-
R1	Opnd 2N,D0-7	Opnd 2NL,D8-a	Opnd 1,D6-a	-	Opnd 2N,D0-7	Opnd 1,C	Opnd 1,C+2,CN	-
3A *	Opnd 2N,D0-7	Opnd 2NL,D8-a	Opnd 1R,D0-5	Opnd 1,D6-a	-	Opnd 1,C	Opnd 1,C+2,CN	-
4A *	Opnd 2N,D0-7	Opnd 2NL,D8-a	Opnd 1NR,D0-6	Opnd 1NL,D7-a	-	Opnd 1,C	Opnd 1,CN+2,CN	-
1B	Opnd 2N,D0-7	Opnd 2NL,D8-a	-	Opnd 1NL,D8-a	Opnd 1NL,D8-a	Opnd 1,C	P,C	b
3B	Opnd 2NL,D8-a	Opnd 2N,D0-7	-	Opnd 1NR,D8-a	Opnd 1NL,D8-a	Opnd 1,C	P,C	c
End 2B's	Opnd 2NL,D8-a	Opnd 2N,D0-7	P1,D0-7	-	Opnd 1NL,D8-a	Opnd 1,C	P,C	0
4B *	Opnd 2N,D8-a	Opnd 2N,D0-7	Opnd 1NL,D8-a	P1,D0-7	Opnd 1NL,D8-a	Opnd 1,C	P,C	b
3B	Opnd 2N,D0-7	Opnd 2N,D8-a	Opnd 1NR,D8-a	P1R,D0-7	Opnd 1NL,D8-a	Opnd 1,C	P,C	c
End 5B's	Opnd 2N,D0-7	Opnd 2NL,D8-a	P2,D0-7	-	P2,D8-a	Opnd 1,C	P,C	0
1C	Opnd 2NL,D8-a	Opnd 2N,D0-7	Opnd 2NL,D8-a	P2,D0-7	P2,D8-a	Opnd 1,C	P,C	0
2C	P2,D8-a	Opnd 2N,D0-7	Opnd 2NL,D8-a	P2,D0-7	-	Opnd 1,C	P,C	0
3C	P2,D8-a	Opnd 2N,D0-7	-	Opnd 2NL,D8-a	P2,D0-7	Opnd 1,C	P,C	b
5C	P2,D8-a	Opnd 2N,D0-7	P2,D8-a	Opnd 2NL,D8-a	P2,D0-7	Opnd 1,C	P,C	c
4C	Opnd 1N,D0-7	Opnd 2N,D0-7	P2R,D8-a	Opnd 2NR,D8-a	P2,D0-7	Opnd 1,C	P,C	c
End 1D's	Opnd 1N,D0-7	Opnd 2N,D0-7	P3,D0-7	P3,D8-a	P2,D0-7	Opnd 1,C	P,C	0
3D	P2,D0-7	Opnd 2N,D0-7	P2+P3,D0-7	-	P1,D0-7	Opnd 1,C	P,C	33
4D *	Opnd 2N,D0-7	P2,D0-7	Opnd 1N,D0-7	P2+P3,D0-7	P1,D0-7	Opnd 1,C	P,C	33
End 2D's	Opnd 2N,D0-7	P2,D0-7	P4,D0-7	P4L,D8-13	P1,D0-7	Opnd 1,C	P,C	0
5D	Opnd 2N,D0-7	P2,D0-7	P4N,C,D4-5	P4N,D6-13	P1,D0-7	Opnd 1,C	P,CN	0
R1	Opnd 2N,D0-7	P2,D0-7	P4N,D6-13	P4N,CN,D0-5	P1,D0-7	Opnd 1,C	P,CN	0

* Cycle CC 1 to CC 4 only

Register Contents Legend:

Opnd 1	: Operand 1	D0-13	: Hex Fraction Digits 0 to 13
Opnd 2	: Operand 2	C	: Characteristic and Sign
F	: Fraction	L	: Digits Left Aligned in Register
N	: Normalized	R	: Digits Right Aligned in Register
P	: Product	a,b,c	: Constants; see table below

Product Derivation:

P1 : (Opnd 2NL,D8-a) x (Opnd 1NR,D8-a)
P2 : P1,D0-7 + (Opnd 2N,D0-7) x (Opnd 1NR,D8-a)
P3 : P2,D8-a + (Opnd 1N,D0-7) x (Opnd 2NR,D8-a)
P4 : P2+P3,D0-7 + (Opnd 2N,D0-7) x (Opnd 1N,D0-7)

Constants Values:

	Precision		
	10	12	14
Constant a	9	11	13
Constant b	5	3	1
Constant c	9	17	25

Figure 3-10. Single-Cycle Chart for Long-Precision Multiply (Precision = 10, 12, 14)

are derived from the data existing at the start of the multiply cycles. Derivation:

Product (B, BX registers) = Multiplicand (A register) x Multiplier (BX register).

Refer also to the corresponding instruction sequence diagrams (FEMD Figures 6347 to 6368) to enable the manipulation within a cycle to be determined.

Step 1 - Operand Fetch and Pre-Normalization

- Cycles are from the end of I-fetch to the end of sequence 4A.
- At the end of sequence 4A, the operand 2 fraction (normalized and left-aligned to bit 00) is in the A and AX registers.
- At the end of sequence 4A, the operand 1 fraction (normalized and left-aligned to bit 04) is in the B and BX registers.
- The sum of the two characteristics is formed in exponent register B.

Sequence 1A, R2 Cycle (RR Format)

Operand 2 high-order bits (characteristic and fraction digits 0 to 5) are gated from the FPR defined by the Rb register and are set to the A and B registers. The B and BX registers are then interchanged.

Operand 2 low-order bits (fraction digits 6 to 13) are read out of the operand 2 FPR, truncated to the precision defined by the variable-precision switch and set into the B register. The B and BX registers are then interchanged.

Operand 2 characteristic is gated from the operand 2 FPR defined by the Rb register to exponent register B.

If operand 2 is zero, the second operand zero latch is set on. Refer to "Setting of Zero HI and LO Latches".

Double EA Cycles, Sequence 1A (RX Format)

Operand 2 high-order bits are read out of the storage location defined by the effective address and set into the A and BX registers. Operand 2 low-order bits are read out of storage, truncated to the precision defined by the variable-precision switch and set into the B register. The B and BX registers are then interchanged.

Operand 2 characteristic is set into exponent register B from bits 00 to 07 of the A register.

If operand 2 is zero, the second operand zero latch is set on. Refer to "Setting of Zero HI and LO Latches".

Sequence 5A

The double-word operand 2 fraction is normalized in the B and BX registers and the resulting fraction is shifted-left two hex digit positions. This leaves digits 0 to 7 of the normalized fraction in the B register and the remaining digits of the fraction in the BX register. The number of digits remaining in the BX register is a function of the variable-precision switch and is 2, 4 or 6 for precisions of 10, 12 or 14 respectively.

As the digits in the BX register are aligned to the left-hand or high-order end of the register, the contents are described in Figure 3-10 as Opnd 2 NL, D8-a, (where N = Normalized; L = Left aligned; D = Hex fraction digits; a = 9, 11 or 13 for precisions of 10, 12 or 14 respectively).

The A register contents, which were used in the RX format to set the operand 2 characteristic (Opnd 2C) to the exponent register B, are now reset.

The B register contains the high-order 32 bits of the normalized operand 2 fraction (Opnd 2N, D0-7).

For the multiplication processes, the B register becomes the operand 2 high-order multiplier field and the BX register the operand 2 low-order multiplier field; refer to "Principles of Long-Precision Multiply".

Sequence 2A

The operand 2 low-order multiplier field (Opnd 2 NL, D8-a) is set to the A register from the BX register via the scratch register.

R1 Cycle

Operand 1 low-order bits are read out of storage to the SDR. In parallel with this operation, the contents of the various registers in the ALS are manipulated in preparation for receiving this information.

The A, AX and B, BX registers are interchanged, and the B register and scratch register are then reset. The BX register is then set to the scratch register and the BX register is then reset. The SDR is gated to the B register and the scratch register is set to the A register.

The result is that the operand 2 normalized fraction is set in the A and AX registers while the BX register is free to receive the remaining section of operand 1 for pre-normalizing and left-adjusting.

In the exponent register area during this cycle, the operand 1 characteristic (Opnd 1C) has been gated from the operand 1 FPR defined by the Ra register to both exponent registers A and B. The exponent CLA output, gated for excess 64 arithmetic, is then set to exponent register B, thus forming the excess 64 sum of the two characteristics of the normalized fractions.

Sequence 3A

The operand 1 fraction is prepared for pre-normalizing by interchanging the low-order digits from the B register to the BX register. The complete fraction is formed by gating the fraction digits 0 to 5 (bits 8 to 31) from the FPR defined by the Ra register to the B register. Since these digits are aligned to the right of the B register, the contents are represented in Figure 3-10 by Opnd 1R, D0-5.

This sequence 3A is allowed to run only to CC 4 time.

If operand 1 is zero, both the zero HI and zero LO latches will be on at the completion of this cycle. Refer to "Setting of Zero HI and LO Latches".

Sequence 4A

The operand 1 double-word fraction is normalized in the B and BX registers. When this process is complete, the first shift-left-four operation is performed to left-align the fraction. Thus bits 00 to 03 of the B register are zero, followed by the normalized operand 1 fraction in the remainder of the B and BX registers.

The contents are represented in Figure 3-10 as Opnd 1 NR, D0-6 in the B register and Opnd 1NL, D7-a in the BX register. The second shift-left-four operation to completely left-align the operand 1 normalized fraction is made at the start of the next cycle (Step 2).

This sequence 4A is allowed to run only to CC 4 time.

Setting of Zero HI and LO Latches

During the operand 2 fetch (sequence 1A cycle) operation, both the zero HI and zero LO latches are set if operand 2 has a zero fraction. This action causes the second operand zero latch to be set. If this latch is on at the end of sequence 5A, an R1 cycle is called and zero is set to both halves of the operand 1 FPR; end execute is then signalled, terminating the multiply instruction.

During the operand 1 fetch (sequences 2A and 3A), the zero HI and zero LO latches are again set if the operand 1 fraction is zero. The 'on' state of these latches at this time does not cause the multiply instruction to be terminated but instead, causes both the pre-normalization process in sequence 4A and the testing for exception conditions in sequence 5D to be skipped. For this condition, the multiply process continues but a zero product is automatically developed.

Step 2 - Low-Order Product

- Cycles are from the start of sequence 1B to the end of sequence 2B cycle.
- Operand 2 low-order bits are multiplied by operand 1 low-order bits.
- The product low-order bits are reset.

Sequence 1B

A second shift-left-four is performed to left-align the operand 1 normalized fraction correctly. The B register now contains the digits 0 to 7 of the operand 1 fraction; these digits are the operand 1 high-order bits referred to in "Principles of Long-Precision Multiply." The BX register contains fraction digits 8 to 9, 8 to 11, or 8 to 13 for precisions of 10, 12 or 14 respectively; these digits are the operand 1 low-order bits referred to under the same heading.

The high-order 32 bits of the operand 1 fraction are temporarily stored from the B register to the high-order section of operand 1 FPR defined by the Ra register for use in the later sequence 4C (in step 4 of the instruction execution).

The B register is then reset, the BX register is set to the scratch register and the shift counter is set to values of 5, 3 or 1 for precision settings of 10, 12 or 14 respectively. These values will be used in the following sequence 3B to right-align the operand 1 low-order part in the BX register in preparation for the multiplication process.

In Figure 3-10, contents of exponent register B are shown at this time as P, C since they represent the characteristic of the product (result) prior to post-normalization.

Sequence 3B

The low-order digits of operand 1 are right-aligned in the BX register by a series of shift-right-four operations, one more than the value set into the shift counter on sequence 1B. This shifting positions the digits in readiness to act as a multiplier. The shift counter is then set for the first multiply sequence to values 9, 17 or 25 for precision of 10, 12 or 14 respectively. As with fixed-point multiplication, this setting represents one more than the number of multiplier digits.

Sequence 2B

This sequence is active for the complete low-order multiplication process.

As the A register contains the operand 2 low-order bits and the BX register contains the operand 1 low-order bits, the product of the bits will be the first step in the multiplication process. These cycles are described in "FP Common Multiply Cycles." After the low-order product is formed, the BX register is reset, truncating unwanted bits. The result in the B register is shown in Figure 3-10 as product 1, digits 0 to 7 (P1, D0-7).

Step 3 - First Cross-Product

- Cycles are from the start of sequence 4B to the end of sequence 5B cycles.
- Operand 2 high-order bits are multiplied by operand 1 low-order bits.
- The product is formed on the first cross-multiply high-order bits.
- The product low-order bits are stored for future use.

Sequence 4B

The B and BX registers are interchanged, placing product 1 digits 0 to 7 (P1, D0-7) in the BX register. The scratch register is then set to the B register and the shift counter is set to values of 5, 3 or 1 for precisions of 10, 12 or 14 respectively. These operations are preparatory to right-aligning the product and the multiplicand digits for the first cross-multiply operation.

Sequence 4B is allowed to run only to CC 4 cycle time.

Sequence 3B

Shift-right-four operations are performed on the B and BX registers until the digits are right-aligned in the two registers (shift counter = 0).

The A and AX registers are interchanged and the shift counter is set to 9, 17 or 25 for precisions of 10, 12 or 14 respectively. These operations prepare the fields for the first cross-multiply process.

The contents of the registers are identified as follows:

A register	:	Operand 2 high-order digits (Opnd 2 N, D0-7).
B register	:	Operand 1 low-order digits (Opnd 1 NR, D8-a).
BX register	:	Product 1 high-order digits (P1 R, D0-7).

Sequence 5B

This sequence is active for the complete multiplication process of developing the first cross-product.

Prior to the multiplication, the B and BX registers are interchanged, thus making operand 1 low-order digits the multiplier (BX register) and operand 2 high-order bits the multiplicand (A register). The product is formed on top of the product 1 high-order bits (B register), the process being described in "FP Common Multiply Cycles." The result of the multiplication is called product 2. The low-order bits of this product are temporarily stored by clearing the scratch register, setting the BX register to the scratch register then resetting the BX register. The second cross-product is later formed on these low-order bits.

Step 4 - Second Cross-Product

- Cycles are from the start of sequence 1C to the end of sequence 3D.
- Operand 1 high-order bits are multiplied by operand 2 low-order bits.
- The product is formed on the first cross-product low-order bits.
- The first and second cross-product low-order bits are added.
- The second cross-product low-order bits are reset.

A series of compute cycles are taken to re-align factors and set up the fields for the third and fourth multiply sequences.

For the second cross-product, the operand 2 low-order bits become the multiplier and these must be right-aligned in the BX register. As there are many fields which have to be retained and at any time only one spare ALS register exists, five sequences are required to prepare for the third multiplication.

Sequence 1C

The A, AX and B, BX registers are interchanged. The A register is then gated to the vacant B register.

Sequence 2C

The A register is reset and set with the contents of the scratch register. The scratch register is then reset.

Sequence 3C

The BX register is set to the scratch register and the BX register is then reset. The shift counter is set to 5, 3 or 1 for precisions of 10, 12 or 14 respectively, and the B and BX register interchanged.

Sequence 5C

The A register is gated to the B register.

Sequence 4C

Shift-right-four operations are performed on the B and BX registers until the shift counter is zero, right-aligning the digits in these registers in preparation for the second cross-multiply operation. The A register is reset during the shift process.

The shift counter is set to 9, 17 or 25 for precisions of 10, 12 or 14 respectively, and the contents of the FPR defined by the Ra register (stored in sequence 1B) are read out to the A register.

The digits and controls are now set up for the second cross-multiply operation. The contents of the relevant registers are as follows:

A register	:	Operand 1, high-order bits (Opnd 1 N, D0-7).
BX register	:	Operand 2, low-order bits (Opnd 2 NR, D8-a).
B register	:	Product 2, low-order bits (P2 R, D8-a).

Sequence 1D

This sequence is active for the complete multiplication process which develops the second cross-product. The product is formed with the product 2 low-order bits left in the B register after the previous cycle, as described in "FP Common Multiply Cycles."

Sequence 3D

The product 2 high-order bits are added to the product 3 high-order bits by setting the B register into the C register, resetting the A register and gating the scratch register to the A, B and C registers. The addition is completed by gating the CLA to the B register.

The shift counter is set to 33 in preparation for the fourth multiply sequence and the unwanted bits from the second cross-product are discarded by resetting the BX register.

Step 5 - High-Order Product

- Cycles are from the start of sequence 4D to the end of sequence 2D cycles.
- Operand 2 high-order bits are multiplied by operand 1 high-order bits.
- The product is formed on high-order bits of the two cross-multiply operations.

Sequence 4D

The A, AX and B, BX registers are interchanged and operand 1 high-order bits are once again read out of their temporary storage into FPR defined by the Ra register and set into the B register.

If a high-order carry from sequence 3D is present, it is latched up in the C0 multiply correction latch for addition on a later cycle.

This sequence 4D is allowed to run only to CC 4 time.

Sequence 2D

This sequence is active for the complete multiplication process which develops the high-order product, as described in "FP Common Multiply Cycles."

The B and BX registers are first interchanged, leaving the contents of the relevant registers as follows:

A register (multiplicand)	:	Operand 2, high-order bits (Opnd 2N, D0-7)
BX register (multiplier)	:	Operand 1, high-order bits (Opnd 1N, D0-7).
B register (initial factor)	:	Sum of cross-product high-order bits (P2+P3, D0-7).

The multiplication is initiated and is a full 32-bit multiply. The product formed is 64 bits or 16 hex digits. This product, after normalization, will be the result and it is represented in Figure 3-10 by P4.

As no guard digits are allowed for the normalizing process, the low-order byte (two digits) of the BX register is reset.

Step 6 - Post-Normalization, Exception Test and Result Store

- Cycles are from the start of sequence 5D to the end of R1 cycle.

- Post-normalization is performed when necessary.

- For exponent underflow conditions, the B register is set to zeros.

- Exponent overflow or underflow exceptions are tested except for the 'operand 1 equals zero' condition.

- The result characteristic and fraction are assembled and stored in long-precision format in operand 1 FPR.

Sequence 5D

As the final multiplication was performed with 32 bits of normalized fraction, the number of zeros in the high-order position of the fraction will always be less than eight. This means that only one normalization cycle is required. As it is also necessary to align the high-order bit of the fraction in bits 08 to 11 of the B register (to leave room for the result characteristic) the two processes are combined.

The B and BX registers are given a shift-right-four operation and a second shift-right-four operation is made only if B register bits 00 to 07 still contain a significant digit.

If the zero HI and zero LO latches are on (operand 1 equals zero), no further action is taken; if neither or only one of them is on, the exponent is tested for overflow or underflow conditions. If there are no exception conditions, the result exponent is gated from exponent register B to the B register. For exponent overflow conditions, the result exponent is also gated to the B register. For an exponent underflow, the result exponent is not set to the B register but the B register is reset to zeros.

R1 Cycle

The B register is set to the operand 1 FPR high-order section, the B and BX registers are interchanged, and the B register is set to the SDR and stored in the low-order section of operand 1 FPR.

For exponent underflow conditions, the B to BX register interchange is not taken and the zeros in the B register are stored in both halves of the FPR defined by the Ra register.

For exponent overflow or underflow conditions with a mask bit of one, the appropriate FP exception latch is set and a program interrupt is requested.

EXECUTION OF LONG-PRECISION MULTIPLY (PRECISION = 8)

For operands with a precision of 8, the length of the fraction is eight hex digits (32 binary bits). This means that fraction multiplication can be done in one 32-bit multiply step and is, in effect, a high-order multiply process only. Thus, many cycles are saved in comparison with the long-precision multiply process.

The instruction sequence and timings for long-precision multiply (shown in FEMD Figures 6347 to 6368) apply to a precision of 8; the skipped sequences are shown in these Figures. A single-cycle chart for long-precision multiply (precision = 8) is given in Figure 3-11.

Step 1 - Operand Fetch and Pre-Normalization

- Cycles are from the end of I-fetch to the end of sequence 4A.

- Operand 1 fraction, normalized and left-aligned to bit 00, is set in the A and AX registers.

- Operand 2 fraction, normalized and left-aligned to bit 04, is set in the B and BX registers.

The operations for the operand fetch and pre-normalization cycles differ from those for precisions of 10, 12 or 14 in that the low-order bits are truncated to hold only digits 6 and 7 in the BX register. Therefore, after a full left-alignment of the 8-digit fraction, the whole fraction can be contained in the full-word A or B register. Thus, at the end of these aligning cycles, the AX register is blank and the BX register contains only operand 1 digit 7; this digit is shifted into the B register at the start of the next cycle, leaving the BX register also blank.

Cycle	A Reg	AX Reg	B Reg	BX Reg	Scratch Reg	Exp Reg A	Exp Reg B	Shift Counter
RR Format only								
Seq. 1A	Opnd 2C, D0-5	-	-	Opnd 2C, D0-5	-	-	Opnd 2, C	-
R2 Cycle	Opnd 2C, D0-5	-	Opnd 2C, D0-5	Opnd 2, D6-7	-	-	Opnd 2, C	-
RX Format only								
EA, EA, 1A	Opnd 2C, D0-5	-	Opnd 2C, D0-5	Opnd 2, D6-7	-	-	Opnd 2, C	-
5A	--	-	Opnd 2N, D0-7	-	-	-	Opnd 2, CN	-
R1	Opnd 2N, D0-7	-	Opnd 1, D6-7	-	Opnd 2N, D0-7	Opnd 1, C	Opnd 1, C+2, CN	-
3A *	Opnd 2N, D0-7	-	Opnd 1, D0-5	Opnd 1, D6-7	-	Opnd 1, C	Opnd 1, C+2, CN	-
4A *	Opnd 2N, D0-7	-	Opnd 1NR, D0-6	Opnd 1NL, D7	-	Opnd 1, C	Opnd 1+2, CN	-
1B *	Opnd 2N, D0-7	-	Opnd 1N, D0-7	-	-	Opnd 1, C	P, C	33
End 2B's	Opnd 2N, D0-7	-	P, D0-7	P, D8-13	-	Opnd 1, C	P, C	-
5B	Opnd 2N, D0-7	-	PN, CN, D0-5	PN, D6-13	-	Opnd 1, C	PN, C	-
R1	Opnd 2N, D0-7	-	PN, D6-13	PN, CN, D0-5	-	Opnd 1, C	PN, C	-

* Cycle CC 1 to CC 4 only

Register Contents Legend:

Opnd 1	:	Operand 1	D0-13	:	Hex Fraction Digits 0 to 13
Opnd 2	:	Operand 2	C	:	Characteristic and Sign
F	:	Fraction	L	:	Digits Left Aligned in Register
N	:	Normalized	R	:	Digits Right Aligned in Register
P	:	Product			

Product Derivation:

P + (Opnd 2N, D0-7) x (Opnd 1N, D0-7)

Figure 3-11. Single-Cycle Chart for Long-Precision Multiply (Precision = 8)

Step 2 - Multiplication

- Cycles are from the start of sequence 1B to the end of sequence 2B.
- Operand 2 high-order bits are multiplied by operand 1 high-order bits.

Sequence 1B

The left alignment is completed with a shift-left-four operation and the BX register is then reset. The shift counter is set to 33 in preparation for the multiplication process.

Sequence 2B

This sequence controls the complete multiplication process. For details of the multiply cycles, refer to "FP Common Multiply Cycles".

The B and BX registers are interchanged, the 32-bit multiplication process is started and the 64-bit product is formed. The low-order byte (two digits) of the BX register is reset, leaving a product of 14 digits.

Step 3 - Post-Normalization, Exception Test and Product Store

- Cycles are from the start of sequence 5B to the end of R1 cycle.
- Post-normalization is performed when necessary.
- Exponent underflow conditions cause the B register to be set to zeros.
- Exponent overflow or underflow exceptions are tested except for the 'operand 1 equals zero' condition.
- The result characteristic and fraction are assembled and stored in long-precision format in operand 1 FPR.

The operation differs from the long-precision multiplication (precisions of 10, 12 or 14) cases only in the controlling sequence for the post-normalization and exception test cycles. For a precision of 8, the sequence is sequence 5B. Refer to "Sequence 5D" and "R1 Cycle" in Step 6 of the previous section for details of the remaining cycles.

FP COMMON MULTIPLY CYCLES

- The principles of the FP multiply loop are similar to those for fixed-point multiply.
- The number of multiply cycles is determined by the initial setting of the shift counter.
- The multiply cycles continue until the shift counter equals zero.
- The B register extension sign bit is used as a sign bit for the partial product field (B, BX registers).
- The state of the subtract trigger is used as a sign bit for the multiplicand field (A register).

The FP multiply loop uses the same principles of operation as fixed-point multiply (refer to Principles of Operation - Processing Unit, Form Y33-0002).

Summarized, these principles are as follows:

1. The multiplier is located in the BX register; the multiplier digit in bit position 31.
2. The multiplicand is located in the A register.
3. The product is developed in the B register and a shift-right-one operation is performed at each multiply cycle, forming the complete result product in the B and BX registers.
4. The "carry-look-ahead group-of-ones" principle is used, as with fixed-point multiply.
5. The types of cycle for a multiplier digit of 1 are:
   a. A one, part of a group of zeros : Add and shift.
   b. The first one of a group of ones : Subtract and shift.
   c. A one, part of a group of ones but not the first : Shift only.
6. The types of cycle for a multiplier digit of 0 are:
   a. A zero, part of a group of ones : Subtract and shift.
   b. A zero, the first after a group of ones : Add and shift.
   c. A zero, part of a group of zeros but not the first of a group of ones : Shift only.
7. The subtract trigger is turned on when the first one of a group of ones becomes the multiplier bit (BX register bits 30, 31 = 1, 1). The trigger turned off when the first zero after a group of ones becomes the multiplier bit (BX bits 30, 31 = 0, 0).
8. At the start of each cycle, the subtract trigger is copied into the subtract trigger backup. Thus the subtract trigger backup is on for the second one of a group of ones and remains on until the first zero after the group of ones.
9. An arithmetic overflow on the arithmetic type of cycle is corrected during the shift-right-one that follows.

As the multiplicand field can contain up to 32 bits of data, an extension sign bit position must be used on the A and B registers to determine the sign of partial result fields participating in the arithmetic cycle. Since the A register and the subtract trigger are always inverted in parallel, the state of the subtract trigger is used to indicate the A register sign.

The basic CPU does not have a trigger or latch which can perform a similar function for the B register. The special floating-point 'B register extension sign' bit is used for this purpose. Note that this bit does not take part in the normal arithmetic process and that the carry-out (C0 latch) of the B register can be regarded as the carry-in to this extension sign bit position.

At the start of each multiply cycle except the first, the B register is given a shift-right-one operation. The 'B00 shift-right-one insert' into the B register bit 00 position is the data bit that would normally be developed as a result bit (in the extension sign bit position) if it took part in the arithmetic process. The bit inserted into the B register extension sign (on the same shift-right operation) is the sign bit that would be developed in a position one to the left of the extension sign position (overflow position) if two bits, equivalent to the extension sign bits, were to be assumed for this position. (Bits equal to the sign bit can be assumed at the left of System/360 signed data without altering the data value or sign.)

The two bits that are inserted into the B register bit 00 and the B register extension sign, as described previously, are developed separately and are temporarily stored in the FP mult/div bit store latch (refer to the instructional diagram, Figure 3-12). From this latch, each bit is shifted or set into the appropriate bit position.

The data bit developed in the extension sign bit position which gives the 'B00 shift-right-one insert' is a one if either one or all three of the following is a one: B register extension sign bit, subtract trigger (A register extension sign bit) and C0 latch. The data bit condition is given by the EXOR of these three conditions as represented by the ODD blocks.

The B register extension sign bit carry-out is a one if two or more of these same three conditions is a one. This carry-out provides a carry-in to the overflow position. As described previously, the overflow position can be assumed to contain bits equal to the bits in the extension sign bit positions. The result in the overflow position will be one if either one or all three of the following is a one: B register extension sign bit carry-out, B register

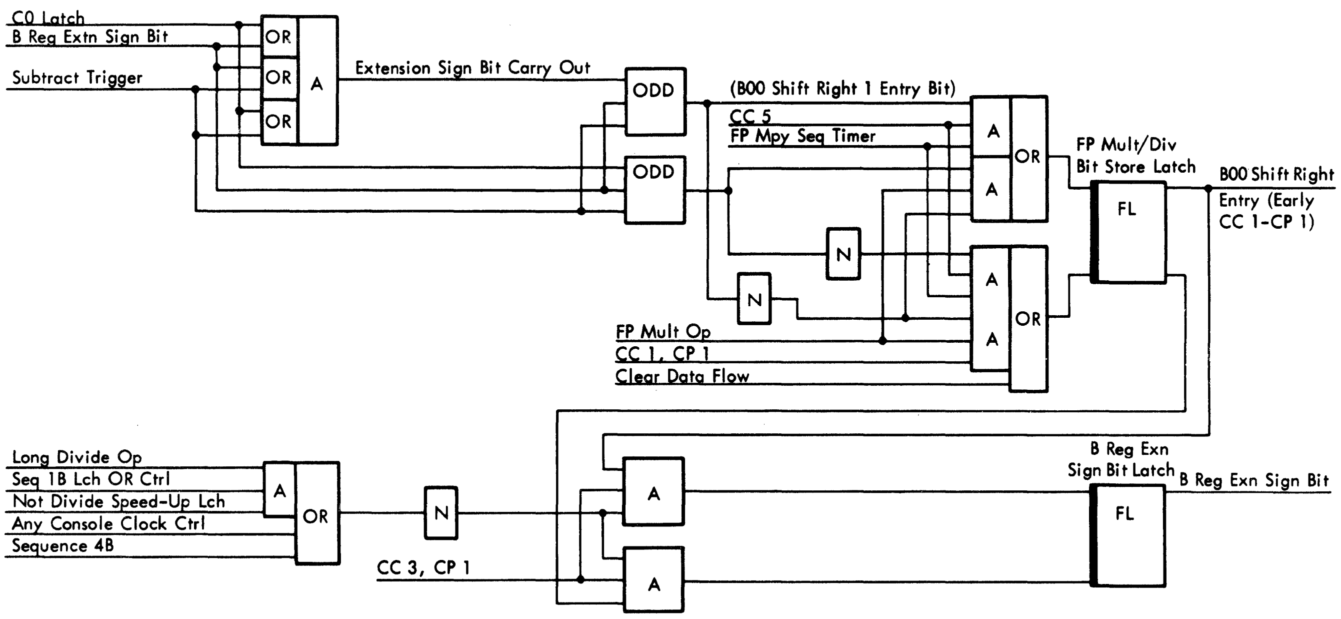


Figure 3-12. Instructional Diagram of B Register Extension Sign Bit for Multiply Cycles

extension sign bit and subtract trigger (A register extension sign bit). The sign bit condition is given by the EXOR of these three conditions as shown in Figure 3-12.

The logic and timing of these decisions, together with the normal multiply logic, is shown in FEMD Figures 6395 and 6396. The initial shift counter values for each type of operation is shown in Figure 3-13.

On the last cycle, when the shift counter equals zero, a sign bit of zero can be assumed. As with fixed-point multiply operations, the zero sign bit causes an arithmetic cycle only if the subtract trigger backup is on; this cycle will always be an add cycle. If the multiply sequence is the high-order

Instruction Precision	Variable-Precision Switch Value	Controlling Sequence	Initial Shift Counter Value
Short	-	2A	25
Long	8	2B	33
Long	10	2B	9
	10	5B	9
	10	1D	9
	10	2D	33
Long	12	2B	17
	12	5B	17
	12	1D	17
	12	2D	33
Long	14	2B	25
	14	5B	25
	14	1D	25
	14	2D	33

Figure 3-13. Initial Shift Counter Values for Common Multiply Loop

multiply (sequence 2D) of a long-precision multiply instruction and the C0 multiply correction latch is on, the correction of one is made in the last cycle by forcing a carry-in (subtract trigger on) for the add cycle.

On the last cycle (shift counter equalling zero) of all but the high-order multiply (sequence 2D) of a long-multiply instruction, the A register is restored to its true condition if it is inverted. That is, the A register is re-inverted if the subtract trigger is on at the completion of this cycle.

DIVIDE INSTRUCTIONS (SHORT-PRECISION OPERANDS)

- The dividend (operand 1) is divided by the divisor (operand 2).
- The quotient replaces the dividend in the operand 1 field.
- A remainder is not preserved.
- The dividend, divisor and quotient are all short-precision fields.
- The low-order halves of the FPR's are ignored and remain unchanged.
- The division process consists of a fraction division and a characteristic subtraction.
- The sign of the quotient is determined by the rules of arithmetic.

- An exponent overflow exception occurs if the final quotient characteristic exceeds 127.
- An exponent underflow exception occurs if the final quotient characteristic is less than zero and the corresponding mask bit is one.
- For exponent underflow conditions, the characteristic sign and fraction of the quotient are made zero.
- When the divisor fraction is zero, the operation is suppressed and an FP divide exception is taken.
- When the dividend fraction is zero, the quotient fraction will be zero and the quotient sign and characteristic will be made zero.
- When the dividend fraction is zero, exceptions due to exponent underflow or overflow are not taken.
- The significance exception is never set for FP divide operations.

Instruction Execution

- The divisor (operand 2) is pre-normalized and set into the A register.
- The dividend (operand 1) is pre-normalized and set into the B register.
- The quotient fraction developed is formed in the scratch register.
- The quotient is normalized during the division process.
- Six quotient digits are developed.

The floating-point divide operation is executed in the A and B registers and the quotient is formed in the scratch register.

The divisor (operand 2) is pre-normalized in the B register, then shifted-left one digit for initial alignment. If the divisor fraction is zero, an FP divide exception is signalled and a program interrupt requested. For normal conditions, the fraction is set into the A register.

The dividend (operand 1) is then pre-normalized in preparation for the first divide cycle. If the dividend is zero, the divide operation proceeds (as for a normal non-zero dividend) and a quotient fraction of zero results. The quotient fraction is stored but the quotient sign and characteristic are reset to give a true zero result.

The quotient is developed in the scratch register. The rules of this divide operation are outlined in "FP Common Divide Cycles" in this chapter. The quotient fraction is always developed in a normalized form, so that post-normalization cycles are not necessary. The quotient is formed from the quotient characteristic and sign (exponent register B) and the quotient fraction (scratch register) and is stored in the high-order half of the original operand 1 FPR.

For details of timing and sequence, refer to FEMD Figures 6369 to 6374, and 6398.

Operand 2 Fetch and Pre-Normalizing Cycles

- For RR format, the cycles involved are from the end of I-fetch to the end of sequence 5A (I-cycle, sequence 1A, sequence 5A).
- For RX format, the cycles involved are from the end of I-fetch to the end of sequence 5A (EA cycle and sequence 1A, sequence 5A).
- Operand 2 fraction digits are fetched, pre-normalized, shifted-left one digit and placed in the BX register.
- Operand 2 characteristic is placed in exponent register B.
- Operand 1 characteristic is EXOR'ed into exponent register B.
- If operand 2 fraction is zero, the FP divide exception latch is turned on and end execute is signalled.
- The shift counter is set to 24.

Operand 2 is first read out and set to the B register. The operand 2 characteristic is set to exponent register B and the shift counter is set to 24.

On sequence 5A, the operand 2 fraction (in the B register) is normalized by repeated shift-left-four operations until the bits 8 to 11 become significant. One more shift-left-four operation is performed to complete the initial alignment of the operand 2 fraction; it is then placed in the BX register by a B to BX register interchange.

The operand 1 characteristic is set to exponent register B, thus forming the partial difference of the operand 1 and operand 2 characteristics in that register.

If the operand 2 fraction is zero, the zero HI and FP divide exception latches are set. This action causes normalization cycles on sequence 5A to be suspended and end execute to be signalled at CC 4 of that cycle. A program interrupt request is generated.

Operand 1 Fetch and Pre-Normalizing Cycles

- The cycles involved are sequence 2A and 4A.
- The operand 2 aligned fraction is placed in the A register via the scratch register.
- The operand 1 fraction is normalized and remains in the B register.
- The excess 64 difference of the two characteristics is formed in exponent register B.

On sequence 2A, the operand 1 fraction is read out from the FPR defined by the Ra register and set to the B register. The BX register is cleared by transferring the operand 2 aligned fraction to the scratch register and resetting the BX register.

If the operand 1 fraction is zero, the zero LO latch is set for later use. For this condition, the divide cycles are not skipped and a zero result is generated by the division.

The difference between the operand 1 and operand 2 characteristics is formed in exponent register B by gating the exponent CLA to that register. The scratch register is then gated to the A register via the HW funnel and the ABC funnel.

The operand 1 fraction is normalized on sequence 4A and, at the completion of this sequence, the operands are aligned in the registers in preparation for the divide cycles.

Divide Cycles

- Division is performed on sequence 1B cycles.
- The quotient is formed in the scratch register.
- The quotient is developed in normalized form.

The division cycles are a series of sequence 1B compute clock cycles (for details, refer to "FP Common Divide Cycles"). At the end of these cycles

Cycle	A Reg	AX Reg	B Reg	BX Reg	Scratch Reg	Exp Reg A	Exp Reg B	Shift Counter
RX Format only								
EA, EA, 1A	Opnd 2C, D0-5	-	Opnd 2C, D0-5	Opnd 2, D6-a	-	-	Opnd 2, C	24
RR Format only								
1A	Opnd 2, D0-5	-	Opnd 2, D0-5	-	-	-	Opnd 2, C	24
R2	Opnd 2, D0-5	-	Opnd 2, D0-5	Opnd 2, D6-a	-	-	Opnd 2, C	24
5A	-	-	Opnd 2N, D0-7	Opnd 2N, D8-a	-	-	Opnd 2, C	24
2A	Opnd 2N, D8-a	-	Opnd 2N, D8-a	Opnd 2N, D0-7	Opnd 2N, D8-a	-	Opnd 2, C	24
R1	Opnd 2N, D0-7	Opnd 2N, D8-a	Opnd 1, D6-a	-	Opnd 2N, D0-7	Opnd 1, C	Opnd 1C-2C	24
3A	Opnd 2N, D0-7	Opnd 2N, D8-a	Opnd 1, D0-5	Opnd 1, D6-a	Opnd 2N, D0-7	Opnd 1, C	Opnd 1C-2C	24
4A	Opnd 2N, D0-7	Opnd 2N, D8-a	Opnd 1NR, D0-6	Opnd 1N, D7-a	Opnd 2N, D0-7	Opnd 1, C	Opnd 1C-2C	24
**	Opnd 2N, D0-7	Opnd 2N, D8-a	Partial Result		QN, D0-5	Opnd 1, C	Opnd 1C-2C	0
R1	Opnd 2N, D0-7	Opnd 2N, D8-a	-	Partial Result	QN, D0-5	Opnd 1, C	Opnd 1C-2C	b
4B	Opnd 2N, D0-7	Opnd 2N, D8-a	-	Partial Result	-	Opnd 1, C	QC	b
* R1	Opnd 2N, D0-7	Opnd 2N, D8-a	Partial Result		-	Opnd 1, C	QC	0
**	Opnd 2N, D0-7	Opnd 2N, D8-a	Remainder	-	QR, D6-a	Opnd 1, C	QC	0
1C	Opnd 2N, D0-7	Opnd 2N, D8-a	QR, D6-a	-	QR, D6-a	Opnd 1, C	QC	0
2C (N-10)	Opnd 2N, D0-7	Opnd 2N, D8-a	Partial Left Shift	-	QR, D6-a	Opnd 1, C	QC	0
R1	Opnd 2N, D0-7	Opnd 2N, D8-a	QL, D6-a	-	QR, D6-a	Opnd 1, C	QC	0

* This R1 Cycle is shown as a single-cycle stop for convenience. In practice the first Sequence 1B of the divide cycle will be performed in conjuction with this cycle.

** : End of Sequence 1B Cycles or Sequence 1B, 2B, 3B, Cycles

Registers Contents Legend:

Opnd 1	:	Operand 1	D0-13	:	Hex Fraction Digits 0 to 13
Opnd 2	:	Operand 2	C	:	Characteristic and Sign
Q	:	Quotient	L	:	Digits Left Aligned in Register
N	:	Normalized	R	:	Digits Right Aligned in Register

Constants Values:

	Precision		
	10	12	14
Constant a	9	11	13
Constant b	16	24	32

Figure 3-14. Single-Cycle Chart for Long-Precision Divide (Precision = 10, 12, 14)

the normalized quotient is contained in the scratch register and the remainder in the B register. Exponent register B contains the quotient characteristic and sign.

Quotient Store Cycle

- The cycle involved is a sequence 2B cycle.
- The quotient is assembled and stored in the original operand 1 FPR.
- For zero LO or exponent underflow conditions, the quotient is set to zero.
- The remainder is not stored.
- Exponent underflow and overflow exceptions are tested.

The remainder in the B register is reset and the final shift-left-one operation of the scratch register is performed. The quotient result is then assembled. If zero LO or an exponent underflow has occurred, then the result must be true zero and the reset B register represents the quotient to be stored.

For all other conditions, the quotient is formed by gating exponent register B (bits 0 to 7) and the scratch register (bits 8 to 31) to the corresponding bits of the B register. The B register is then gated to the FPR defined by the Ra register, thus storing the quotient in the original operand 1 FPR.

INTRODUCTION TO DIVIDE INSTRUCTIONS (LONG-PRECISION OPERANDS)

- The dividend (operand 1) is divided by the divisor (operand 2).
- The quotient replaces the dividend in the operand 1 field.
- A remainder is not preserved.
- The dividend, divisor and quotient are all long-precision fields.
- The divisor process consists of a fraction division and a characteristic subtraction.
- The sign of the quotient is determined by the rules of arithmetic.
- An exponent overflow exception occurs if the final quotient characteristic exceeds 127.
- An exponent underflow exception occurs if the final quotient characteristic is less than zero and the corresponding mask bit is one.
- For exponent underflow conditions, the characteristic sign and fraction of the quotient are made zero.
- When the divisor fraction is zero, the operation is suppressed and an FP divide exception is taken.
- When the dividend fraction is zero, the quotient fraction will be zero, and the quotient sign and characteristic will be made zero.
- When the dividend fraction is zero, exceptions due to exponent underflow or overflow are not taken.
- The significance exception is never set for FP divide operations.

EXECUTION OF LONG-PRECISION DIVIDE (PRECISION = 10, 12, 14)

For details of the sequence and timing of the cycles described in the following steps, refer to FEMD Figures 6375 to 6392. For a summary of the register contents at each major single-cycle point of the divide operation, refer to Figure 3-14.

Step 1 - Operand 2 Fetch and Pre-Normalizing

- Cycles are 1A, R2, 5A and 2A (RR format) or EAEA 1A, 5A and 2A (RX format).
- Operand 2 is set to the B and BX registers and normalized.
- Operand 2 characteristic is set to exponent register B.
- The shift counter is set to 24.
- Operand 2 is tested for a zero divisor.

EAEA Cycle with Sequence 1A (RX Format)

On these cycles, the operand 2 is read out and set to the B and BX registers, the operand 2 high-order bits are set to the A register, the operand 2 characteristic is set to exponent register B (from the A register) and the shift counter is set to 24. This action prepares the operand 2 fraction for the pre-normalization process.

Sequence 1A, R2 Cycle (RR Format)

On these cycles the operand 2 is read out and set to the B and BX registers in preparation for normalization, the operand 2 characteristic is gated to exponent register B and the shift counter is set to 24.

NOTE: For both the above sets of cycles, the operand 2 zero latch and the FP divide exception latch are set if the operand 2 fraction is zero.

Sequence 5A

On this sequence, the operand 2 is pre-normalized until bits 08 to 11 of the B register are significant. The normalized fraction is then given two shift-left-four commands to align the fraction into bit 00 of the B register; digits 0 to 7 of the normalized fraction are left in the B register and the remaining digits in the BX register. The A register is reset.

If the divide exception latch was set in the previous cycle, end execute is signalled at CC 4 of this cycle and the divide instruction is terminated.

Sequence 2A

The low-order section of the normalized operand 2 fraction is set into the A register by gating it via the scratch register. The B and BX registers are interchanged and the low-order bits, now in the B register, are tested to determine if they are zero; if zero, the divide speed-up latch is set.

This latch indicates that the divisor is 32 bits or less and the division arithmetic cycles can be single cycles similar to those of the short-precision divide instruction (see "FP Common Divide Cycles"). If the divide speed-up latch is not on, the divisor contains more than 32 bits and two separate arithmetic cycles are required for each addition and subtraction.

Step 2 - Operand 1 Fetch and Pre-Normalizing

- Cycles are R1, 3A and 4A.
- Operand 2 fraction (normalized and left-aligned) is placed in the A and AX registers.
- Operand 1 fraction is fetched to the B and BX registers, normalized and shifted-left-four.
- Operand 1 characteristic is gated to exponent registers A and B.
- The difference (in excess 64 arithmetic) of the two characteristics is formed in exponent register B.
- Operand 1 (dividend) is tested for zero.

R1 Cycle

The operand 1 low-order bits are read out to the SDR. The B register and the scratch register are reset, and the A, AX registers interchanged.

The BX register is gated to the scratch register and reset. The SDR is gated to the B register via the ABC funnel, truncating unwanted bits as defined by the setting of the variable-precision switch. The scratch register is gated to the A register via the ABC funnel. Therefore, the operand 2 fraction, normalized and left-aligned, is contained in the A and AX registers.

In parallel with these operations, the operand 1 characteristic is gated to exponent registers A and B and the exponent CLA output is then gated to exponent register B to form the excess 64 difference of the two characteristics. If the B register is zero, the zero LO latch is set.

Sequence 3A

The B and BX registers are interchanged and operand 1 fraction digits 0 to 5 are gated from the FPR to the B register. This means that operand 1 is now in the B and BX registers in preparation for normalization.

If the B register is zero, the zero HI latch is set. Therefore, a zero dividend is signalled by both zero HI and zero LO being on. For this condition, the divide proceeds, but a zero result will be formed on a later cycle.

Sequence 4A

The operand 1 fraction in the B and BX registers is normalized, then shifted-left one digit position to ensure that the divisor is always numerically larger than the dividend for the divide cycles.

Step 3 - Divide Cycles and High-Order Quotient Storing

- Cycles are from the start of sequence 1B to the end of the second set of divide cycles.
- The high-order quotient is developed on the first set of divide cycles and stored in FPR (Ra) high-order bits.
- The low-order quotient is developed on the second set of divide cycles and stored in FPR (Ra) low-order bits.
- The divide cycles are sequence 1B cycles for the divide speed-up 'on' condition.
- The divide cycles are sets of sequence 1B, 2B and 3B cycles for the divide speed-up 'off' condition.

Sequence 1B or Sequences 1B, 2B, 3B

These cycles are the division cycles to develop the high-order six digits of the quotient. They are either single or double arithmetic cycles, dependent on the condition of the divide speed-up latch (refer to "FP Common Divide Cycles").

R1 Cycle

On this cycle, the partial result in the B register is temporarily stored in the low-order section of the operand 1 FPR to leave the B register free for sequence 4B.

Sequence 4B

The LO order quotient latch is set and the shift counter set for the low-order divide cycles. The shift counter values are 16, 24 or 32 for precisions of 10, 12 or 14 respectively.

The high-order quotient is assembled in the B register by gating the scratch register bits 08 to 31 to the corresponding positions of the B register and gating exponent register B to bits 00 to 07 of the B register. The B register is then stored in the high-order half of the operand 1 FPR and the scratch register is reset.

For a zero dividend (zero HI and zero LO latches on) or exponent underflow condition, the scratch register is not gated to the B register and the zeros from the B register are stored as the quotient result.

R1 Cycle

The partial result stored on the preceding R1 cycle is read out to the B register, thus restoring the partial result remaining after the first set of divide cycles.

Sequence 1B or Sequences 1B, 2B, 3B

These cycles are the division cycles to develop the low-order digits of the quotient and are similar to the cycles used for the high-order digits (refer to "FP Common Divide Cycles" in this chapter).

Step 4 - Low-Order Quotient Storing and Testing for Exceptions

- Cycles are sequence 1C, 2C (N = 10 only) and R1.
- The low-order quotient is aligned to the left-hand end of the B register and stored in the low-order operand 1 FPR.
- Exponent underflow condition is tested.

Sequence 1C

The final shift-left-one of the scratch register is performed to shift in the last quotient digit. The B and BX registers are interchanged and the BX register is reset. The low-order quotient in the scratch register is then gated to the B register.

The low-order quotient has to be aligned with bit 00 position prior to storing. This means that on subsequent cycles it must be shifted-left two or four digits for precisions of 12 or 10 respectively.

Sequence 2C (N = 10 only)

Two shift-left-four operations on this cycle align the quotient for the case where precision equals 10.

R1 Cycle

For precisions equal to 10 or 12, two shift-left-four operations are performed, completing the left-alignment of the quotient. The low-order quotient in the B register is then gated to the SDR and stored in the low-order half of the operand 1 FPR.

The exception conditions are now tested. If exponent overflow conditions are present, the exponent overflow exception latch is turned on and a program interrupt is requested. If exponent underflow conditions are present, the exponent underflow exception latch is turned on if the mask bit (PSW 2, bit 6) is a one; a program interrupt is similarly requested.

EXECUTION OF LONG-PRECISION DIVIDE (PRECISION = 8)

For details of the sequence and timing of the cycles described in the following steps, refer to FEMD Figures 6375 to 6384, 6387 and 6388, 6393 and 6394. For a summary of the register contents at each major single-cycle point of the divide operation, refer to Figure 3-15.

Step 1 - Operand 2 Fetch and Pre-Normalizing

- Cycles are 1A, R2, and 5A (RR format) or EAEA 1A and 5A (RX format).
- Operand 2 is set to the B and BX registers and normalized.
- Operand 2 characteristic is set to exponent register B.
- The shift counter is set to 32.
- Operand 2 is tested for a zero divisor.

Cycle	A Reg	AX Reg	B Reg	BX Reg	Scratch Reg	Exp Reg A	Exp Reg B	Shift Counter
RX Format only								
EA,EA,1A	Opnd 2C,D0-6	-	Opnd 2C,D0-5	Opnd 2,D6-7	-	-	Opnd 2,C	32
RR Format only								
1A	Opnd 2,D0-6	-	Opnd 2,D0-5	-	-	-	Opnd 2,C	32
R2	Opnd 2,D0-8	-	Opnd 2,D0-5	Opnd 2,D6-7	-	-	Opnd 2,C	32
5A	-	-	Opnd 2N,D0-7	-	-	-	Opnd 2,C	32
R1	Opnd 2N,D0-7	-	Opnd 1,D6-7	-	Opnd 2N,D0-7	Opnd 1,C	Opnd 1C-2C	32
3A	Opnd 2N,D0-7	-	Opnd 1,D0-5	Opnd 1,D6-7	Opnd 2N,D0-7	Opnd 1,C	Opnd 1C-2C	32
4A	Opnd 2N,D0-7	-	Opnd 1N,D0-6	Opnd 1NL,D7	Opnd 2N,D0-7	Opnd 1,C	Opnd 1C-2C	32
End 1B's	Opnd 2N,D0-7	-	Remainder	-	QR,D0-7	Opnd 1,C	QC	0
1C	Opnd 2N,D0-7	-	QR,D0-7	-	QR,D0-7	Opnd 1,C	QC	0
3C	Opnd 2N,D0-7	-	QC,D0-5	QL,D6-7	QR,D0-7	Opnd 1,C	QC	0
R1	Opnd 2N,D0-7	-	QL,D6-7	QC,D0-5	QR,D0-7	Opnd 1,C	QC	0

Register Contents Legend:

Opnd 1	: Operand 1	D0-13	: Hex Fraction Digits 0 to 13
Opnd 2	: Operand 2	C	: Characteristic and Sign
Q	: Quotient	L	: Digits Left Aligned in Register
N	: Normalized	R	: Digits Right Aligned in Register

Figure 3-15. Single-Cycle Chart for Long-Precision Divide (Precision = 8)

The difference between these cycles and those for precisions of 10, 12 or 14 is that the shift counter is now set to 32, and that sequence 2A is not taken. The divide speed-up latch is automatically on when precision equals 8.

Step 2 - Operand 1 Fetch and Pre-Normalizing

- Cycles are R1, 3A and 4A.
- The operand 2 fraction (normalized and left-aligned) is placed in the A and AX registers.
- The operand 1 fraction is fetched to the B and BX registers, normalized and shifted-left one digit position.
- The operand 1 characteristic is gated to exponent registers A and B.
- Operand 1 is tested for a zero dividend.

These cycles do not differ from the operand 1 fetch and pre-normalize cycles for precisions of 10, 12 or 14.

Step 3 - Divide Cycles

- Cycles are sequence 1B division cycles (divide speed-up latch on).
- The eight-digit quotient is developed in the scratch register.

The division cycles use single arithmetic cycles since the divide speed-up latch is on. For details, refer to "FP Common Divide Cycles" in this chapter.

Step 4 - Quotient Storing and Testing for Exceptions

- Cycles are 1C, 3C and R1.
- The quotient is assembled and stored in the operand 1 FPR.
- For a zero result or exponent underflow conditions, the quotient is set to zeros.
- For exponent underflow or zero dividend, a zero quotient is stored.
- Exponent underflow and overflow exceptions are tested.

Sequence 1C

The final shift-left of the scratch register is performed to shift into this register the last quotient digit. The B and BX registers are interchanged and the BX register reset; this leaves the B register also in a reset condition.

If there has not been either an exponent underflow or a zero dividend (zero HI and zero LO latches on), the eight-digit quotient in the scratch register is gated to the B register.

Sequence 3C

The quotient is right-shifted two digit positions and, if there has not been either an exponent underflow or a zero dividend, exponent register B is gated to the B register bits 00 to 07.

This sequence completes the assembly of the quotient result in the B and BX registers. Note that for exponent underflow or zero divisor conditions this result has been made zero.

R1 Cycle

The high-order quotient bits in the B register are stored in the high-order half of the operand 1 FPR. The B and BX registers are interchanged and the low-order quotient bits are set from the B register to the SDR and stored into the low-order half of the operand 1 FPR.

If exponent overflow conditions are present, the exponent overflow exception latch is set and a program interrupt requested. If exponent underflow conditions are present and the mask bit (PSW 2, bit 6) is a one, the exponent underflow exception latch is set and a program interrupt is requested.

FP COMMON DIVIDE CYCLES

- Single arithmetic type divide cycles are used if the divisor field is 32 bits or less.
- Double arithmetic type divide cycles are used if the divisor field is greater than 32 bits.
- The initial shift counter value depends on the preset precision and the state of the LO order quotient latch.
- The value of the shift counter is decreased on each divide loop.
- Divide loops continue until the shift counter equals zero.

The FP divide cycles loop is entered on each occasion that it is required to develop a quotient during the execution of an FP divide instruction.

The long-precision divide instruction, when precision equals 10, 12 or 14, twice enters a divide loop on each instruction execution. This is because the quotient of these long-precision divide instructions is developed in two sections: the high-order quotient (LO order quotient latch off) and the low-order quotient (LO order quotient latch on).

The type of arithmetic cycles used during the divide loop depends on the length of the divisor.

If the divisor field is 32 bits or less, the divisor is contained in the A register alone so that the addition or subtraction of the divisor to the dividend field can be achieved in one cycle. The common divide loop is called the FP common divide - single arithmetic cyle. This type of cycle occurs on either short-precision divide, long-precision divide (when precision equals eight), or long-precision divide (when precision equals 10, 12 or 14 and the divide speed-up latch is on); for the precision equal to 10, 12 or 14 case, the single arithmetic type of cycle is used for developing both the high-order and the low-order sections of the quotient. The initial values of the shift counter for each of these conditions is shown in Figure 3-16.

Instruction Precision	Variable-Precision Switch Value	Low-Order Quotient Latch	Initial Shift Counter Value
Short	-	-	24
Long	8	-	32
Long	10	On	16
Long	10	Off	24
Long	12	On	24
Long	12	Off	24
Long	14	On	32
Long	14	Off	24

Figure 3-16. Initial Shift Counter Values for FP Common Divide - Single Arithmetic Cycles

When the divisor field is greater than 32 bits, each of the divide arithmetic cycles is made in two steps similar to those used for long-precision add or subtract instructions. This type of divide cycle is called the FP common divide - double arithmetic cycle, and is used for long-precision divide cycles when precision equals 10, 12 or 14 and the divide speed-up latch is off. These double arithmetic cycles are used for developing both the high-order and the low-order sections of the quotient. The initial value of the shift counter for each of these conditions is shown in Figure 3-17.

The divide single arithmetic cycles use one compute clock cycle for each loop (sequence 1B). The divide double arithmetic cycle loop uses three compute clock cycles for each loop (sequences 1B, 2B and 3B). Two of these cycles are used for the

VP Switch Value (Long-Precision Operation)	Low-Order Quotient Latch	Initial Shift Counter Value
10	Off	24
10	On	16
12	Off	24
12	On	24
14	Off	24
14	On	32

Figure 3-17. Initial Shift Counter Values for FP Common Divide - Double Arithmetic Cycles

arithmetic cycles and the third cycle is used for set-up and analysis. The shift counter is decremented on each loop and the divide cycles are completed when the shift counter equals zero.

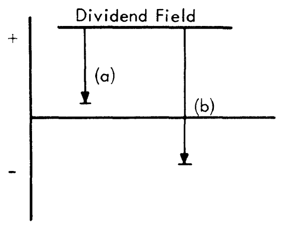
Principles of FP Divide

- Principles of FP divide are similar to those used on fixed-point divide with positive dividend.
- Division cycles are full reduction cycles when the dividend field is positive.
- Division cycles are combined correction and reduction cycles when the dividend field is negative.
- A quotient bit is developed if the result of the arithmetic cycle is positive.
- No quotient bit is developed if the result of the arithmetic cycle is negative.
- The type of arithmetic cycle is reversed if a full reduction cycle results in an overdraw (no quotient bit) or a combined correction and reduction cycle results in a quotient bit.
- The type of arithmetic cycle remains the same if a full reduction cycle is successful (quotient bit developed) or a combined correction and reduction cycle results in an overdraw (no quotient bit).
- The subtract trigger and B register extension sign bit are used respectively as divisor and dividend field sign bit.
- The carry-out of the extension sign bit position is used to determine the sign of the dividend field after the arithmetic operation.

The principles of floating-point divide are similar to those used in fixed-point divide for a positive dividend. That is, the original dividend field is positive and is reduced towards zero by the divisor field. Refer to Principles of Operation - Processing Unit, Form Y33-0002.

If the result of this full reduction cycle is positive, the cycle is termed a successful reduction and a quotient bit is developed. If the result of the full reduction cycle is negative, the cycle is termed an overdraw and no quotient bit is developed. See Figure 3-18.

After an overdraw, a combined correction and reduction cycle is performed as with fixed-point divide. Once again, if the result of this combined



(a) Successful Reduction (Quotient Bit)
(b) Overdraw (No Quotient Bit)

Figure 3-18. Example of Full Reduction Cycle

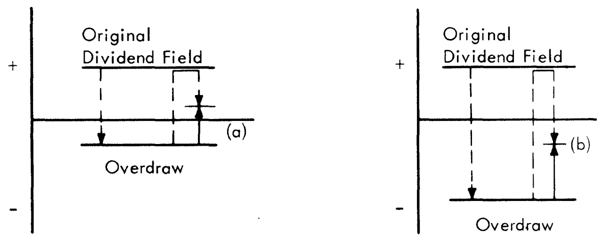
cycle is positive, a successful reduction has occurred and a quotient bit is developed. See Figure 3-19.

Thus a quotient bit is developed whenever the result of the divide arithmetic cycle is positive.

The first cycle of the divide operation is a subtraction cycle, the type of cycle is reversed either when an overdraw occurs on a full reduction cycle, or when a successful reduction occurs on a combined correction and reduction cycle. However, if the full reduction cycle is successful, or if the combined correction and reduction cycle results in an overdraw, the type of divide arithmetic cycle remains the same.

The floating-point fraction is always in true form and, when the fraction is placed in the arithmetic and logic section of the machine, the registers may contain up to 32 bits of data. Therefore, a sign bit position must be available, as an extension of the A and B registers to determine both the signs of the fields in these registers and the sign of the result.

As the original divisor fraction is placed in the A register in true form and the A register and subtract trigger are inverted in parallel, the state of the subtract trigger can be used to simulate an extension sign bit positon. That is, when the subtract trigger is on, the A register is inverted and the sign bit is one; when the subtract trigger is off, the A register is in true form and the sign bit is zero.



(a) Successful Reduction (Quotient Bit)
(b) Overdraw (No Quotient Bit)

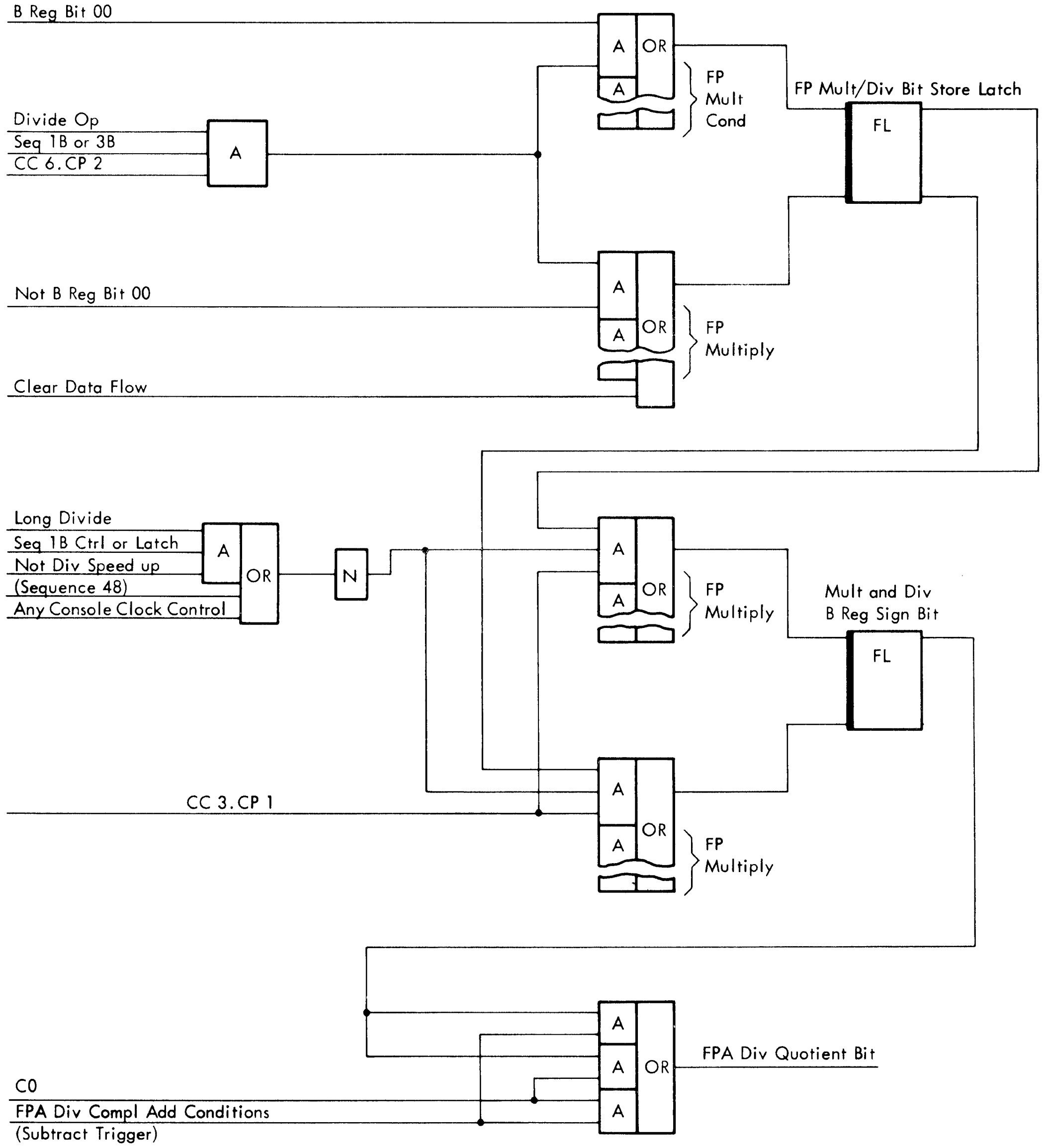
Figure 3-19. Examples of Combined Correction and Reduction Cycle

The dividend field in the B register uses the FP mult/div bit store latch and the B register extension sign bit to perform the function of an extension sign for the B register.

The FP mult/div bit store latch is used as a temporary storage for the data bit left-shifted from B register bit 00 to the B register extension sign bit. The data bit is stored so that the B register extension sign bit is not destroyed prematurely until all the analysis is performed which requires the state of the sign bit from the previous cycle. After this analysis, the content of the FP mult/div bit store latch is moved into the extension sign bit (Figure 3-20).

As with the fixed-point divide instruction a carry-out of the "sign bit" position during the arithmetic operation signifies a positive result and indicates that a quotient bit is developed. In fixed-point divide, this carry-out is the normal C0 signal whereas in floating-point divide, the C0 signal is only the carry-in to the extension sign bit position.

Therefore, the carry-out of the extension sign bit position will occur if any two of the following are a one: B register extension sign bit, subtract trigger (A register extension sign) or C0 signal. Refer to Figure 3-20. No arithmetic is actually performed with the extension sign bit positions. The resultant sign bit is not set into the B register



Refer also to ALD page KT 531

Figure 3-20. FP Quotient Bit Controls

extension sign bit since it would immediately be lost on the next shift-left operation. The true sign is determined by the carry-out of the extension sign bit position.

There are two conditions where the type of arithmetic cycle is reversed by inverting the A register and subtract trigger. The first condition takes place after a full reduction cycle (subtract trigger on) which gives a negative result. The negative result is signalled by 'no carry-out' of the extension sign bit position. As the A register extension sign (subtract trigger) is a one, a no-carry from the extension sign bit position will occur only if both the C0 bit and the B register extension sign bit are zero.

The second condition occurs after a combined correction and reduction cycle (subtract trigger off) which gives a positive result. The positive result is signalled by a carry-out of the extension sign position. As the A register extension sign (subtract trigger) is zero, a carry-out of the extension sign position can occur only if both the C0 bit and the B register extension sign bit are one.

Thus the A register and subtract trigger are inverted for the following two conditions:

Subtract trigger on:	C0 = 0, B reg extn sign bit = 0.
Subtract trigger off:	C0 = 1, B reg extn sign bit = 1.

The foregoing principles are used for both FP divide (single arithmetic cycles) and FP divide (double arithmetic cycles).

FP Common Divide, Single Arithmetic Cycles

- The controlling sequence is sequence 1B.
- The divisor is located in the A register.
- The dividend field is located in the B and BX registers.
- The quotient is developed in the scratch register.
- The quotient is developed in normalized form.
- All the principles of FP divide are followed.

For the flow chart and timing of these cycles, refer to FEMD Figures 6397 and 6398. The flow chart shows that the divide cycle starts with an immediate shift-left-one of the B, BX and scratch registers. A decrement of the shift counter is performed in parallel.

For the first cycle of a divide instruction, the 'not first divide' latch is off and the A register and subtract trigger are automatically inverted. For subsequent divide cycles, the 'invert A register and subtract trigger' logic is as described in "Principles of FP Divide". That is, the inversion occurs if the subtract trigger is on and both the B register extension sign bit and the C0 bit are zeros, or if the subtract trigger is off and both the B register extension sign bit and the C0 bit are ones.

After the arithmetic process, which is performed under control of the common add signal, a carry-out of the extension sign position signifies that the result is positive and that a quotient bit has been developed.

The conditions for this carry-out are described in "Principles of FP Divide" and occur when any two of the following are on: subtract trigger, C0 signal or B register extension sign bit. The logic is used to condition the shift-left entry to the bit 31 position of the scratch register. The quotient bit is entered into this position on the next shift-left-one operation at the beginning of the following cycle.

If no quotient bits are developed on the first four cycles of the high-order divide (scratch register bits 28 to 31 = 0, LO order quotient latch off), the value of the shift counter is increased by four and the value of the exponent is reduced by one. Division cycles recommence with the shift counter starting at its initial value. This process is repeated until the first quotient digit becomes significant. The divide loop then continues until the shift counter equals zero.

At the end of each cycle, the B register bit 00 is set into the FP mult/div bit store latch; this bit would normally be shifted into the B register extension sign bit position. The bit is not set from the FP mult/div bit store latch into the B register extension sign bit until just prior to the arithmetic process. Up to this time, the B register extension sign bit represents the original sign bit of the B register (from the previous cycle); the extension sign bit position does not take part in the arithmetic process as it is needed for the 'invert A register and subtract trigger' logic.

Note that on the initial entry to the divide loop, both the FP mult/div bit store latch and the extension sign bit are off. Note also that the 'not first divide' latch is off at the beginning of the high-order divide cycles on a long-precision divide and is not reset between the high-order and low-order divide cycles.

The quotient bit developed on the last divide cycle is inserted on a special shift-left-one operation of the scratch register on the cycle immediately following each of the common divide loops.

FP Common Divide, Double Arithmetic Cycles

- The controlling sequences are sequences 1B, 2B and 3B.
- The divisor is located in the A and AX registers.
- The dividend field is located in the B and BX registers.
- The quotient is developed in the scratch register.
- The quotient is developed in normalized form.
- All the principles of FP divide are followed.
- The arithmetic process is similar to that used in long-precision add or subtract instructions.

For the flow chart and timing of these divide cycles refer to FEMD Figures 6399 and 6400. The flow chart shows that the divide cycle starts with an immediate shift-left-one of the B, BX and scratch registers. A decrement of the shift counter is performed in parallel.

For the first cycle of the high-order quotient cycles, the 'not first divide' latch is off and the A register and subtract trigger are automatically inverted. For subsequent divide loops, the 'invert A register and subtract trigger' logic (as described in "Principles of FP Divide") is used.

Note that when invert conditions are present, both sections of the divisor have to be inverted. Note also that the subtract trigger is used to provide a carry-in to the high-order arithmetic process if a C0 signal occurred on the first low-order arithmetic process. The actual state of the subtract trigger is preserved over this period by setting its condition into the complement add latch (refer to Figure 3-2). Thus the status of the complement add latch is used instead of the actual subtract trigger in the 'invert A register and subtract trigger' logic. This means that the inversion occurs if the complement add latch is on and both the C0 and the B register extension sign bit latches are zeros, or if the complement add latch is off and both C0 latch and the B register extension sign bit latch are ones.

If, after the double arithmetic cycle, there is a carry-out of the extension sign position, a positive result is indicated and a quotient bit is developed. The conditions for this carry-out are described in "Principles of FP Divide" and occur when any two of the following are ones: complement add latch, C0 latch, or B register extension sign bit. This logic is used to condition the shift left entry to the bit 31 position of the scratch register. The quotient bit is entered into this position on the next shift-left-one at the beginning of the following cycle.

If no quotient bits are developed on the first four cycles of the high-order divide (scratch register bits 28 to 31 = 1, LO order quotient latch off), the value of the shift counter is increased by four and the division cycles recommence with the shift counter starting at its initial value. This process is repeated until the first quotient digit becomes significant. The divide loop then continues until the shift counter becomes zero. The B register bit 00 would normally be shifted into the B register extension sign bit position; however, at the end of each divide loop, the B register bit 00 is set into the FP mult/div bit store latch. The bit is then set from this latch to the extension sign bit just prior to the first arithmetic process. This transfer is delayed as the B register extension sign bit from the previous cycle is needed for the 'invert A register and subtract trigger' logic.

Note that the B register extension sign bit is not set with the sign of the result of the arithmetic process but remains the original B register sign from the previous cycle.

On the last divide cycle, when the shift counter equals zero with the LO order quotient latch on, the BX register is reset. The quotient bit developed on the cycles when the shift counter equals zero is inserted (in the scratch register), on a special shift-left-one operation of the scratch register, on the cycle immediately following each of the common divide loops.

16^n	n	16^{-n}
1	0	1. 0
16	1	0. 062 5
256	2	0. 003 906 25
4 096	3	0. 000 244 140 625
65 536	4	0. 000 015 258 789 062 5
1 048 576	5	0. 000 000 953 674 316 406 25
16 777 216	6	0. 000 000 059 604 644 775 390 625
268 435 456	7	0. 000 000 003 725 290 298 461 914 062 5
4 294 967 296	8	0. 000 000 000 232 830 643 653 869 628 906 25
68 719 476 736	9	0. 000 000 000 014 551 915 228 366 851 806 640 625
1 099 511 627 776	10	0. 000 000 000 000 909 494 701 711 053 237 915 039 062 5
17 592 186 044 416	11	0. 000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
281 474 976 710 656	12	0. 000 000 000 000 003 552 713 678 800 500 929 355 621 377 890 625
4 503 599 627 370 496	13	0. 000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
72 057 594 037 927 936	14	0. 000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
1 152 921 504 606 846 976	15	0. 000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
18 446 744 073 709 551 616	16	0. 000 000 000 000 000 000 054 210 108 624 275 221 700 372 640 043 497 085 571 289 062 5

CUT ALONG LINE

COMMENT SHEET

System/360 Model 44, Floating Point Feature
Field Engineering Theory of Operation Y33-0005

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.

How do you rate this manual: Excellent ____ Good ____ Fair ____ Poor ____

Suggestions 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 (LOCATED ON REVERSE SIDE), STAPLE AND MAIL fold fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM CORPORATION

P.O. BOX 390
POUGHKEEPSIE, N.Y. 12602

ATTENTION: FE MANUALS, DEPT. B96

fold fold

System
Maintenance
Library

System

CUT HERE

Y33-0005-0

S/360 Model 44 Printed in U.S.A. Y33-0005-0

IBM®

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