CPB-329A

GE-235 AUXILIARY ARITHMETIC UNIT

REFERENCE MANUAL

GENERAL ELECTRI

COMPUTER DEPARTMENT

Steven O. Hobb

GE-235
AUXILIARY
ARITHMETIC UNIT

JANUARY 1964

Revised June 1964

GENERAL ELECTRIC
COMPUTER DEPARTMENT

PREFACE

This manual describes the functions and operations of the GE-235 Auxiliary Arithmetic Unit (AAU). It tells how the AAU operates and gives a detailed description of all instructions used in writing the program. Included in the Appendix of the manual is a list of available AAU routines which can be obtained from the Computer Department Program Library, P. O. Box 2961, Phoenix, Arizona 85002.

Detailed information for the GE-235 system is found in the following manuals:

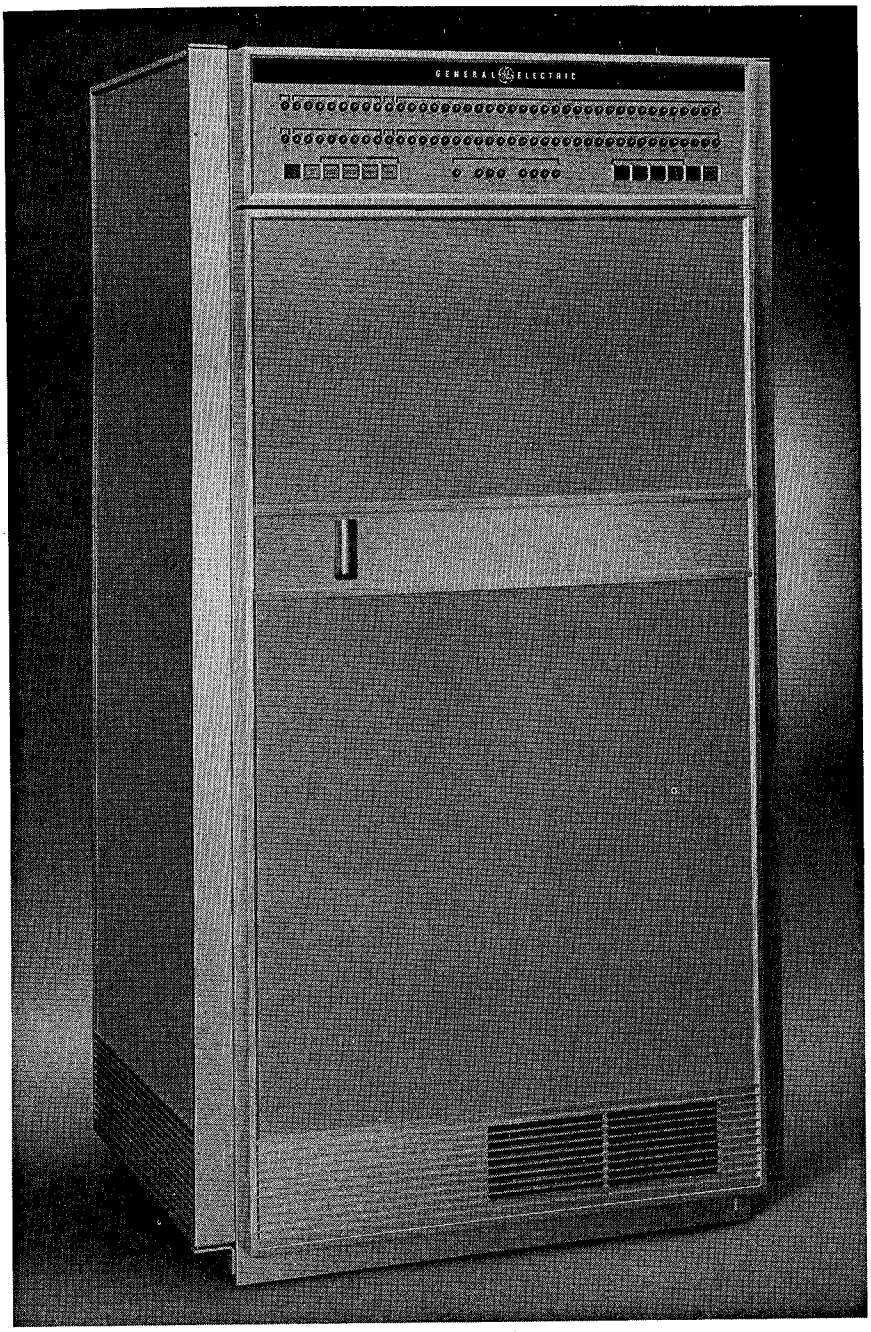
GE-235 System Manual (CPB-267A)

GE-235 Central Processor Reference Manual (CPB-374).

This publication is a new edition and supersedes the GE-235 Auxiliary Arithmetic Unit Reference Manual (CPB-329), which should no longer be used. Send comments about this publication to Technical Writing, General Electric Computer Department, Drawer 270, Phoenix, Arizona 85001.

CONTENTS

ILLUSTRATIONS



GE-235 AUXILIARY ARITHMETIC UNIT

1. GENERAL DESCRIPTION

The Auxiliary Arithmetic Unit (AAU) is one of the many devices designed to further increase the capabilities of the GE-235 Information Processing System to handle mass detailed data. It is an extension of the central processor with large registers permitting calculations of great complexity and provides high-speed, double precision fixed-point and floating-point arithmetic capability.

The arithmetic capability of the AAU includes add, subtract, multiply, divide, normalized and unnormalized floating-point operands and double length (40-bit) fixed-point operands. The central processor of the GE-235 system continues standard arithmetic operations when the system includes an AAU.

FUNCTIONS

The central processor of the GE-235 system performs the following major functions for the AAU:

1. Instruction retrieval
2. Address modification, if required
3. Partial instruction decoding
4. Operand retrieval and storage, when required
5. Control of transmission of instructions and data to the AAU
6. Synchronization of AAU operations with the central processor
7. Formulation of branch decisions based on AAU status indicators
8. Execution of a program interrupt based on AAU trap signals.

The AAU has its exclusive, high-speed data path to the central processor making it an extension of the arithmetic section and not a peripheral. Figure 1 illustrates in block diagram the GE-235 AAU subsystem.

COMPATIBILITY

The basic user-oriented design philosophy of the GE-235 system is the same as that of the GE-205, GE-215, and GE-225, which have proved themselves fast, accurate, reliable, and

economical in widely divergent fields. As a result of their design similarity, all programs and applications originally designed for the other members of the Compatibles/200 systems can immediately be processed on a GE-235 having the same system configuration.

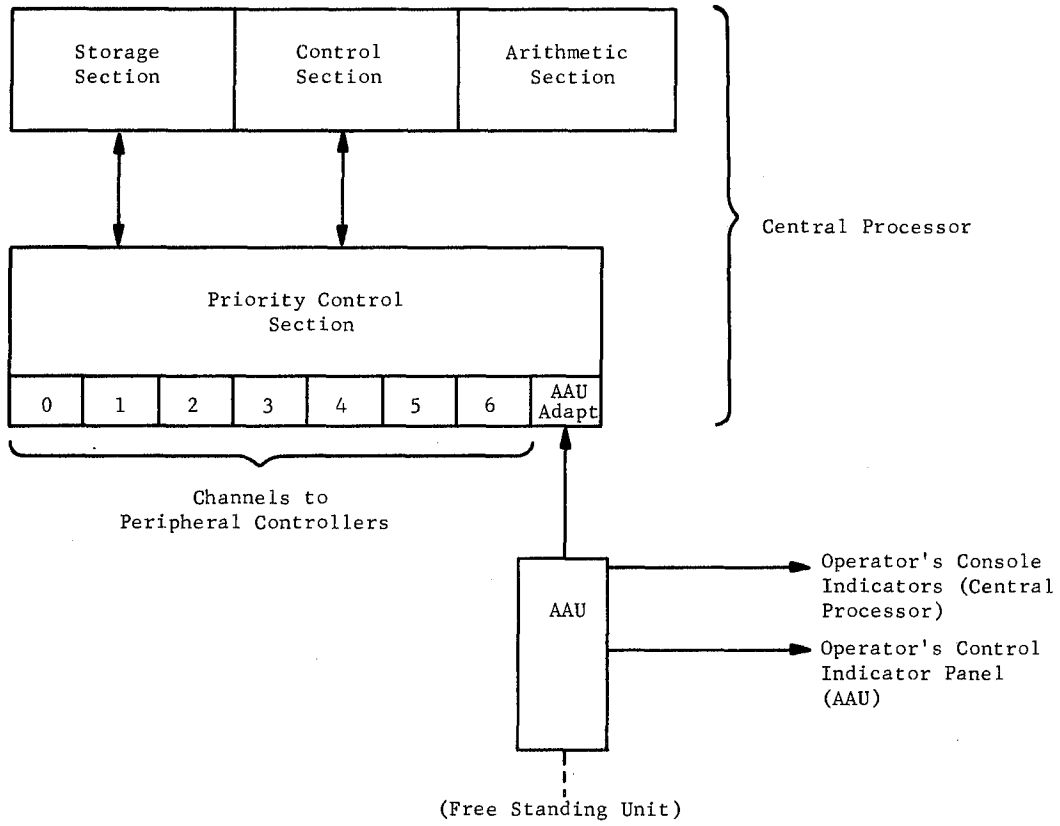


Figure 1. GE-235 AAU Subsystem

CONTROL

All AAU instructions and information (operand words) originate with the central processor. The AAU receives its instructions simultaneously with the central processor. The instruction remains in the I-Register of the central processor until all indexing and accessing of memory for that instruction are complete.

PRIORITY

The AAU is granted access to memory by the central processor priority control. The AAU decodes and executes its own instructions. All communication with the central processor memory is performed on a priority basis.

MODES

The AAU performs calculations--at the option of the programmer--in one of the following three arithmetic modes:

1. Normalized floating-point
2. Unnormalized floating-point
3. Double precision fixed-point

A trapping mode is provided on the GE-235 AAU. This is an important feature which saves program running time by eliminating separate checking instructions following each possibility of overflow, underflow, divide check, or illegal numbers. The trapping mode can be set along with any one of the arithmetic modes under program control.

The central processor is readily adaptable to computing ranges of numbers with fixed points, such as whole numbers or whole numbers with fixed fractions. (For example: dollars and cents where the fractional portion of the number is always two decimal digits.) When the calculations involve numbers that have varying format or floating point, such as in scientific calculations, the central processor would have to keep track of the fractional point (radix point) by program. This is a lengthy and time-consuming operation; however, the GE-235 AAU simplifies programming and saves time since the floating point calculation or consideration is made by the hardware as normalized or unnormalized.

Normalized Floating-Point Mode

A normalized number is one in which the most-significant nonzero digit of the fraction is next to the decimal point. For example, the decimal number 1234 would be:

$$.1234 \times 10^{4}.$$

Both positive and negative numbers are adjusted so that the fractional value of the number is in a prescribed range when using the normalized floating point mode. Normalized floating-point numbers are described in detail under "Data Words" in Chapter 2.

Unnormalized Floating-Point Mode

An unnormalized number is one in which zeros precede the most-significant nonzero digit fraction to the right of the decimal point. For example, the decimal number 001234 would be:

$$.001234 \times 10^{6}.$$

In scientific calculations, the decimal point can be any place in a number. The manner in which two numbers are aligned makes a great deal of difference in the answer if the decimal point is not in alignment. It is possible for the program to be written so as to align each and every decimal point for each and every calculation, but this process is cumbersome. The GE-235 AAU simplifies programming of whole and fractional number calculations, saving time in the process.

Double Precision Fixed-Point Mode

In the fixed-point mode, arithmetic operation in the AAU is the same as it is in the central processor with the exception that the AAU registers are double length. Fixed-point numbers can be in the forms shown below:

62478	(integer)
.62478	(fraction)
62.478	(mixed number)

Trapping Mode

This mode can be set or reset by the program with any one of the arithmetic modes. In the trapping mode, an AAU overflow, underflow, illegal number, or divide check will cause a special program interrupt in the central processor. The trapping mode, however, will not interfere with the operation of the Automatic Program Interrupt option in the GE-235 central processor.

REGISTERS

All AAU data words are first placed in the memory of the central processor as two 20-bit words, with bit meaning dependent upon the mode of operation selected. Executing a load operation will bring the two data words from memory through the central processor M-Register to be checked for parity. The AAU has a 40-bit buffer register which accepts the two 20-bit words to form one 40-bit AAU word, an Instruction Register to hold AAU instructions, a 40-bit Adder, and two other 40-bit registers, known as the AX- and QX-Registers. The size of these two registers permits both floating-point and fixed-point calculations on larger numbers than would normally be processed.

The AX-, BX-, QX-, and IX-Registers and the adder (SX) perform functions similar to their counterpart registers (A, B, Q, and I) in the GE-235 central processor. Figure 2 is a block diagram showing the data and control timing paths for the functional logic and the major registers of the GE-235 AAU.

AX-Register

The AX-Register is a 40-bit accumulator which performs the following functions:

- Holds the addend prior to addition and the most-significant bits of the sum after addition
- Holds the minuend prior to the subtraction and the most-significant bits of the difference after subtraction
- Holds the most-significant bits of the product after multiplication
- Holds the most-significant bits of dividend prior to division and the entire quotient after division
- Receives the 40-bit operand (two GE-235 computer words) from the central processor M-Register via the BX-Register during a load operation, and provides the 40-bit operand to be stored in memory via the BX- and M-Registers during a store operation.

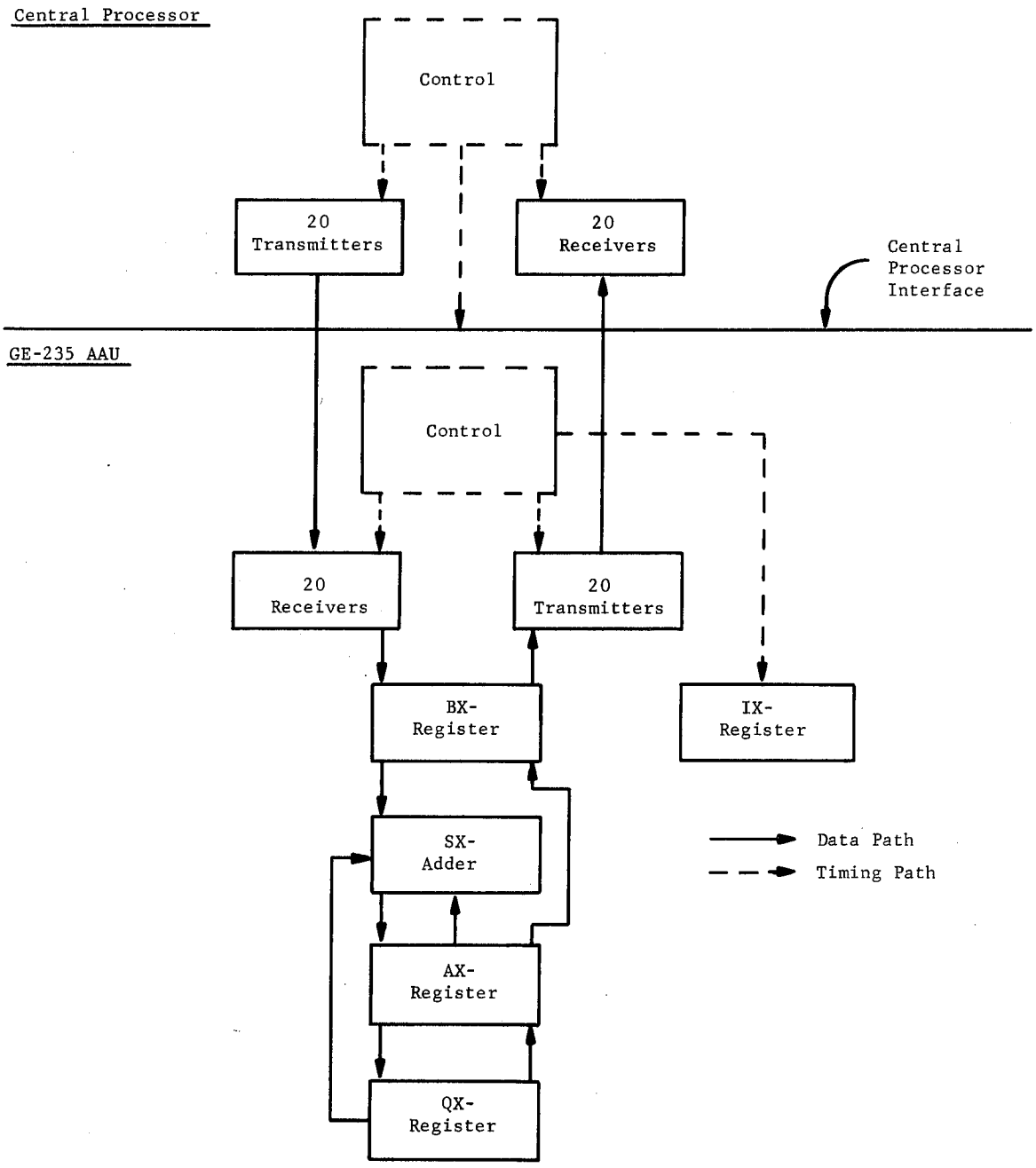


Figure 2. Major Registers And Functional Logic Diagram

BX-Register

The BX-Register is a 40-bit buffer register and distributor which performs the following functions:

- Holds the addend prior to addition
- Holds the subtrahend prior to subtraction
- Holds the multiplicand prior to multiplication
- Holds the divisor prior to division
- During a load operation, it receives two 20-bit words sequentially from the M-Register, assembles them into one 40-bit operand, and transmits the operand to the AX-Register. During a Store operation, it disassembles the 40-bit operand received from the AX-Register and transmits two 20-bit words, most significant half first, to the M-Register
- Performs all complementation that is required by the AAU.

QX-Register

The QX-Register is a 40-bit accumulator which performs the following functions:

- Conditionally holds the least-significant bits of the sum after addition
- Conditionally holds the least-significant bits of the difference after subtraction
- Holds the least-significant bits of the product after multiplication and the entire multiplier before multiplication
- Holds the least-significant bits of the dividend prior to division and the remainder after division.

IX-Register

The IX-Register is a 7-bit instruction register. It contains the current AAU command being executed. The IX-Register receives bits 2, 3, 4, 16, 17, 18, and 19 from the central processor I-Register.

SX-Register

The SX-Register, or adder, performs the following functions:

- Forty bit full binary adder during fixed point operations
- Thirty-one bit full adder for the mantissa during floating point operations
- Nine bit full adder for the exponent during floating point operations
- Adds the carry to the one's complement on all negate operations.

2. PROGRAMMING

The Auxiliary Arithmetic Unit is an on-line device and not a peripheral of the GE-235 system. It has its own channel (see Figure 1) for access to the central processor and has no separate controller.

FUNCTIONS OF THE CENTRAL PROCESSOR

The central processor performs the function of instruction retrieval. An immediate decision is reached if the retrieved instruction is for the AAU. Once the AAU is given an instruction to perform, it then operates independently of the central processor. The central processor is restrained from receiving priority until the AAU instruction is completed.

General Instructions

All general instructions are examined for zero content of bits 5-15. If all bits are zero, the instruction is immediately transferred to the AAU and executed.

Arithmetic and Load/Store Instructions

Either of these types of instruction may be address modified by the central processor. After possible indexing, the instruction is executed by the AAU using the central processor I-Register as the memory operand address register.

Test and Branch Instructions

If the instruction is a BAR (test and branch for the AAU), the central processor interrogates the various status indicators of the AAU. The AAU replies to the query and the central processor performs the appropriate branching decision.

TRAPPING MODE

The trapping mode is a standard feature of the GE-235 AAU, operating in a manner similar to the Automatic Program Interrupt (API) of the central processor. The trapping mode is not a normal mode of operation of the AAU. Unless this mode is set by program, the main program will not be interrupted in the event of overflow, underflow, divide check, or illegal number errors. A discussion of the use of the trapping mode will be found under "Programming Considerations," page 39.

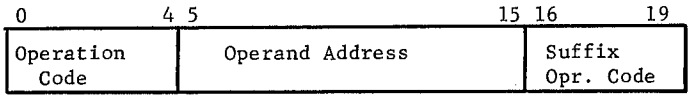
WORD FORMATS

The formats for instruction and data words for the GE-235 AAU are compatible with the GE-215/225 AAU.

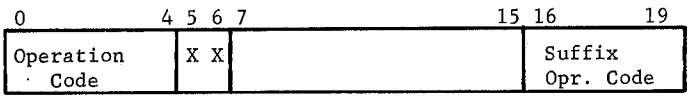
Instruction Words

Instruction words are contained in a single address word consisting of 20 bits. The basic formats for the 3 types of instruction words are as follows:

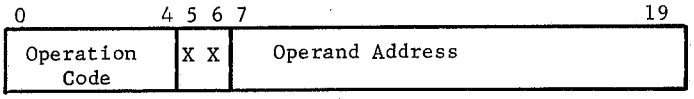
GENERAL INSTRUCTION WORDS



TEST-AND-BRANCH INSTRUCTION WORDS



ARITHMETIC INSTRUCTION WORDS



Data Words

Data for all AAU operations can exist in memory in any of three different modes--fixed-point, normalized floating-point, and unnormalized floating-point. All AAU data words exist in the central processor memory as two 20-bit words with the bits of each word having meaning according to the mode selected. Thus, when an instruction to load the AAU with the contents of memory location 3200 (FLD 3200) is received and executed by the AAU, the contents of 3200 and 3201 are brought into the AAU. The format in which the contents of 3200 and 3201 are interpreted depends upon the mode in which the AAU is operating.

FIXED-POINT WORDS in memory are in the format as shown in Figure 3. To illustrate, the FLD 3200 instruction is used. Note that the signs of word one and word two are identical for fixed-point words. Thus, when two data words from memory enter the AAU, they appear in the AX-Register as one 40-bit word. As shown, the fixed-point word in the AX-Register consists of 38 information bits, plus two identical sign bits. Similarly, a fixed-point word in the QX-Register also consists of 38 information bits and 2 sign bits.

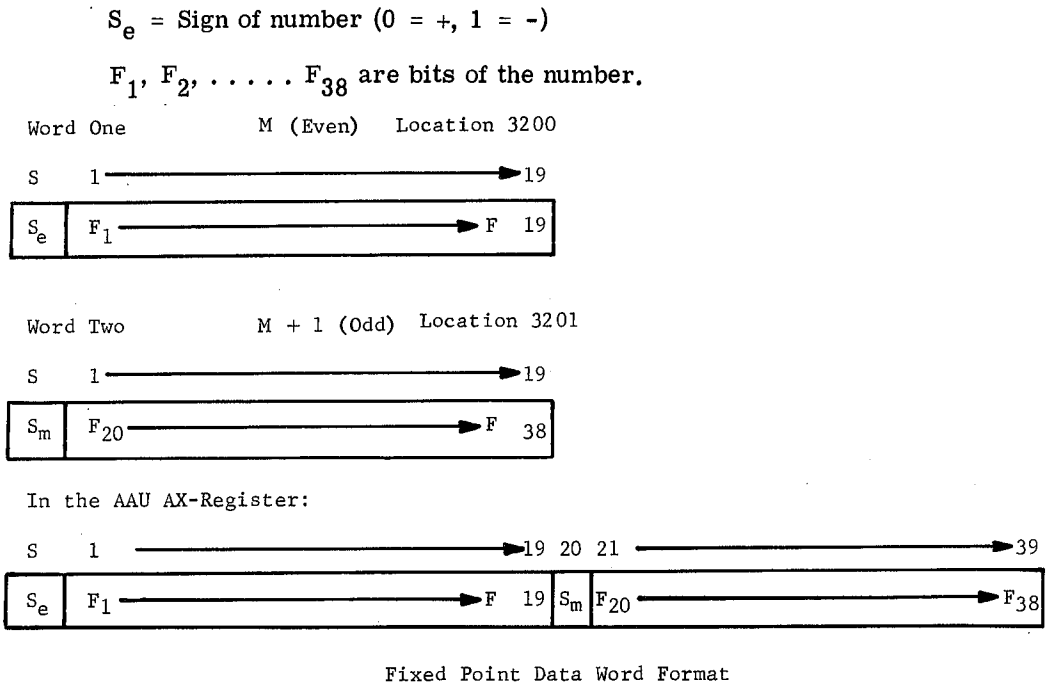
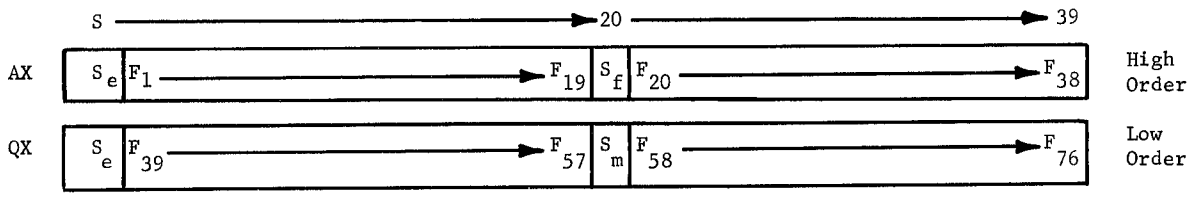


Figure 3. Fixed-Point Data Word Format

Bit one--S_e--is the sign bit of the entire word (38 bits), bit 20--S_m--has no significance in fixed-point words. Any arithmetic operation will cause bit position S_m to agree with bit position S_e.

If The Fixed-Point Number is the product of a multiply operation, it is stored in the AX- and QX-Registers in the format shown in Figure 4.



NOTE: F_{76} now has the value 2^0 and F_1 has the value 2^{75}. All signs set to agree.

Figure 4. Format of a "Product" of a Fixed-Point Number

An example of fixed-point numbers follows:

Value	AX-Register Bit Position S	1 - 19	20	21 - 39	Remarks
Largest Positive	0	1....1	0	1.....1	
+1	0	0....0	0	0.....1	
Zero	0	0....0	0	0.....0	
-1	1	1...1	1	1.....1	Two's complement notation for negative numbers.
Largest Negative	1	0...0	1	0.....1	

An Illegal Fixed-Point Number is not generated by the AAU, except as the result of overflow or underflow. If an arithmetic operation is attempted, the operation may or may not be performed; however, the underflow and illegal numbers are set.

Example: Illegal fixed-point numbers

AX-Register

S	1 - 19	20	21 - 39
1	0 0	0	0 0
1	0 0	1	0 0

FLOATING-POINT WORDS consist of two parts, an exponent and a mantissa. The 4 basic terms used in floating-point arithmetic operations are as follows:

1. Exponent. As used with the AAU, the exponent (or characteristic) is the 9 most-significant bits (8 numeric bits and a sign bit) of a double word (see Figure 5). These bits designate to what power of two the mantissa portion of the word must be raised.

2. Mantissa. In the AAU, the mantissa is the 30 least-significant bits of a double word. The radix point (see Radix Point described below) for these 30 bits is assumed to be to the left of the most significant of the 30 bits. Thus, the mantissa is fractional in value (see Figure 5).

   The mantissa is multiplied by two raised to the exponential power expressed in bits 0-8 to give the entire word the desired numeric value.

3. Radix Point. The point in any numbering system which separates the whole integers from the fraction. Thus, the decimal point is a radix point for the decimal system; a binary point is the radix point for the binary system. Because the computer and AAU operate in binary rather than in decimal digits, the term "decimal point" should be replaced by the term "binary point."

4. Normalization. In the AAU, positive and negative numbers are normalized, or adjusted, so that the mantissa lies in the prescribed range; the absolute value of the mantissa must be greater than (or equal to) 1/2 and less than 1. Algebraically, this is expressed as:

$$1/2 \leq (\text{mantissa}) < 1$$

Positive Numbers Are Normalized by shifting the mantissa left until its most-significant bit (bit 9 in the AX-Register) is a 1-bit. For each position shifted left, one is subtracted from the exponent.

Negative Numbers Are Normalized by shifting the mantissa left until its most-significant bit (bit 9 in the AX-Register) is a 0-bit or there is a 1-bit followed by zeros in all other mantissa bits. For each position shifted left, one is subtracted from the exponent.

The AAU generates normalized results of addition, subtraction, multiplication, and division in the AX-Register if it has been set into the normalized floating point mode by the programmer. The numbers to be operated on do not have to be in normalized form prior to the operation.

Floating-point numbers are stored in memory and in the AX-Register in the following forms and formats, as shown in Figure 5. The binary point is assumed to be before the first bit (bit 9) of the mantissa. This format produces a binary number with a 30-bit mantissa and a binary characteristic range of -256 to +255. This is approximately equal to a decimal number with a 9-digit mantissa and a decimal range of 10^{-77} to 10^{+77}. The fact that two words are used allows one of the sign positions to be applied to the exponent.

S_e = Sign of the exponent part

S_m = Sign of the mantissa (fraction) part

$E_1 \ldots E_8$ = Bits of the exponent

$M_9 \ldots M_{39}$ = Bits of the mantissa (fraction)

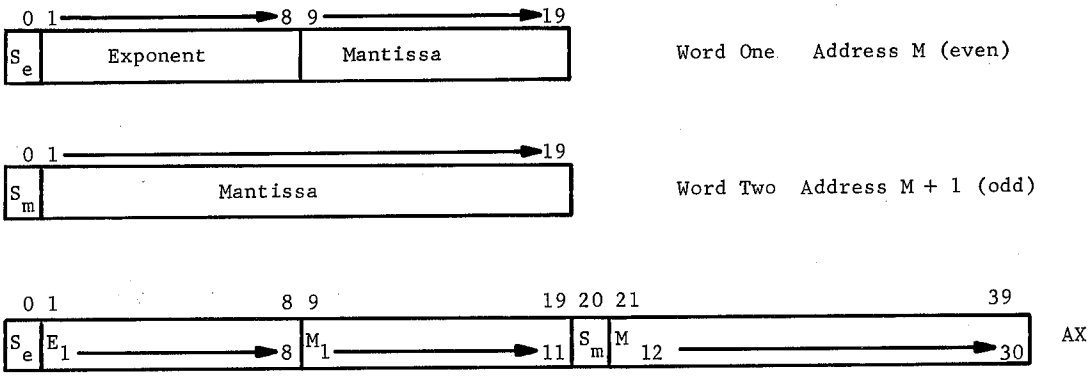


Figure 5. Floating-Point Data Word Format

If the Floating-Point Number is the result of multiplication, addition, or subtraction, the minor half (low order) appears in the QX-Register in the format shown in Figure 6. The major half (high order) will appear in the AX-Register.

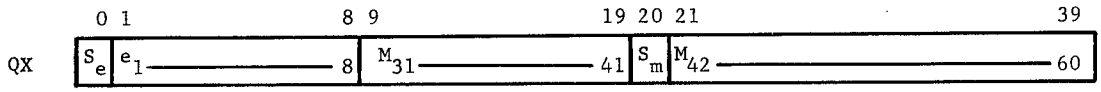


Figure 6. Floating-Point Number

The value of the QX exponent--e--is set equal to the value of the AX exponent--E--minus thirty (e=E-30). The sign of the QX fraction--S_m--is set to agree with the sign of the AX fraction--S_m.

The AAU operates in the fractional floating-point mode; that is, a fractional mantissa and an integral exponent. The use of any AAU instructions for floating point operations assumes that the memory operands are in the floating-point format.

An example of floating-point numbers follows:

	AX-Register Bit Positions				
	S | 1 8	9 19	20	21 39
Largest Positive	0 | 11.....11	11.....1.	.0.	.11.....11
+1	0 | 00.....01	10.....0.	.0.	.00.....00
Smallest Positive	1 | 00.....01	00.....0.	.0.	.00.....01
0	0 | 00.....00	00.....0.	.0.	.00.....00
Smallest Negative	1 | 00.....01	11.....1.	.1.	.11.....11
-1	0 | 00.....01	10.....0.	.1.	.00.....00
Largest Negative	0 | 11.....11	00.....0.	.1.	.00.....01

The "largest negative" number is the number farthest from zero with negative sign. The "smallest negative" is the number nearest to zero with negative sign.

An Illegal Floating-Point Number is one having a mantissa which is one past the range of the AAU in either the negative or positive direction. If any arithmetic operation is attempted, the complete operation will not be performed, the underflow and illegal number will be set. Illegal floating-point numbers are shown in the following examples:

Example 1.

	AX-Register Bit Positions			
	S	1 - 8	9 19	20	21 - 39
Illegal floating point number (s)	1	any	0.....0.	1..	0....0
	0	any	0.....0.	1..	0....0

This number has a mantissa which is one past the range of the AAU in the negative direction.

Example 2. The following number is treated as zero for any multiply or divide operation (not add or subtract). Any multiply or divide operation attempted will cause the exponent to be reset to zero. The operation will then proceed as though the number were zero.

Treated as zero

```
S 1 - 8 9 - 19 20 21 - 39
0 1.....1 0......0  0 0......0   or any legal exponent
```

Example 3. The following number has an exponent which is one past the range of the AAU in either the positive or negative direction. It may be loaded and stored and operated on arithmetically. The exponent will be treated as -256_{10}.

$1 \times 2^{\pm 256}$

```
S 1 - 8 9 10 - 19 20 20 - 39
1 0.....0 1 0.......0 0  0......0   or any legal mantissa
                                    except all zeros
```

SUBROUTINES are required initially to create floating-point words from BCD data or to create BCD words from floating-point format. This conversion from one to another is possible using conversion routines CD225C2.006/8 obtained from the GE Program Library or through the local sales office of the General Electric Computer Department.

EXPONENTIAL ARITHMETIC

To perform arithmetic operations in the floating-point format, several requirements must be met. For addition and subtraction problems, the exponents of the numbers involved must be equal. It is not probable that a common exponent will be used in all problems; however, the AAU automatically adjusts exponents. The AAU adjusts the exponent with the smaller numeric value. Adjustment is accomplished by automatically shifting the mantissa of the word with the smaller exponent right and incrementing its exponent by the number of positions shifted.

Addition and Subtraction

Once exponents are equalized, addition or subtraction of the mantissas can occur. The exponent of the answer is the adjusted exponent while the mantissa of the answer is the sum or difference of the shifted mantissas.

Multiplication

For multiplication, exponents are added and mantissas are multiplied. The resultant exponent in multiplication is the algebraic sum of the original exponents, while the resultant mantissa is the product of multiplying the original mantissas.

Division

In division, the exponent of the divisor is subtracted from the exponent of the dividend, and the mantissa of the dividend is divided by the mantissa of the divisor. The resultant exponent is the algebraic difference between the dividend and the divisor exponents. The resultant mantissa is the result of the algebraic division of the dividend mantissa by the divisor mantissa. In summary, floating-point division causes the subtraction of exponents and division of mantissas.

OVERFLOW AND UNDERFLOW in floating-point arithmetic operations, can result during both the normalized and unnormalized arithmetic modes.

Overflow occurs if the exponent of a final result in the AX-Register exceeds $+377_8$ ($+225_{10}$).

Underflow occurs when the exponent of a final result in the AX-Register becomes less than -400_8 (-256_{10}).

OVERFLOW OR UNDERFLOW can result at any of these times:

- During the formation of the initial estimate of the result exponent
- During a right shift one and add one as a result of mantissa overflow
- During the normalization of a result (normalized mode). Normalization involves shifting the mantissa left N places and subtracting N from the exponent, possibly causing underflow, N being the number of leading zeros in a positive mantissa or leading ones in a negative mantissa.

No single AAU instruction will ever result in setting both overflow and underflow.

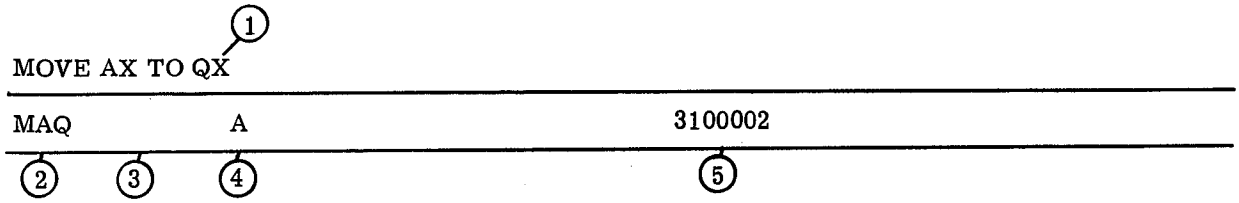
Whenever underflow occurs, the AX-Register is cleared to zero if the AAU is in either the normalized or unnormalized floating-point modes. This assures that zero is always a valid replacement for the true result when underflow occurs. The clearing of AX does not cause reset of the underflow.

INSTRUCTIONS

Instructions for the GE-235 AAU are divided into 3 categories, as follows:

1. General Instructions:
   a. Control (sets mode of operation)
   b. Data Transfer (load/store) within the AAU
   c. Reset (underflow/overflow)
   d. Normalization
2. Arithmetic Instructions:
   a. Arithmetic operations: add, subtract, multiply, and divide
   b. Data Transfer (load/store) between the AAU and central processor
3. Test-and-Branch Instructions.

The description of instructions in this manual use the following format, explained in the key below.



1. Name of instruction (operation to be performed).
2. General Assembly Program mnemonic operation code.
3. General Assembly Program Operand field symbol:
   a. M - indicates that the operand field in this instruction is occupied by the address of a memory location (address may be either actual or symbolic).
   b. A three-character mnemonic code indicates the specific condition (true or false) of a test and branch instruction.
4. General Assembly Program symbol X field:
   a. X - indicates that the address in the operand field of the instruction may be automatically modified by using address modification words. Omission of this symbol indicates the instruction cannot be modified in this way.
   b. A - indicates that the AAU registers are affected by the instruction in a similar manner to the registers of the central processor.
5. Representation of the machine coding of the instruction in octal notation.

General Instructions

General Instructions do not require memory reference. The operation codes are defined by bits S-4 and 16-19 of the instruction word, with all other bits zero. The general instructions are further defined by type of instruction, as follows:

1. Control
2. Data Transfer (within the AAU)
3. Reset
4. Normalize

CONTROL INSTRUCTIONS set the mode of operation to be performed upon the data words received. The GE-235 AAU is set to normalized floating-point mode under the 3 following conditions:

1. Power-on time
2. When the CLEAR switch is depressed on the AAU
3. When the RESET MODES switch is depressed on the central processor console.

It is necessary to set the mode of operation by a SET MODE instruction before giving an arithmetic instruction to obtain positive control over the AAU. Once a SET MODE instruction is executed, the AAU executes all arithmetic instructions in that mode until the mode is changed by another SET MODE instruction.

SET FIXED-POINT MODE

SET	FIXPOINT	3500010

The AAU is set to execute AAU arithmetic instructions in double precision fixed-point mode.

SET NORMALIZED FLOATING-POINT MODE

SET	NFLPOINT	3100010

The AAU is set to execute AAU arithmetic instructions in normalized floating-point mode.

SET UNNORMALIZED FLOATING-POINT MODE

SET	UFLPOINT	3200010

The AAU is set to execute AAU arithmetic instructions in unnormalized floating-point mode.

SET TRAPPING MODE

SET	TRPMODE	3100001

The AAU is set to interrupt the central processor upon detection of subsequent overflow, underflow, or divide check conditions. An illegal number causes an underflow condition in the trapping mode.

The overflow hold, underflow hold, divide check, and illegal number statuses are not reset by this instruction.

The AAU remains in the trapping mode until execution of a SET NTPMODE instruction or until the RESET MODES switch on the central processor console is depressed.

SET TRAPPING MODE OFF

SET	NTPMODE	3200001

The AAU trapping mode is turned off.

The overflow hold, underflow hold, divide check, and illegal number statuses are not affected by this instruction.

DATA TRANSFER INSTRUCTIONS are similar to certain central processor data transfer instructions. They involve transfer of data between arithmetic registers within the AAU and are specified by mnemonic codes similar to those for the central processor. AAU data transfer instructions are identified by placing the letter A in the X column (card column 20) of the General Assembly Program coding sheet.

CLEAR AX-REGISTER

CAX	A	3200005

The AX-Register of the AAU is cleared to all zeros.

CLEAR QX-REGISTER

CQX	A	3500005

The QX-Register of the AAU is cleared to all zeros.

LOAD AX FROM QX

LAQ	A	3600002

The contents of the QX-Register replace the contents of the AX-Register. The contents of the QX-Register are unchanged.

LOAD QX FROM AX

LQA	A	3200002

The contents of the AX-Register replace the contents of the QX-Register. The contents of the AX-Register are unchanged.

MOVE AX TO QX

MAQ	A	3100002

The contents of the AX-Register replace the contents of the QX-Register. The contents of the AX-Register are reset to zeros.

EXCHANGE AX AND QX

XAQ	A	3500002

The contents of the AX- and QX-Registers are interchanged.

RESET INSTRUCTIONS are used to reset the OVERFLOW HOLD, UNDERFLOW HOLD, and DIVIDE CHECK indicators.

RESET INDICATORS	
RIN	3500004

The OVERFLOW HOLD, UNDERFLOW HOLD, and DIVIDE CHECK indicators are turned off. The internal illegal number indicator is also turned off.

RESET OVERFLOW HOLD	
ROV	3100004

The OVERFLOW HOLD indicator is turned off.

RESET UNDERFLOW HOLD	
RUN	3200004

The UNDERFLOW HOLD indicator is turned off.

The OVERFLOW HOLD, UNDERFLOW HOLD, and DIVIDE CHECK indicators will have been set as a result of overflow, underflow, divide check and illegal number conditions.

NORMALIZE INSTRUCTION. This instruction operates correctly only on floating-point data. The instruction itself is not mode dependent, and operates correctly in all three arithmetic modes.

NORMALIZE AX AND QX	
NOX	3100005

The floating-point operand in the AX- and QX-Registers is normalized by shifting the mantissa left until a 1-bit is detected in the most-significant bit position, AX (9) for positive numbers, or a 0-bit is detected in AX (9) for negative numbers. For each bit position shifted, one is subtracted from the exponent in AX. Bits from QX (9) shift into position AX (39). After normalization, the exponent in the QX is set to 30 less than the exponent of AX. The sign of QX is set to the original sign of the AX.

Zero is defined as a normalized number.

Arithmetic Instructions

Arithmetic instructions (add, subtract, multiply, and divide) include data transfer (load/store) operations on data words according to the mode (fixed, normalized, or unnormalized) as specified by a control SET MODE instruction. Figure 7 is an illustration of the initiation of an instruction.

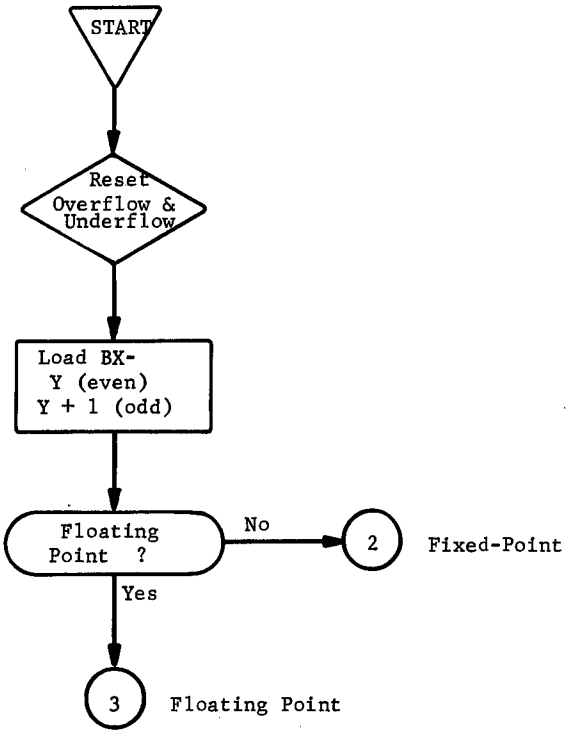


Figure 7. Initiation of Arithmetic Instruction

Bits S-4 of the instruction word defines the operation to be performed. Bits 5-6, if set, specify index word modification. Bits 7-19 specify the operand (memory) address of the data word.

If index word modification is specified, the central processor performs the required index modification. The instruction is then executed by the AAU using the central processor I-Register as the memory operand address register.

If index word modification is not specified, the operand address must be an even numbered location in memory and be greater than 15 for proper execution. If the address is odd, either the contents of the odd location will be loaded into both halves of the BX-Register, or the contents of BX will be stored so that first the most-significant half of BX (S-19) appears in the odd location and then BX (20-39) is written over BX (S-19) in the same odd location.

Floating point operations assume that the operands of the data word are already in floating-point format. If the operand is not in floating-point format, it can be converted by means of a subroutine furnished for this purpose (CD225C1.006/8).

ARITHMETIC OPERATIONS are performed by the AAU upon operands whose memory address is greater than 15, unless the instruction is index modified. Instructions for arithmetic operations are: FAD, FSU, FMP, and FDV.

Each instruction is described for each of the 3 operating modes: fixed, normalized, and unnormalized.

ADD

FAD	M	X	31MMMMM

The contents of memory location M and M+1 are added algebraically to the contents of the AX-Register. The contents of M and M+1 are unchanged.

Fixed-Point Mode

- The sum of the contents of M and M+1 plus the contents of AX is left in AX-Register with bits 0 and 20 (sign bits) set to agree
- QX Register is unchanged
- Overflow is set if the capacity of the AX-Register is exceeded in a positive direction during the add operation. The AX-Register remains in an overflow condition.
- Underflow is set:
  1. If the capacity of the AX-Register is exceeded in a negative direction during an add operation. The AX-Register remains in an underflow condition.
  2. If an illegal number is detected at the start of the operation in either the AX- or BX-Registers. An illegal number causes both underflow and illegal number conditions to be set.

Normalized Floating Point Mode

- The floating-point number in M and M+1 are loaded into the BX-Register
- QX Register is cleared
- A check is made to determine if either the AX- or BX-Register contain an illegal number in the mantissa. If either or both do, the arithmetic operation is terminated with an UNDERFLOW (and illegal number) indication in the AAU
- The AX- and BX-Registers are examined for zero content and if either are found to contain zeros, the nonzero number is placed in the AX-Register and the contents are normalized. If neither AX nor BX are zero, then the exponents are compared
- If the exponent of the number in BX is algebraically smaller than the exponent in AX, BX and AX are interchanged--the smaller exponent is placed in AX
- Exponents are aligned by right shifting the AX-Register, adding one to the exponent for each position shifted, until the exponents are equal. Bits shifted out of AX-39 are shifted into QX-9; bits shifted out of QX-39 are lost
- When the exponents are aligned, the mantissas are algebraically added and mantissa overflow, if it occurred, is corrected. The sum of the mantissas in AX and QX is normalized

- Mantissa overflow is corrected by shifting the mantissa one bit position to the right and adding one to the exponent

- AX and QX mantissas are examined for zero content and, if found to contain zeros, the exponents are cleared and the operation is terminated. The AX- and QX-Registers are cleared

- If AX and QX mantissas are not zeros, the sum in the AX- and QX-Registers is normalized. Leading zeros (for positive mantissas) or ones (for negative mantissas) are stripped by left-shifting AX and QX and subtracting from the exponent. Bits from QX-9 are reintroduced into AX-39; AX and QX are left-shifted together. Zeros are introduced into QX-39.

- Exponent of QX is set to AX_E-30 so that the minor half of the sum will be properly scaled. If QX_E underflows, no error is set.

- The QX mantissa sign (bit 20) is set to the same value as the AX mantissa sign

- A check is made to determine if overflow or underflow occurred

- Overflow is set if the exponent sum in the AX-Register exceeds $+255_{10}$. AX- and QX-Register remain in an overflow condition

- Underflow is set:

  1. If an illegal number is detected in the mantissa at the start of the operation in either the AX or BX Registers

  2. If the exponent sum in AX is less than -256_{10}. The AX- and QX-Registers are cleared.

Unnormalized Floating-Point Mode. Same conditions prevail as set forth for normalized floating point mode, except the sum of mantissas in the AX- and QX-Registers is left in unnormalized form.

An execution of a fixed point add operation is shown in Figure 8.

Figures 9 and 10 illustrate the execution of a floating-point add operation.

Figure 11 shows how holds are set and the initiation of the trap operation takes place.

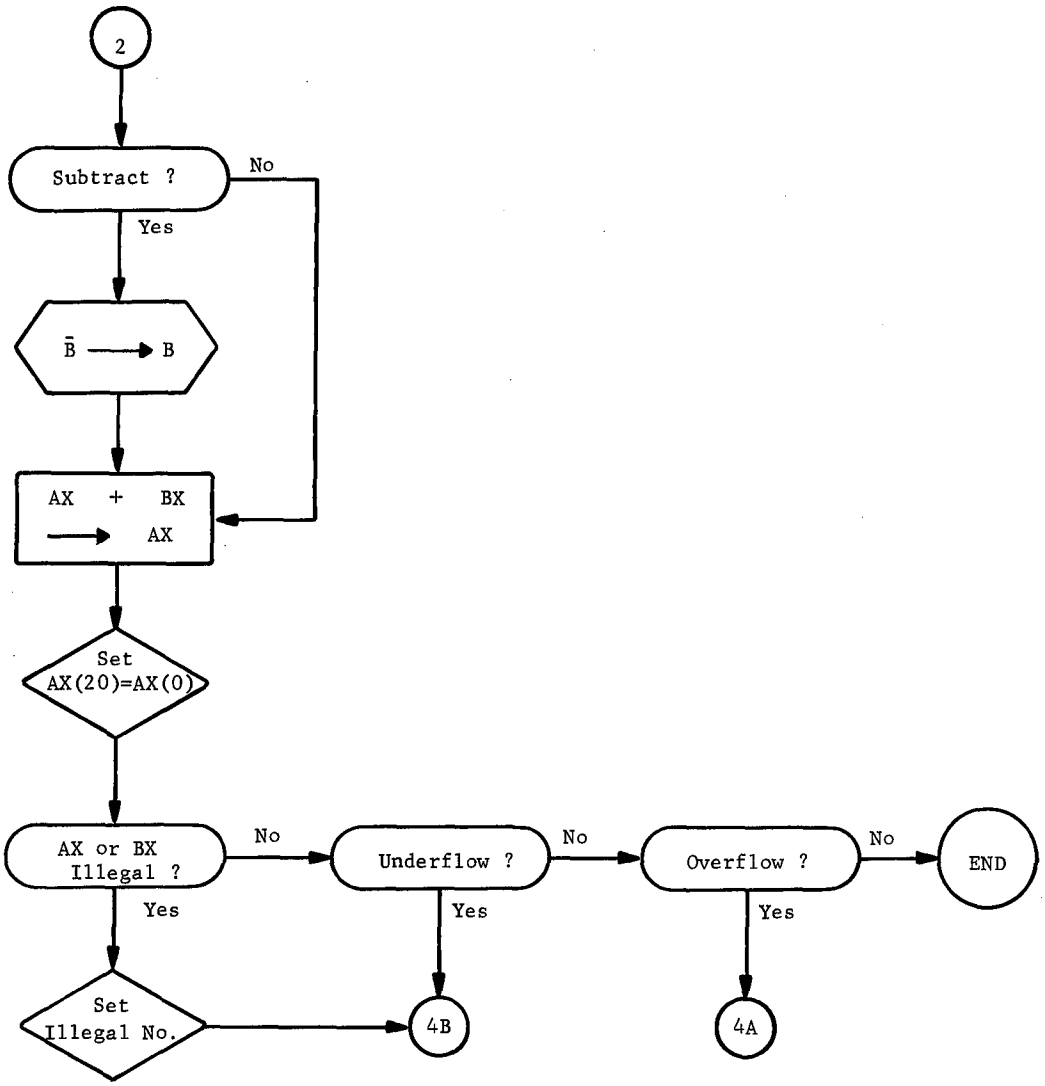


Figure 8. Fixed-Point FAD, FSU

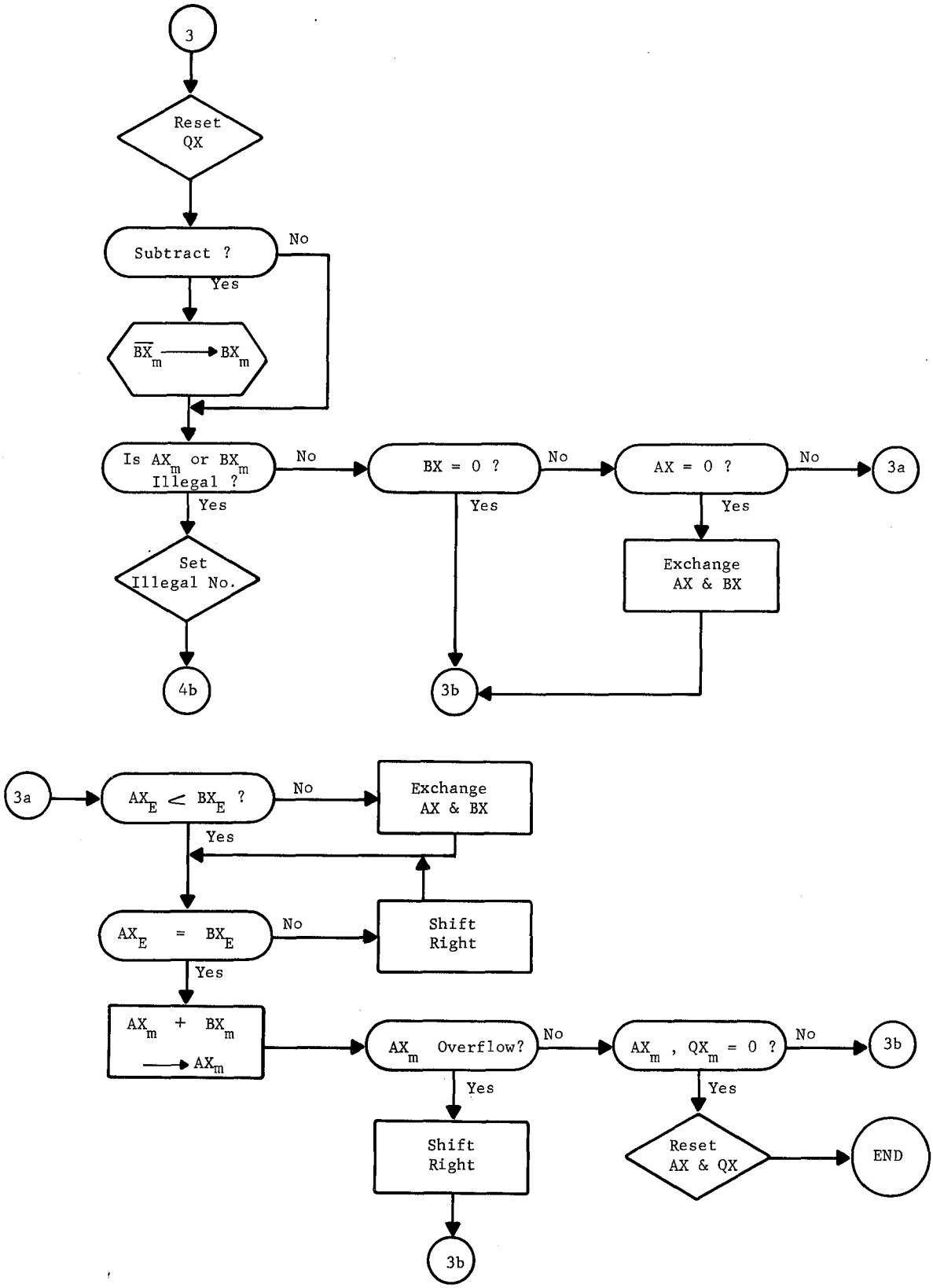


Figure 9. Floating-Point FAD, FSU

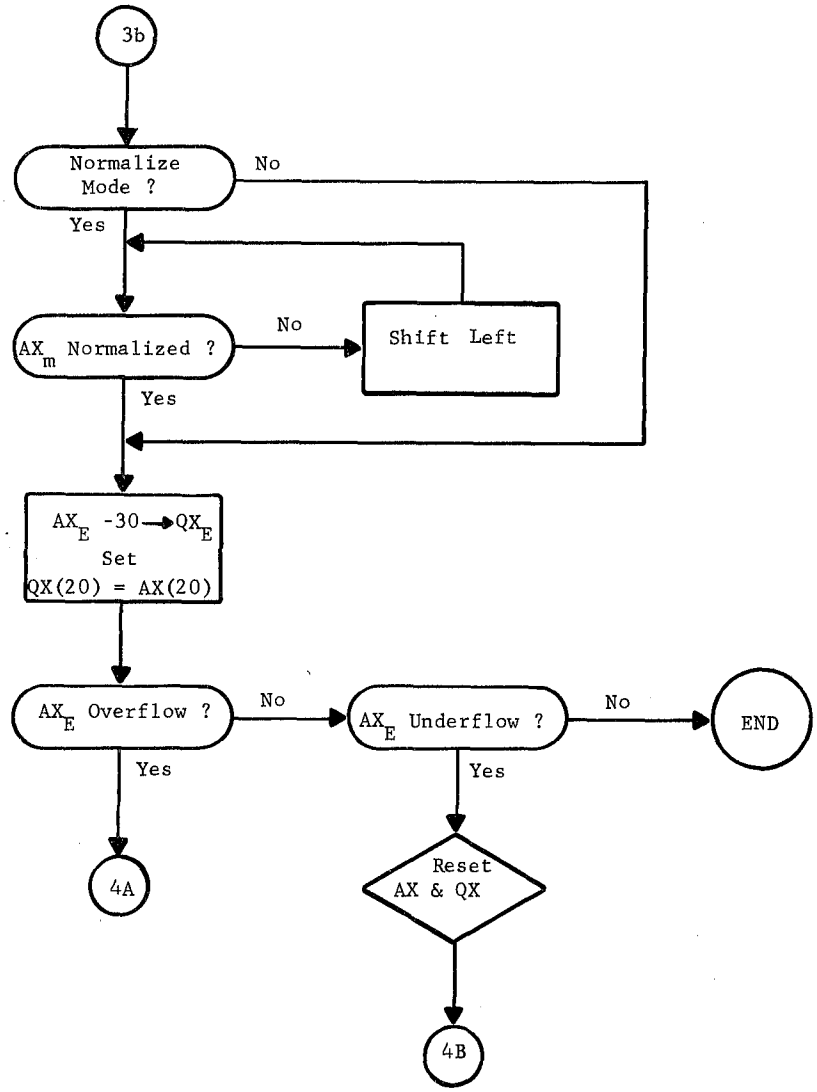


Figure 10. Normalization, Floating-Point Numbers

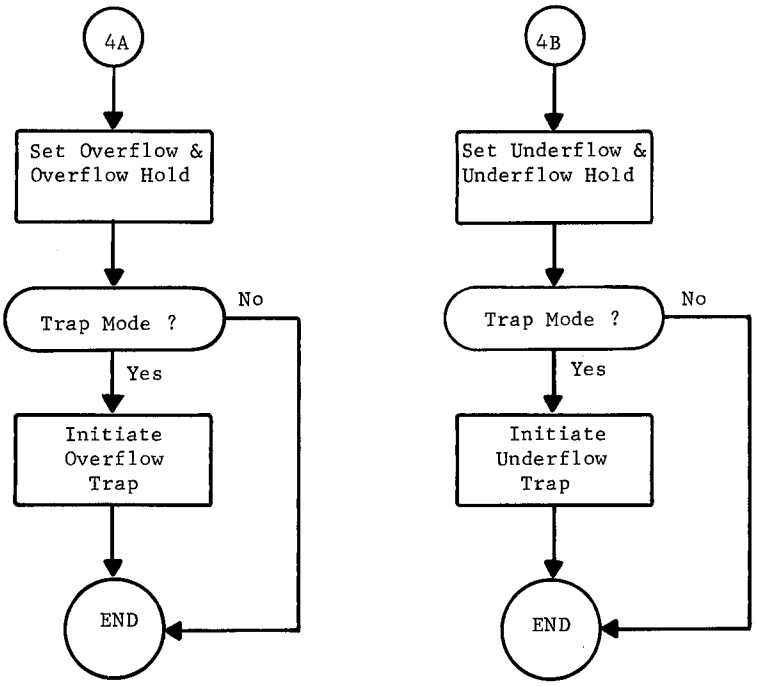


Figure 11. Setting Holds and Initiation of Trap Operation Mode

SUBTRACT

FSU	M	X	32MMMMM

The contents of memory location M and M+1 are algebraically subtracted from the contents of the AX-Register. The contents of M and M+1 are unchanged.

Fixed-Point Mode

- The contents of M and M+1 are loaded into the BX-Register. BX is complemented and added algebraically to the contents of the AX-Register. The difference is left in the AX-Register with bits 0 and 20 (sign bits) set to agree
- The QX-Register is unchanged
- Overflow is set if the capacity of the AX-Register is exceeded in a positive direction during a subtract operation. The AX-Register remains in an overflow condition
- Underflow is set:
  1. If the capacity of the AX-Register is exceeded in a negative direction during a subtract operation. The AX-Register remains in an underflow condition.
  2. If an illegal number is detected at the start of the operation in either the AX- or BX-Registers.

Normalized Floating-Point Mode

- The floating-point number in M and M+1 is loaded into the BX-Register. The BX mantissa is complemented
- The QX-Register is cleared
- A check is made to determine if either the AX- or BX-Register contains an illegal number in the mantissa. If either or both do, the arithmetic operation is terminated with an UNDERFLOW (and illegal number) indication in the AAU
- If both registers contain legal numbers, they are examined for zero content and if either one is found to contain zeros, the nonzero number is placed in the AX-Register and the contents are normalized. If neither register contains zeros, then the exponents are compared
- If the exponent of the number in BX is algebraically smaller than the exponent in AX, BX and AX are interchanged--the smaller exponent is placed in AX
- Exponents are aligned by right-shifting the AX-Register, adding one to the exponent for each position shifted, until the exponents are equal. Bits shifted out of AX-39 are shifted into QX-9; bits shifted out of QX-39 are lost
- When exponents are aligned, the mantissas are algebraically added and mantissa overflow, if it occurred, is corrected. The sum of the mantissas in AX and QX is normalized.
- Mantissa overflow is corrected by right-shifting the mantissa one bit position and adding one to the exponent.
- AX and QX mantissas are examined for zero content and if found to contain zeros the exponents are cleared and the operation is terminated. The AX- and QX-Registers are cleared
- If AX and QX mantissas are not zeros, the difference in the AX- and QX-Registers is normalized. Leading zeros (for positive mantissas) and ones (for negative mantissas) are stripped by left-shifting AX-QX and subtracting from the exponent; bits from QX-9 are reintroduced into AX-39; AX and QX are left-shifted together. Zeros are introduced into QX-39
- Exponent of QX is set to AX_E-30 so that the minor half of the difference will be properly scaled. If QX_E underflows, no error is set
- The QX mantissa sign bit (20) is set to the same value as the AX mantissa sign
- A check is made to determine if overflow or underflow occurred
- Overflow is set if the exponent result in AX-Register exceeds $+255_{10}$. AX- and QX-Registers remain in an overflow condition
- Underflow is set:

  1. If an illegal number is detected in the mantissa at the start of the operation in either the AX or BX-Registers.
  2. If the exponent result in AX is less than -256_{10}. The AX- and QX-Registers are cleared.

Unnormalized Floating-Point Mode. Same conditions prevail as set forth for normalized floating-point mode, except: difference of mantissas in the AX- and QX-Registers is left in unnormalized form.

Figure 8 illustrates the execution of the FSU instruction during a fixed-point mode of operation.

Figures 9 and 10 illustrate the execution of the FSU instruction during the floating-point mode of operation.

MULTIPLY

FMP M X	35MMMMM

The contents of memory location M and M+1 are multiplied algebraically by the contents of the QX-Register. The contents of M and M+1 are unchanged.

Fixed-Point Mode

- M and M+1 are loaded into the BX-Register. The AX-Register is cleared. Bit 20 of the QX-Register is set equal to QX-0
- A check is made to see if either BX- or QX-Registers contain an illegal number. If neither or both do, the arithmetic operation is terminated with an UNDERFLOW (and illegal number) indicators in the AAU
- If both Registers contain legal numbers, they are examined for zero content and if either or both contain zeros, the AX- and QX-Registers are cleared and the arithmetic operation is terminated
- If the BX- and QX-Registers contain legal nonzero numbers they are multiplied and their product is stored in the AX- and QX-Registers. The product is a 76-bit fixed-point number, right adjusted in AX and QX with four identical sign bits
- Overflow is not possible with legal nonzero numbers, therefore overflow is not tested
- Underflow is set if an illegal number is detected at the start of the operation in either the BX- or QX-Registers.

Normalized Floating-Point Mode

- M and M+1 are loaded into the BX-Register. The AX-Register is cleared
- The exponent of QX is transferred to AX_E
- A check is made to determine if BX_m or QX_m contain an illegal number. If either or both do, the operation is terminated and an UNDERFLOW (and illegal number) indication in the AAU
- If both registers contain legal numbers, the exponents of AX and BX are added and stored in AX_E. This completes the addition of the exponents operation and the sum of QX_E and BX_E and stored in AX_E

- The BX and QX mantissas are examined for zero content and, if found to contain zeros, the operation is terminated with AX- and QX-Registers cleared
- If the mantissas of BX and QX are nonzero, legal numbers, the mantissas in AX and QX are shifted right together until a 1-bit is detected in position QX-39. Addition or subtraction of AX_m plus BX_m forms the first partial product in AX and continues in this fashion until a 60-bit product is formed in AX_m and QX_m. The products is normalized and the exponent in QX-Register is set to AX_E-30. If QX_E underflows, no error is set. The mantissas signs QX(20) and AX(20) are set to agree
- A check is made to determine if overflow or underflow occurred
- Overflow is set if the product in the AX-Register exceeds $+255_{10}$
- Underflow is set:

  1. If an illegal number is detected in the mantissa at the start of the operation in either the QX- or BX-Registers.
  2. If the exponent product in AX is less than -256_{10}. The AX- and QX-Registers are cleared.

Unnormalized Floating-Point Mode. Same condition prevails as set forth for normalized floating-point mode, except:

Product in AX- and QX-Registers is left in unnormalized form.

Figure 12 illustrates the execution of the FMP instruction during a fixed-point mode of operation.

Figure 13 illustrates the execution of the FMP instruction during a floating-point mode of operation.

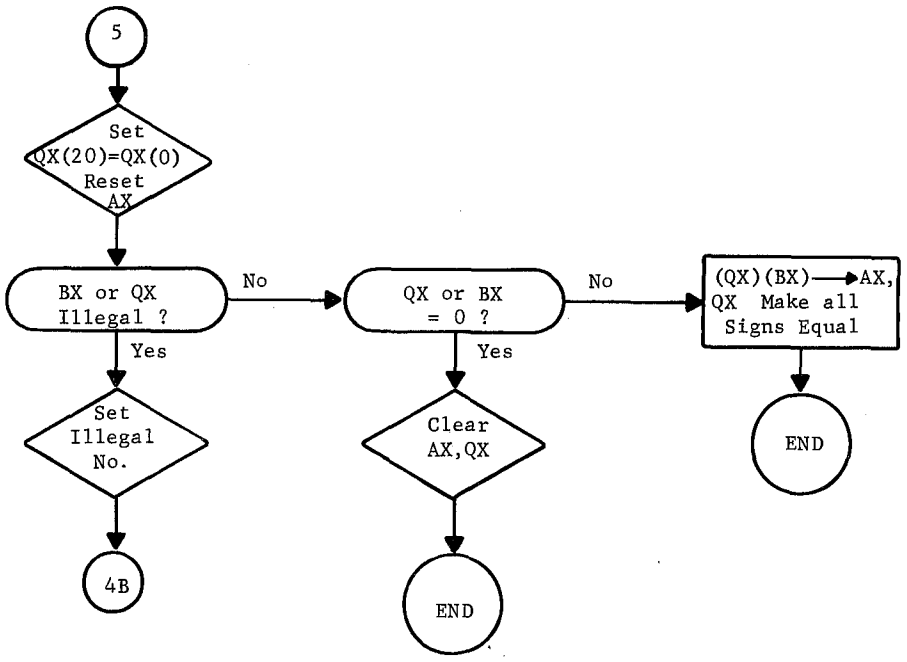


Figure 12. Fixed-Point FMP

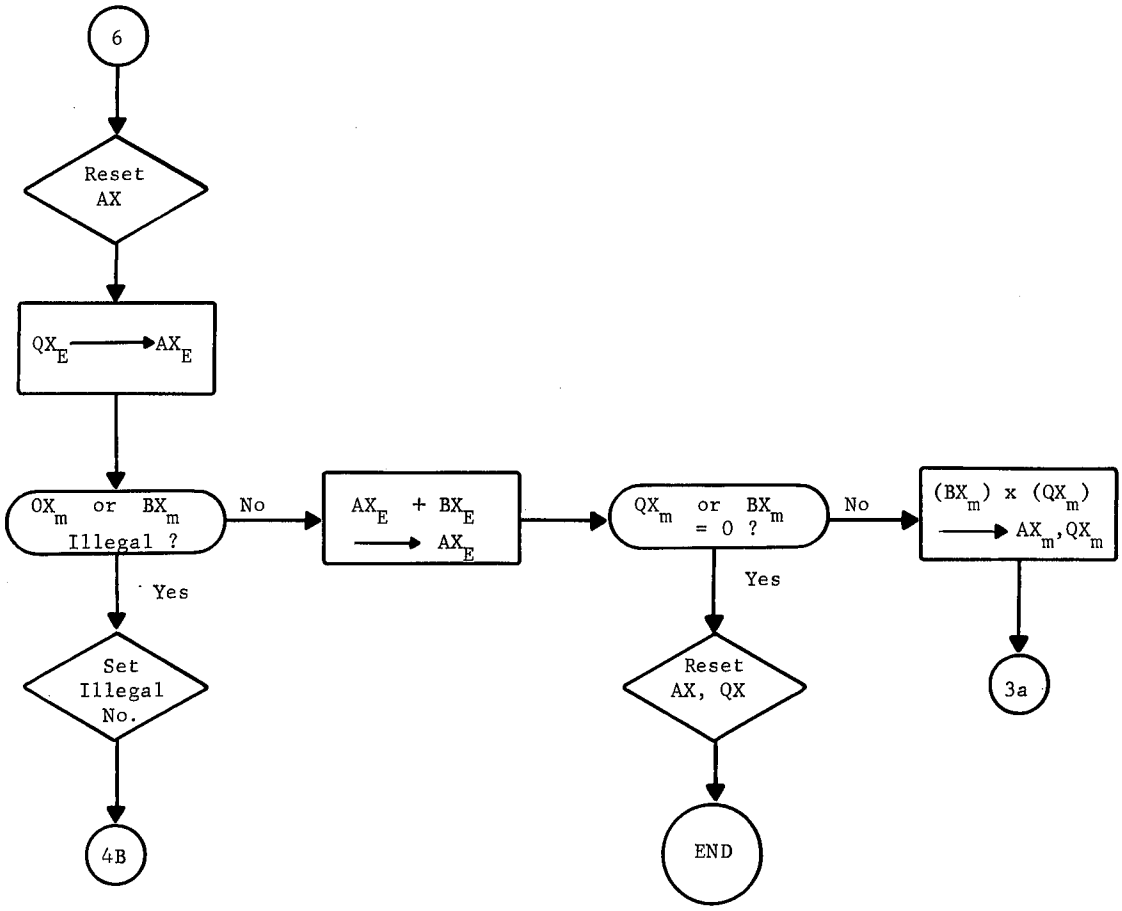


Figure 13. Floating-Point FMP

DIVIDE

FDV M X	36MMMMM

The contents of AX and QX are divided algebraically by the contents of M and M+1. The contents of M and M+1 are unchanged.

Fixed-Point Mode

- M and M+1 are loaded into the BX-Register
- If the AX-Register is negative (AX-0 determines the sign of the entire number in AX- and QX-Registers), the AX- and QX-Registers are complemented
- A check is made to determine if either the AX-, QX-, or BX-Registers contain illegal numbers. If any of them do, the arithmetic operation is terminated with an UNDERFLOW (and illegal number) indicator set in the AAU. The original dividend is left in the AX- and QX-Registers (or the 2's complement if the original dividend was negative)

- If the registers contain legal numbers the BX-Register is checked to determine if it contains zeros (this is the divisor). If the divisor in BX is zero, a divide check error condition exists and the OVERFLOW and DIVIDE CHECK indicator will be set. If the subsystem is in the trapping mode the divide check error condition will not set the OVERFLOW indicator, but it will set the DIVIDE CHECK indicator.

- A check is made to determine if the absolute value of that part of the dividend in the AX-Register is equal to or greater than the absolute value of the divisor. If true, the instruction is terminated with a divide check error and the OVERFLOW and DIVIDE CHECK indicators are set. The original dividend is in the AX- and QX-Registers (or the 2's complement if the original was negative)

- A check is made to determine the zero content of the AX- and QX-Registers. If they are zero, the instruction is terminated

- The contents of the AX- and QX-Registers are divided algebraically by the contents of the BX-Register. The quotient is stored in the AX-Register, with the remainder in the QX-Register and the sign bits set to the sign of the original dividend

- The QX-Register is tested for an illegal number. The only way that QX can contain an illegal number is when the remainder is zero and the sign of the original dividend is negative. If it is found to be illegal, the QX-Register is cleared

- The AX-Register is tested for an illegal number. The only way that AX can contain an illegal number is when the quotient is zero and the sign negative. If it is found to be illegal, the AX-Register is cleared

- Overflow is set if a divide check error is detected during a divide operation. The AX- and QX-Registers are not cleared

- Underflow is set if an illegal number is detected in either the AX-, QX-, or BX-Registers.

Normalized Floating-Point Mode

- The floating-point number in M and M+1 is loaded into the BX-Register

- If the dividend is negative--AX-20 determines the sign of the dividend--the mantissas of AX and QX are complemented

- The exponent of the remainder is formed by subtracting 30 from the exponent in AX. AX_E-30 to QX_E. If the exponent underflows, it is left in QX_E as a positive number (-255 -30 = +29) and underflow is not set

- The mantissas of the AX-, QX-, and BX-Registers are checked for illegal numbers. If there is an illegal number present, the operation is terminated with an UNDERFLOW (an illegal number) indicator in the AAU

- The exponent of the quotient is formed by algebraically subtracting the divisor exponent from the dividend exponent

- A check is made to determine if the mantissa in the BX-Register contains zeros. If so, the OVERFLOW and DIVIDE CHECK indicators are set in the AAU

- A check is made to determine if the absolute value of the dividend mantissa is equal to or greater than the absolute value of the divisor mantissa. If it is, the mantissas of AX and QX are shifted right one place and the exponents are increased by one

- Another check is made to determine if the absolute value of the dividend mantissa is equal to or greater and if it is the OVERFLOW and DIVIDE CHECK indicators are set in the AAU. If they are not equal to or greater, the mantissas of AX and QX are checked for zero

- If both AX- and QX-Registers contain zero, AX and QX are cleared and the arithmetic operation is terminated

- If the mantissas of AX and QX are nonzero, the contents (mantissas) are divided algebraically by the mantissa of BX. The quotient is stored in AX and the remainder is stored in QX with the sign of the original dividend

- The mantissa in AX is normalized

- The mantissa of QX is tested for illegal number or zero content. If it is found true, the AX-Register is cleared

- Overflow is set:

  1. If the exponent of the quotient in the AX-Register exceeds $+255_{10}$.

  2. If a divide check error is detected at the start of the operation and the AAU is in the non-trapping mode. Overflow is not set if the AAU is in the trapping mode.

- Underflow is set:

  1. If an illegal number is detected in the AX-, QX-, or BX-Registers

  2. If the resultant AX exponent is less than +256, the AX- and QX-Registers are cleared

Unnormalized Floating-Point Mode. Same conditions prevail as set forth for normalized floating-point mode, except:

Quotient in the AX-Register is left in unnormalized form.

Figure 14 illustrates the execution of the FDV instruction during a fixed-point mode of operation.

Figures 15, 16 and 17 illustrate the execution of the FDV instruction during a floating-point mode of operation.

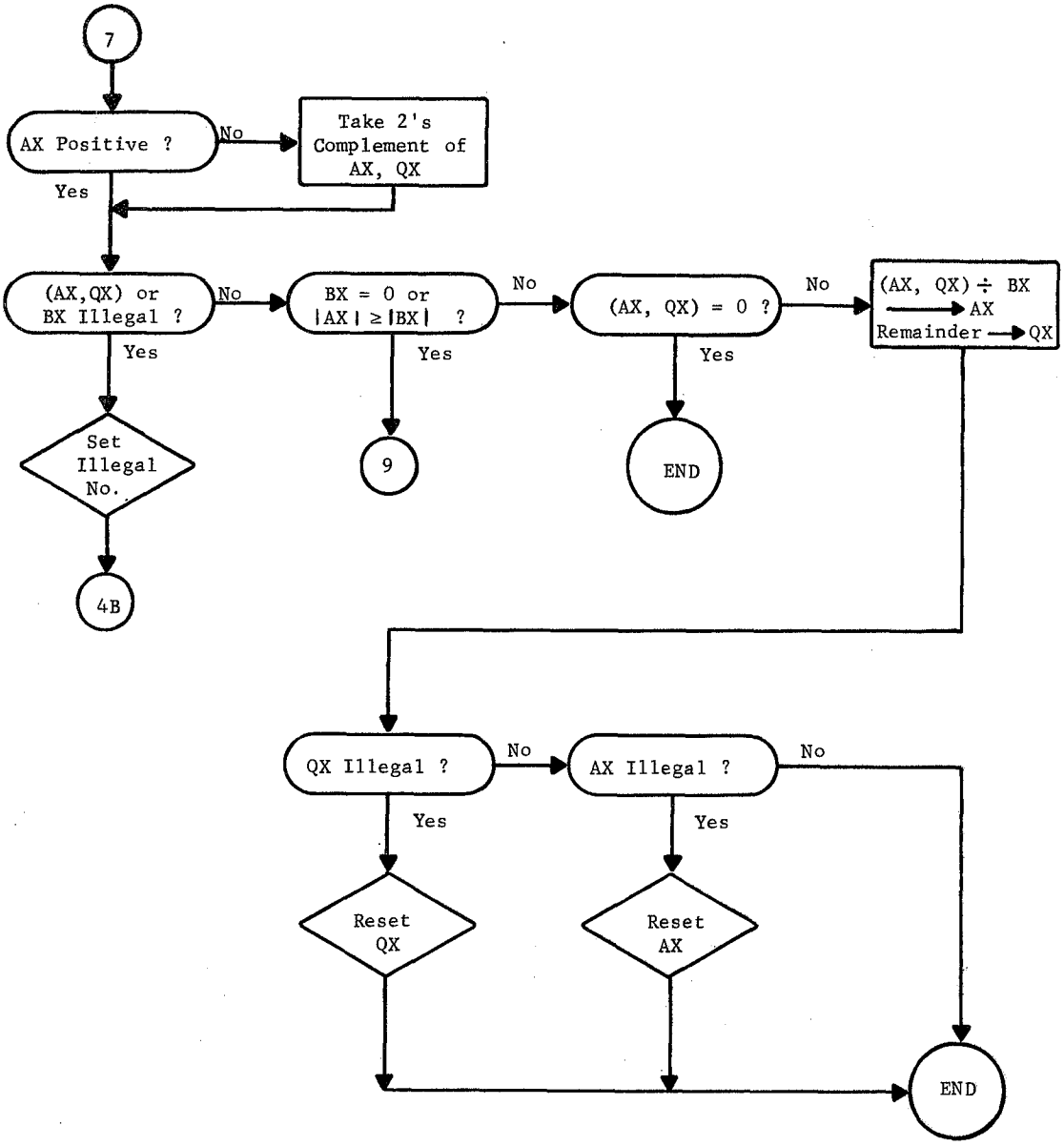


Figure 14. Fixed-Point FDV

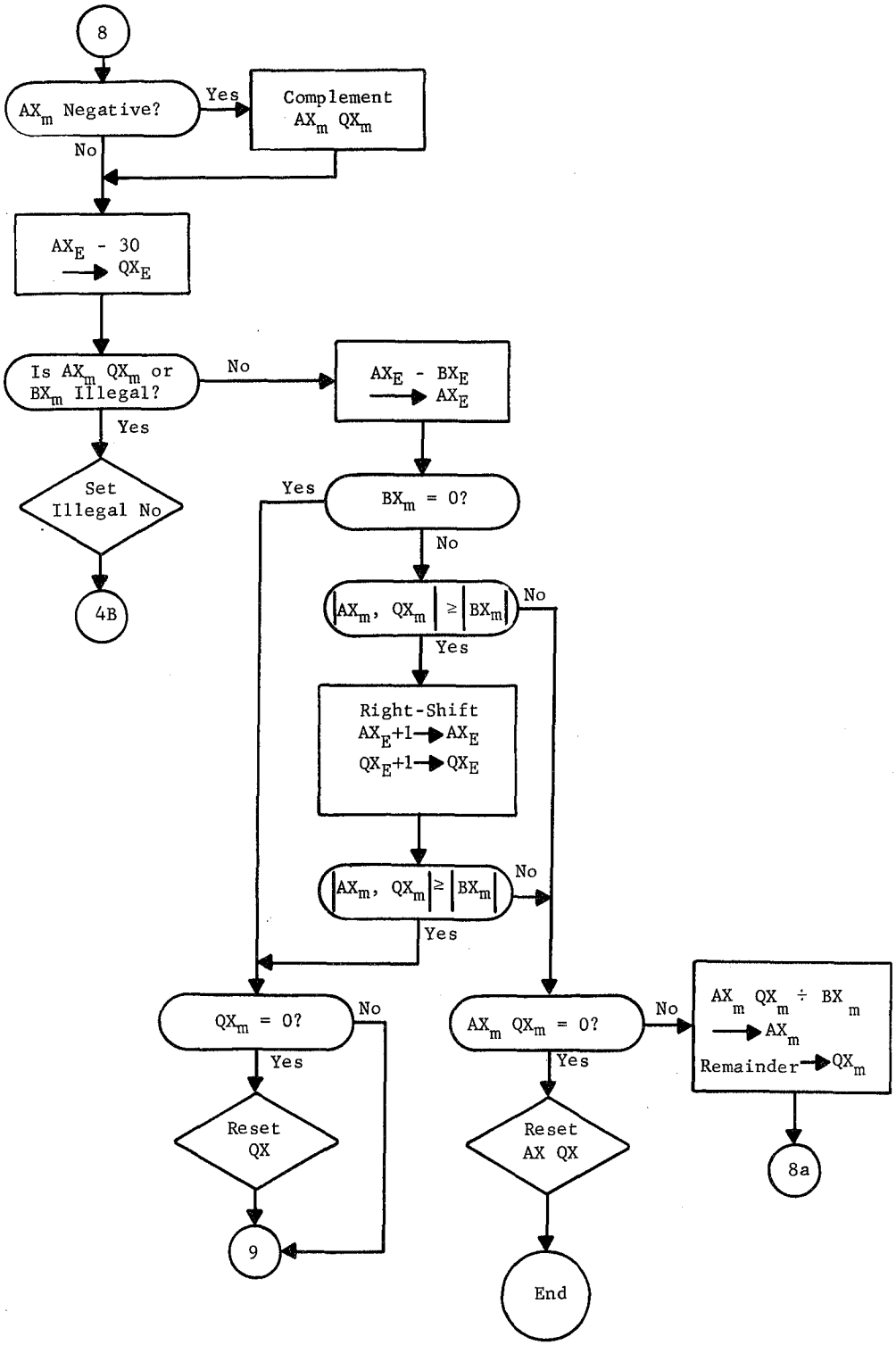


Figure 15. Floating-Point FDV

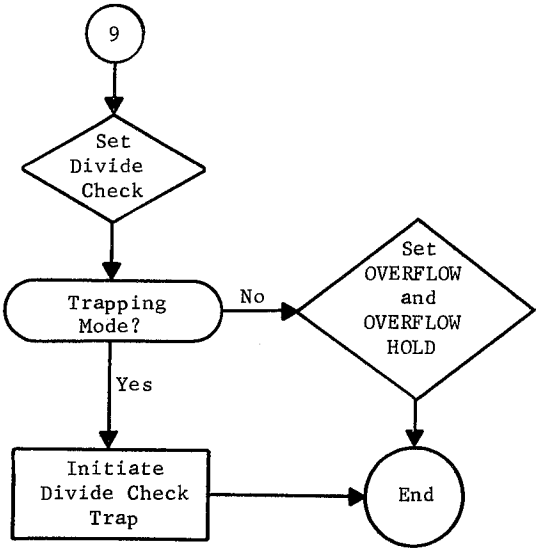


Figure 16. Setting Divide Check

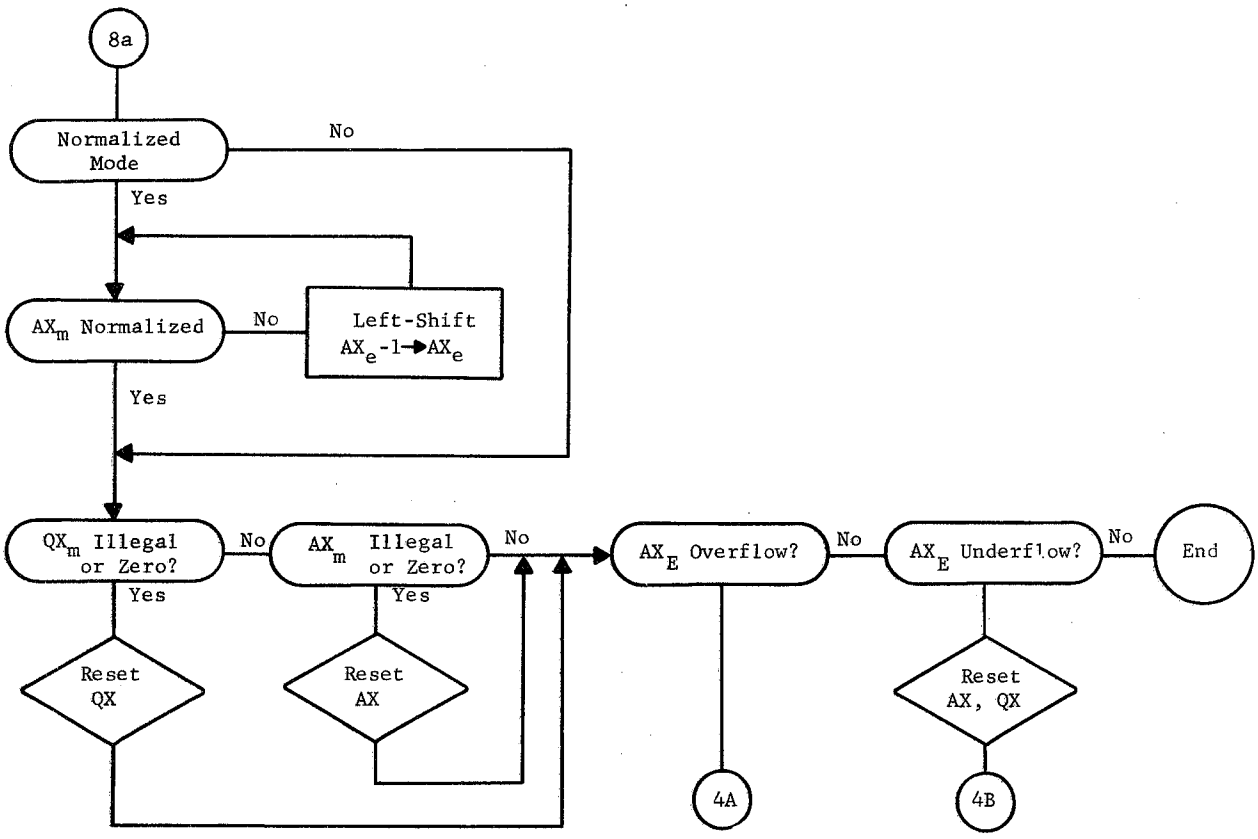


Figure 17. Normalization Floating-Point FDV

DATA TRANSFER INSTRUCTIONS are those which transfer data between the AAU and central processor. The instruction must specify a memory location address greater than 15 unless it is index modified. The instructions for data transfer are FLD and FST. They are not dependent on an operating mode.

LOAD AX REGISTER

FLD	M	X	30MMMMM

- Contents of M and M+1 replace contents of AX-Register.
- Contents of M (even) are loaded into AX 0-19
- Contents of M+1 (odd) are loaded into AX 20-39
- Contents of M and M+1 are unchanged
- Contents of M (odd) are loaded into AX 0-19 and 20-39.
- Illegal numbers may be loaded. Only when an attempt is made to use the illegal number as an arithmetic operand is the illegal condition set.

STORE AX REGISTER

FST	M	X	33MMMMM

- Contents of AX-Register replace contents of M and M+1 if M is even
- Contents of AX-Register are not changed
- Contents of AX-Register 0-19 are stored in M if M is odd and then the contents of AX 20-39 are written over 0-19 in the same location M.

Test-and-Branch Instructions

All AAU test-and-branch instructions are indicated by BAR in the Operation field followed by a 3-letter mnemonic code in the Operand field of the General Assembly Program coding sheet. Test-and-branch instructions interrogate the AAU for specific conditions which, if true, cause the next sequential instruction to be executed. If false, the second sequential instruction is executed. A 7 must be shown in the X field of the coding sheet.

Each test-and-branch instruction is described below.

BRANCH ON AAU READY

BAR	BAR	7	2514720

The AAU is tested to determine if it is ready to receive an instruction.

BRANCH ON AAU NOT READY

BAR	BAN	7	2516720

The AAU is tested to determine if it is not ready to accept an instruction.

BRANCH ON AAU MINUS

BAR	BMI	7	2514721

The AX-Register is tested for a minus sign in bit position 0, if in the fixed-point mode or for a minus sign in bit position 20, if in the floating-point mode.

BRANCH ON AAU PLUS

BAR	BPL	7	2516721

The AX-Register is tested for a plus sign in bit position 20 (sign of the mantissa) if in the floating-point mode or for a plus sign in bit position 0, if in the fixed-point mode.

BRANCH ON AAU ZERO

BAR	BZE	7	2514722

The AX-Register is tested for total zero content.

BRANCH ON AAU NOT ZERO

BAR	BNZ	7	2516722

The AX-Register is tested for nonzero content.

BRANCH ON OVERFLOW

BAR	BOV	7	2514723

The OVERFLOW indicator is tested for the on condition (indicator not reset).

BRANCH ON NO OVERFLOW

BAR	BNO	7	2516723

The OVERFLOW indicator is tested for the off condition (indicator not reset).

NOTE: These are not tests for the OVERFLOW HOLD indicator.

BRANCH ON UNDERFLOW

BAR	BUF	7	2514724

The UNDERFLOW indicator is tested for an on condition (indicator is not reset).

BRANCH ON NO UNDERFLOW

BAR	BNU	7	2516724

The UNDERFLOW indicator is tested for an off condition (indicator is not reset).

NOTE: These are not tests for the UNDERFLOW HOLD indicator.

BRANCH ON OVERFLOW HOLD ON

BAR	BOO	7	2514725

The AAU is tested for the OVERFLOW HOLD on. If the indicator is on, it is turned off.

BRANCH ON OVERFLOW HOLD OFF

BAR	BON	7	2516725

The AAU is tested for the OVERFLOW HOLD off. If the indicator is on, it is turned off.

BRANCH ON UNDERFLOW HOLD ON

BAR	BUO	7	2514726

The AAU is tested for the UNDERFLOW HOLD on. If the indicator is on, it is turned off.

BRANCH ON UNDERFLOW HOLD OFF

BAR	BUN	7	2516726

The AAU is tested for the UNDERFLOW HOLD off. If the indicator is on, it is turned off.

BRANCH ON ERROR				BRANCH ON NO ERROR			
BAR	BER	7	2514727	BAR	BNE	7	2516727
The AAU is tested for either the OVERFLOW, UNDERFLOW, or DIVIDE CHECK indicators on (indicators are not reset).				The AAU is tested for either the OVERFLOW, UNDERFLOW, or DIVIDE CHECK indicators off (indicators are not reset).			

NOTE: These are not tests for the OVERFLOW HOLD or UNDERFLOW HOLD indicators. The divide check tested by this command is an internal "instruction divide check." The program accessible DIVIDE CHECK is a "hold" type indicator, whereas the internal divide check is not.

BRANCH ON DIVIDE CHECK ON				BRANCH ON DIVIDE CHECK OFF			
BAR	BDC	7	2514730	BAR	BDN	7	2516730
The DIVIDE CHECK indicator is tested for an on condition. If the indicator is on, it is turned off.				The DIVIDE CHECK indicator is tested for an off condition. If the indicator is on, it is turned off.			

BRANCH ON ILLEGAL NUMBER				BRANCH ON NO ILLEGAL NUMBER			
BAR	BIL	7	2514731	BAR	BNI	7	2516731
The AAU illegal number condition is tested for an on condition.				The AAU illegal number condition is tested for an off condition.			

NOTE: If the illegal number is on, it is turned off by either of these instructions.

BRANCH ON FIXED-POINT MODE ON				BRANCH ON FIXED-POINT MODE OFF			
BAR	BFX	7	2514732	BAR	BNX	7	2516732
The FIXED-POINT indicator is tested for an on condition.				The FIXED-POINT indicator is tested for an off condition.			

NOTE: If the indicator is on, it is not reset by either of these instructions.

BRANCH ON UNNORMALIZED FLOATING-POINT ON				BRANCH ON UNNORMALIZED FLOATING-POINT OFF			
BAR	BUP	7	2514733	BAR	BNP	7	2516733
The FLOATING-POINT UN-NORM indicator is tested for an on condition.				The FLOATING-POINT UN-NORM indicator is tested for an off condition.			

NOTE: If the indicator is on, it is not reset by either of these instructions.

BRANCH ON NORMALIZED FLOATING FLOATING-POINT ON				BRANCH ON NORMALIZED FLOATING-POINT OFF			
BAR	BNF	7	2514734	BAR	BNN	7	2516734
The FLOATING-POINT NORMAL indicator is tested for an on condition.				The FLOATING-POINT NORMAL indicator is tested for an off condition.			

NOTE: If the indicator is on, it is <u>not</u> reset by either of these instructions.

PROGRAM CONSIDERATIONS

Unlike the requirements for peripherals with controllers having access to the central processor through its priority control, the AAU is always connected to the central processor and only a SET instruction is required to select the mode of operation.

Setting the Calculation Mode

Once a SET MODE instruction is given, it need not be given again until the programmer wants to change modes. The instructions required to perform the desired arithmetic operations follow the mode setting instructions. To illustrate, Figure 18 shows a program which begins calculations in the normalized floating-point non-trapping mode and then switches to the unnormalized floating-point non-trapping mode as directed by the new SET instruction shown on line 12.

General Assembly Program Coding:

	Symbol	Opr	Operand	X	REMARKS
	1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31
1	START	SET	NFLPOINT		Set normalized floating-point mode
2		FLD	DR		
3		FSU	CR		
4		FST	TFM 1		
5					
6					
7					
8					
9					
10					
11		FST	ADIDO		
12		SET	UFLPOINT		Set unnormalized floating-point mode
13		FLD	GA		
14					
15					
16					

Figure 18. Setting Mode For Calculation

Setting the Trapping Mode

The trapping mode, a standard feature of the GE-235 AAU, operates in a manner similar to the Automatic Program Interrupt (API). Automatic Program Interrupt is explained in the GE-235 Central Processor Reference Manual (CPB-374).

API is not affected by the trapping mode. If a peripheral device terminates on exactly the same memory cycle in which a trap occurs, the program interrupt (API) will be taken first. The AAU trap signal is not dropped until the trap is executed, but the trap is not executed if the central processor is executing the priority program. The first instruction in sequence--after program control is returned to the main program (from API mode)--will not be executed, the AAU trap will be executed.

If a peripheral device should become ready while the central processor is executing the trap routine, control is transferred to the API routine and executed. Upon completion of the API routine, control is transferred back to the trap routine.

An API routine should not use the trap mode. If it does, the trap interrupt does not take place. It is remembered however, and the trap interrupt is executed after control is returned to the central processor from the API routine.

When the trapping mode is entered by a SET TRPMODE instruction, operation is as follows:

- The OVERFLOW HOLD, UNDERFLOW HOLD and DIVIDE CHECK indicators (an illegal number sets the UNDERFLOW indicator) are neither reset nor disabled
- If the programmer desires to test these indicators while in the trapping mode, he should issue a reset indicators (RIN) instruction at the same time that the SET TRPMODE is given. If any of the indicators are on at the time that the SET TRPMODE instruction is given, the AAU will not take trapping action
- Trapping action is only taken on subsequent AAU arithmetic operations
- Interrogation of any of error indicators causes them to be reset, the reset feature operates the same in both trap and non-trap modes
- At the completion of each AAU arithmetic instruction, the AAU checks itself to see if that instruction resulted in an overflow, underflow, or divide check condition:

  1. If the instruction results in an overflow, the contents of the P-Register (address of the next instruction) are stored in location 211_8 and an unconditional branch to location 205_8 is executed.
  2. If the instruction results in an underflow, the contents of the P-Register is stored in location 211_8, and a branch to location 206_8 is executed.
  3. If the instruction is not executed because of a divide check error, the contents of the P-Register is stored in location 211_8, and a branch to location 207_8 is executed.

     NOTE: When operating in the non-trapping mode, the divide check error will set the OVERFLOW and OVERFLOW HOLD indicators. However, when operating in the trapping mode only the DIVIDE CHECK indicator is set.

4. Bits 0-4 of the word at location 211_8 where the contents of the P-Register are stored has special meanings and are set/reset in accordance with the conditions that exist within the central processor at the time of the interrupt. These conditions are as follows:

Bit	Set if	
0	KON	is in effect
1	TON	is in effect
2	REMEMBER CARRY	is on (central processor)
3	DECMODE	is in effect
4	OVERFLOW	is on (central processor)

These indicators enable the programmer to restore, at the completion of the trap routine, the conditions that existed at the time of the interrupt

- The contents of the G-Register (Index group number) are not disturbed by the trap. The modification group in effect before the trap exit will still be in effect after the trap. An additional index group (34) in locations 210_8-213_8 is provided for use by the trap routine. This index group is established automatically when the central processor accepts the trap-interrupt.

Prior to exiting from a trap, a SET TRPMODE instruction must be issued.

If it is not desired to continue in the trap mode, the SET TRPMODE must be followed by a SET NTPMODE instruction. Additional trap interrupts are ignored and not remembered while executing a trap routine.

The following assembly language coding will cause the indicators to be set for testing during a trapping routine.

Symbol	Opr	Operand	X	REMARKS
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31
	SET	TRPMODE		
	RIN			"Reset Indicators"
	FLD	MP		

The following assembly language coding will cause a divide check error and demonstrates utilization of the trapping mode:

```
START   SETNFLPOINT          Sets Arithmetic Mode
        SETTRPMODE
        FLD     NUM          Load Dividend
        FDV     ZERO         Divide by zero - next instruction from 207(8) trap
NEXT    FST     ANSWER
DIVCHC  XXX     XXXXX        Coding to correct Divide Check Error
          |       |
          v       v
        SETTRPMODE           Must be given prior to return
RETURN  BRU     O        1   Return to "NEXT" - Location 211(8)
```

3. OPERATING THE GE-235 AAU

OPERATING LOGIC

Once the AAU is granted access to memory, two data words are brought from memory through the central processor M-Register and checked for parity. Each data word then enters a buffer register in the AAU where the 40-bit double word format--in the predetermined mode for the AAU--is formed.

The AX , BX , QX , and IX-Registers and the adder--SX--perform functions similar to their counterpart registers--A, B, Q, and I--in the GE-235 central processor.

The GE-235 central processor performs the function of instruction retrieval. An immediate decision point is reached if the retrieved instruction is an AAU instruction; that is, if bits S and 1 are both ones. If the retrieved instruction has either of the two undefined operation codes (34_8 or 37_8), an error condition results.

If the instruction is either an arithmetic or a load/store instruction, any of which may be index word modified by the central processor, then any indexing function required is performed. After possible indexing, the instruction is executed by the AAU using the central processor I-Register as the memory operand address register.

If the instruction is one of the AAU test-and-branch instructions (BAR), the central processor interrogates the various status indicators. The AAU replies to the query, and the central processor forms the appropriate branching decision.

The entire process is synchronous. During the period that the AAU is executing an instruction, the central processor is inhibited from retrieving the next instruction. Each instruction in turn is executed.

AAU CONTROLS AND INDICATORS

The GE-235 AAU operator panel is composed of register displays, mode and alarm indicators and operating switches. A detailed description follows Figure 19, showing the operator panel of the AAU.

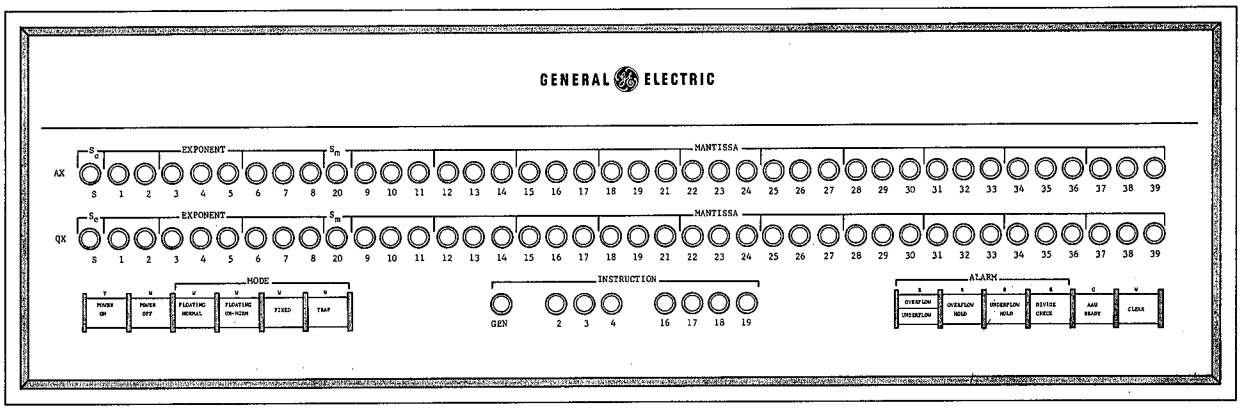


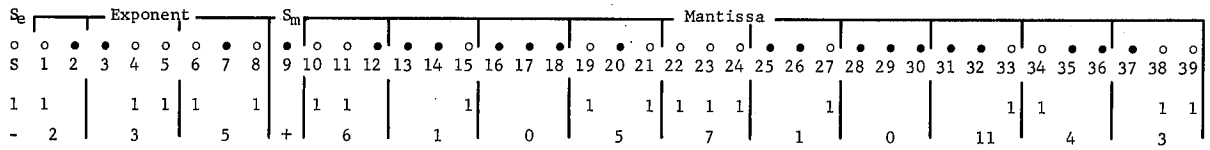
Figure 19. GE-235 AAU Control and Indicator Panel

AX- and QX-Register Indicators

The operator can see the contents of the AX- and QX-Registers by the display lights along the top of the panel. The light for the sign of the mantissa in both the AX- and QX-Registers is labeled S_m and is placed before the mantissa group rather than in bit position 20. This is done to present a more meaningful picture of the floating-point word and to expedite reading the contents of the registers in floating-point modes. In the fixed-point mode, where the grouping of bits into mantissa and exponent has no meaning, the sign of the mantissa, bit 20 is ignored.

The AX and QX-Registers can be read visually only during a pause or a halt in the program. The contents of the registers assist the programmer or service engineer in diagnosing trouble in either the program or the equipment. The operator should note all information on the panel at the time of an unprogrammed halt. The sign is negative when the light is on and positive when the light is off. Line divisions between the register lights assist in reading the register contents in octal numbers. The following example illustrates reading one of these registers. The indicators can be converted to octal and written in octal.

Example:



This information would be written as $(-235+610\ 571\ 014\ 3)_8$.

POWER ON - Switch

This switch turns the power on in the AAU when the switch is pressed. It glows yellow when power comes on.

POWER OFF - Switch

When this switch is pressed the POWER ON switch lights goes out and power is turned off for the AAU.

MODE - Indicators

There are four mode indicator lights, one for each of the four operating modes. When the program specifies the mode, the applicable mode indicator is lit.

When the program makes an unscheduled halt, the operator should note which of the four indicators is lit.

FLOATING NORMAL. When illuminated it indicates the AAU is operating in normalized floating-point mode.

FLOATING UN-NORM. When illuminated it indicates the AAU is operating in unnormalized floating-point mode.

FIXED. When illuminated it indicates the AAU is operating in fixed-point mode.

TRAP. When illuminated it indicates the AAU is operating in trapping mode.

INSTRUCTION - Indicators

The display lights of the instruction register show the contents of the IX-Register for bits 2, 3, and 4 (of the operation code) and bits 16, 17, 18, and 19 (of the operand).

The entire instruction is sent from the central processor to the AAU, however, since all general instructions for AAU operations are code 3 (bits 0-1) and bits 5-15 are all zeros, only those bits affecting AAU operation are displayed.

ALARM - Indicators

There are four alarm indicators which indicate to the operator any unusual conditions that have been encountered as a result of an arithmetic operation.

OVERFLOW/UNDERFLOW is a split indicator and is illuminated in its area for the conditions as shown on the next page.

Overflow.

- In the fixed-point mode is caused by a positive sum requiring in excess of 38 significant bits
- In the floating-point mode is caused by a positive exponent requiring in excess of 8 significant bits
- Is caused upon the detection of a divide check error condition provided the AAU is operating in the non trapping mode. It is not set if the AAU is operating in the trapping mode.

Underflow.

- In the fixed-point mode is caused by a number whose decimal value is -2^{38} or less
- In the floating-point mode is caused by an exponent whose decimal value is -256 or less

Both Alarms Are Reset by:

- The execution of any AAU instruction, except a test and branch (BAR) instruction
- The depression of the CLEAR switch on the AAU
- The depression of the RESET ALARM switch on the central processor.

OVERFLOW HOLD is turned on in the same manner as the OVERFLOW. The OVERFLOW HOLD alarm will be set on a detection of overflow regardless of the trap mode setting.

Alarm is Reset by:

- Either of the overflow hold test (BAR-BOO or BAR-BON) instructions
- The overflow hold reset (ROV) instruction
- The reset indicator (RIN) instruction
- Depressing the CLEAR switch on the AAU
- Depressing the RESET ALARM switch on the central processor.

UNDERFLOW HOLD is turned on in the same manner as the UNDERFLOW. The UNDERFLOW HOLD alarm is also set upon a detection of any illegal number specified as an operand. It also is set regardless of the trap mode setting.

Alarm is Reset by:

- Either of the underflow hold test (BAR-BUO or BAR-BUN) instructions.
- The underflow hold reset (RUN) instruction

- The reset indicator (RIN) instruction
- Depressing the CLEAR switch on the AAU
- Depressing the RESET ALARM switch on the central processor

DIVIDE CHECK is turned on if an attempt is made to divide any number by zero.

- In the fixed-point mode, it is turned on if the magnitude of that part of the dividend in the AX-Register is equal to or greater than the magnitude of the divisor
- In the floating-point mode it is turned on if the magnitude of the dividend mantissa is equal to or greater than twice the magnitude of the divisor mantissa
- The alarm is set regardless of the trap mode setting.

Alarm is Reset by:

- The reset indicator (RIN) instruction
- Either of the divide check test (BAR-BDC or BAR-BDN) instructions
- Depressing the CLEAR switch on the AAU
- Depressing the RESET ALARM switch on the central processor

AAU READY - Indicator

This indicator is illuminated whenever the AAU is ready to execute an instruction. It means that the AAU power is on and that the AAU is not in a test mode.

CLEAR - Switch

This switch clears and resets the following:

- The OVERFLOW alarm
- The OVERFLOW HOLD alarm
- The UNDERFLOW alarm
- The UNDERFLOW HOLD alarm
- The DIVIDE CHECK alarm
- The illegal number condition (UNDERFLOW)
- The AX-Register to zero
- The QX-Register to zero
- The mode of operation is switched to normalized floating-point mode
- The AAU is set to the non-trapping condition

CENTRAL PROCESSOR AAU INDICATORS

There are 8 AAU indicators placed above the operator's panel of the central processor. They are for the convenience of the operator who, due to the arrangement of the equipment, might not be able to see the operator's panel on the AAU. The indicators show the same condition as the indicators on the AAU. Figure 20 shows the relationship of the AAU indicators and the control panel of the central processor.

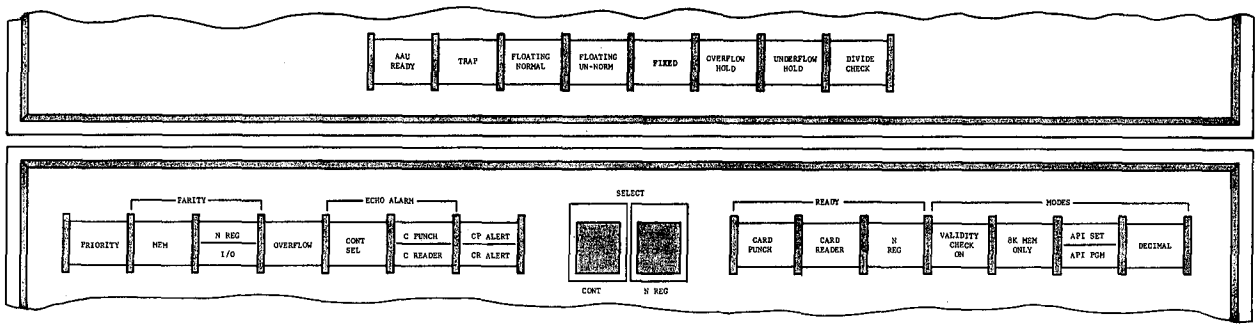


Figure 20. Central Processor AAU Indicators

APPENDIX A

AAU PROGRAM LIBRARY ROUTINES

Internal Data Routines

Library Number

Fixed-Point BCD to Binary--Provides binary-coded-decimal conversion to binary for AAU operations. — CD225C1.002

Floating Point Binary to BCD--Provides floating-point binary to floating-point BCD conversion with rounding. — CD225C2.006

Floating-Point Binary to BCD--Provides rapid floating-point binary to floating-point BCD conversion, utilizing the decimal (BCD) arithmetic option — CD225C2.008

Math Routines

Bessel Functions of Orders 0 and 1--The purpose of these subroutines is to evaluate Bessel functions of types I, J, K, and Y for orders 0 and 1 for specified ranges of the argument, as follows: — CD225D5.004

--- Bessel Function Type	Range of the Argument
I_0, I	$0 \leq X < \infty$
J_0, J	$0 \leq X < \infty$
K_0, K_1	$0 < X < \infty$
Y_0, Y_1	$0 < X < \infty$

Complex Arithmetic with Trace Option--Provides a means of handling complex floating-point numbers by means of a package of subroutines. — CD225D1.004

Fixed-Point - ARCTANGENT--Computes the arctangent in radians of a fixed-point number. The binary point is considered to be at 8; thus the range is between -255 and +255. — CD225D2.024

Fixed Point - EXPONENTIAL--Computes α^x where α is 2, e, or 10, and x is the independent variable in fixed-point form at a binary point of 8. — CD225D2.028

Fixed-Point - LOG (10, 2, e)--Computes log α^x where α is 2, e, or 10, and x is the independent variable. The input and output are fixed-point numbers at a binary point of 8. — CD225D2.026

Fixed Point - SIN COS--Computes the sine or cosine of a fixed-point argument in radians between -256 and +256. CD225D2.020

Fixed-Point - SQUARE ROOT--Computes the square root of a positive fixed point number. The binary point is considered to be at the far left or at zero. This routine takes the square root of any positive 38-bit number. CD225D2.022

Floating-Point - ARCTANGENT--Computes the arctangent in radians of a normalized floating-point number, in the range of -2^{255} to $+2^{255}$. CD225D2.010

Floating-Point - EXPONENTIAL--Computes α^x where α is 2, e, or 10, and x is the independent variable in normalized floating-point form. CD225D2.012

Floating-Point - LOG (10, 2, e)--Computes $\log \alpha^x$ where α is 2, e, or 10, and x is the independent variable in normalized floating-point form. CD225D2.014

Floating-Point - SINE/COSINE--Computes the sine or cosine of a normalized floating-point argument in radians. CD225D2.008

Floating-Point - SQUARE ROOT--Computes the square root of a normalized floating-point number, in the range of 0 to 2^{255}. CD225D2.006

GAMMA Function--Evaluates the GAMMA function for all real arguments except those near integral negative values where the function goes to infinity. CD225D5.006

Least Squares POLYNOMIAL CURVE FIT--Fits polynomials up to tenth order through a points (X_1, Y_1), (X_2, Y_2), ... (X_n, Y_n) by the method of least squares, and to calculate residuals and total variance to facilitate choosing the best fit. CD225D6.002

Least Squares POLYNOMIAL CURVE FIT Program--Fits polynomials up to tenth order through a maximum of 400 points (X_i, Y_i). Standard error, variance, and residuals are provided. CD225D6.004

LINEAR Programming--Finds the optimum solution of a group of restrictive linear equations. The solution permits efficient allocation of limited resources to meet desired objectives. CD225D7.002

LINEAR Simultaneous Equations--Solves a set of n linear real simultaneous equations in n unknowns whose coefficients are represented in normalized floating-point form and to store the results in the same form. CD225D4.012

MATRIX Transpose--Finds the transpose of a real matrix A(m,n) whose elements are double words and to store the results in the same form in matrix B(n,m). CD225D4.006

Multiple LINEAR Regression--Calculates entirely in floating-point binary, the representation having a significant part of 30 binary positions and sign. The accuracy is just beyond that of 9 decimal digits. CD225D3.002

Normalized Floating-Point Matrix Inversion--Computes the inverse of an N x N matrix, A, whose elements are real and represented in normalized floating-point form, and to store the result, B, in the same form. CD225D4.010

Normalized Floating-Point Matrix Multiply--Multiplies a normalized floating-point real scalar s by a real matrix A(m,n) whose elements are represented in normalized floating-point form, and to store the resultant matrix B(m,n) in the same form. CD225D4.004 CD225D4.004

Normalized Floating-Point Matrix Add or Subtract--Computes the sum or difference of two m x n real matrices, A B, whose elements are represented in normalized floating-point form and to store the result C in the same form. CD225D4.002

Normalized Floating-Point Scalar Multiply--Multiplies a normalized floating-point real scalar s by a real matrix A(m,n) whose elements are represented in normalized floating-point form and to store the resultant matrix B(m,n) in the same form. CD225D4.008

Roots of POLYNOMIAL--Calculates all n roots of the polynomial. CD225D5.002

Simulated Floating-Point--Simulates the AAU in its floating-point mode as may be practical with software. CD225D1.000

Input/Output Routines

Floating Field Input--Reads cards and convert to binary the data punched in any convenient columns of the cards. The data may be fixed- or floating-point, decimal, octal or alphanumeric. CD225E1.004

General Purpose Output Program--Sets up information for output to the on-line punch and/or to the on-line printer. Output is buffered. CD225E1.006

Packed Data Reading Program--Reads fixed-format decimal cards and to convert the fields into BCD, floating-point or fixed-point. Also to check, at the user's option, a field of (at most) 18 characters on each card in order to verify that the card does indeed belong to the correct deck. CD225E1.008

Compilers/Translators

WIZ Compiler Mod 1--Algebraic compiler CD225H2.002

WIZ Two Compiler--Algebraic compiler CD225H2.004

Assembly Systems

ZOOM--A Macro Assembler CD225F1.002

APPENDIX B

LIST OF INSTRUCTIONS

The abbreviations used in the mnemonics of the Operand and Symbol X field of the General Assembly Program (GAP) are as follows:

M Memory Location
7 Channel Number (Priority Control)
A AAU Transfer Instruction
X Address Modification

Opr	Operand	X	Octal	Description	Time In Microseconds
BAR	BAN	7	2516720	Branch on AAU not ready	12
BAR	BAR	7	2514720	Branch on AAU ready	12
BAR	BER	7	2514727	Branch on error	12
BAR	BDC	7	2514730	Branch on DIVIDE CHECK on	12
BAR	BDN	7	2516730	Branch on DIVIDE CHECK off	12
BAR	BIL	7	2514731	Branch on illegal number	12
BAR	BFX	7	2514732	Branch on fixed-point mode on	12
BAR	BMI	7	2514721	Branch on AAU minus	12
BAR	BNE	7	2516727	Branch on no error	12
BAR	BNF	7	2514734	Branch on normalized floating-point on	12
BAR	BNI	7	2516731	Branch on no illegal number	12
BAR	BNN	7	2516734	Branch on normalized floating-point off	12
BAR	BNO	7	2516723	Branch on no overflow	12
BAR	BNP	7	2516733	Branch on unnormalized float-point off	12
BAR	BNU	7	2516724	Branch on no underflow	12
BAR	BNX	7	2516732	Branch on fixed-point mode off	12
BAR	BNZ	7	2516722	Branch on AAU not zero	12

Opr	Operand	X	Octal	Description		Time In Microseconds
BAR	BON	7	2516725	Branch on OVERFLOW HOLD off		12
BAR	BOO	7	2514725	Branch on OVERFLOW HOLD on		12
BAR	BOV	7	2514723	Branch on no overflow		12
BAR	BPL	7	2516721	Branch on AAU plus		12
BAR	BUF	7	2514724	Branch on underflow		12
BAR	BUN	7	2516726	Branch on UNDERFLOW HOLD off		12
BAR	BUO	7	2514726	Branch on UNDERFLOW HOLD on		12
BAR	BUP	7	2514733	Branch on unnormalized float-point on		12
BAR	BZE	7	2514722	Branch on AAU zero		12
CAX		A	3200005	Clear AX-Register		6
CQX		A	3500005	Clear QX-Register		6
FAD	M	X	31MMMMM	Add	F N U	18 24-30 24-30
FDV	M	X	36MMMMM	Divide	F N U	96 66-78 66-72
FLD	M	X	30MMMMM	Load AX-Register		18
FMP	M	X	35MMMMM	Multiply	F N U	30-60 24-48 24-48
FST	M	X	33MMMMM	Store AX-Register		18
FSU	M	X	32MMMMM	Subtract	F N U	18 24-30 24-30
LAQ		A	3600002	Load AX from QX		6
LQA		A	3200002	Load QX from AX		6
MAQ		A	3100002	Move AX to QX		6
NOX			3100005	Normalize AX and QX		12
RIN			3500004	Reset indicators		6
ROV			3100004	Reset OVERFLOW HOLD		6

Opr	Operand	X	Octal	Description	Time In Microseconds
RUN			3200004	Reset UNDERFLOW HOLD	6
SET	FIXPOINT		3500010	Set fixed-point mode	6
SET	NFLPOINT		3100010	Set normalized floating-point mode	6
SET	NTPMODE		3200001	Set trapping mode off	6
SET	TRPMODE		3100001	Set trapping mode on	6
SET	UFLPOINT		3200010	Set unnormalized floating-point mode	6
XAQ		A	3500002	Exchange AX and QX	6

APPENDIX C

AAU HALT CONDITIONS

The program will hang-up and the central processor will halt when the central processor is in the AUTO mode and the following conditions occur:

- The PRIORITY alarm light is red on the central processor console

Possible Cause	Corrective Action
A 34XXXXX or 37XXXXX instruction was given.	Return the program with notification to the programmer.

- The PRIORITY alarm light on the central processor console may or may not come on.

Possible Cause	Corrective Action
A program which had an AAU used channel 7 for another peripheral and a SEL instruction called for channel 7.	Return the program with notification to the programmer.