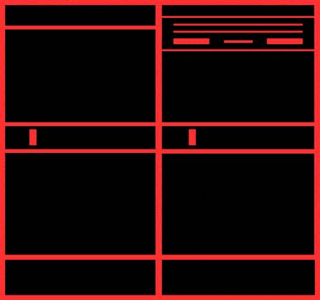
THE COMPATIBLES / 200

# GE-205 / 215 / 225 Auxiliary Arithmetic Unit



GENERAL ELECTRIC

CPB-325A

# GE-205 / 215 / 225
# AUXILIARY ARITHMETIC UNIT

January, 1964

Rev. March, 1965

GENERAL ELECTRIC

COMPUTER DEPARTMENT

## PREFACE

This manual describes the functions and operations of the Auxiliary Arithmetic Unit (AAU). It 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.

The following manuals contain detailed information for the COMPATIBLES/200 systems:

GE-225 System Manual (CPB-244)
GE-225 Programming Reference Manual (CPB-252A)

This publication is a new edition and supersedes the GE-215/225 Auxiliary Arithmetic Unit Reference Manual (CPB-325). Send comments about this publication to General Electric Computer Department, Drawer 270, Phoenix, Arizona, 85001.

Comments on this publication may be addressed to Technical Publications, Computer Department, General Electric Company, P. O. Box 2961, Phoenix, Arizona, 85002.

# CONTENTS

# ILLUSTRATIONS

# 1. GENERAL DESCRIPTION

The Auxiliary Arithmetic Unit (AAU) is one of the many devices designed to aid the GE-205/215/225 Information Processing Systems in the high-speed handling of mass detailed information. Although connected to the central processor through the priority control section, the AAU is not merely another peripheral device. The AAU can be considered as an extension of the central processor with larger registers permitting calculations of greater complexity. It is particularly useful in applications requiring numerous floating-point or double length fixed-point calculations. The central processor of the GE-205/215/225 system continues standard arithmetic operations when the system includes an AAU.

## FUNCTIONS

The central processor of the GE-205/215/225 systems 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 central processor operations
7. Formulation of branch decisions based on AAU status indicators.

## COMPATIBILITY

The basic user-oriented design philosophy of each of the GE-205/215/225 systems is the same, and the systems have proved themselves fast, accurate, reliable and economical in widely divergent fields. As a result of the design similarity of the systems, all programs and applications originally designed for one of the systems can immediately be processed on a member system having the same system configuration.

## CONTROL

All AAU instructions and information (operand words) originate with the central processor. The AAU and the central processor receive instructions simultaneously. An 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 system which includes an AAU can still do standard arithmetic operations in the central processor simultaneously with other computer operations. 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 has to keep track of the fractional point (radix point) by program. This is a lengthy and time-consuming operation; however, the AAU simplifies programming and saves time since the hardware makes a normalized or unnormalized floating-point calculation or consideration.

### Normalized Floating-Point Mode

A normalized number is one in which the most significant nonzero digit of the fraction is immediately to the right of 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 the normalized floating-point is used. 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. The program can be written to align each decimal point for each calculation, but this process is cumbersome. The 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, except 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) |

## REGISTERS

All AAU data words are first placed in the memory of the central processor as two 20-bit words in sequential memory locations. Bit meaning depends 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 those performed by their counterpart registers (A, B, Q, and I) in the central processor. Figure 1 is a block diagram showing the data and control timing paths for the functional logic and the major registers of the 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 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.

### 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.

### SX-Register

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

- Acts as a 40-bit full binary adder during fixed-point operations.

- Acts as a 31-bit full adder for the mantissa during floating-point operations.
- Acts as a 9-bit full adder for the exponent during floating-point operations.
- Adds the carry to the 1's complement on all negate operations.

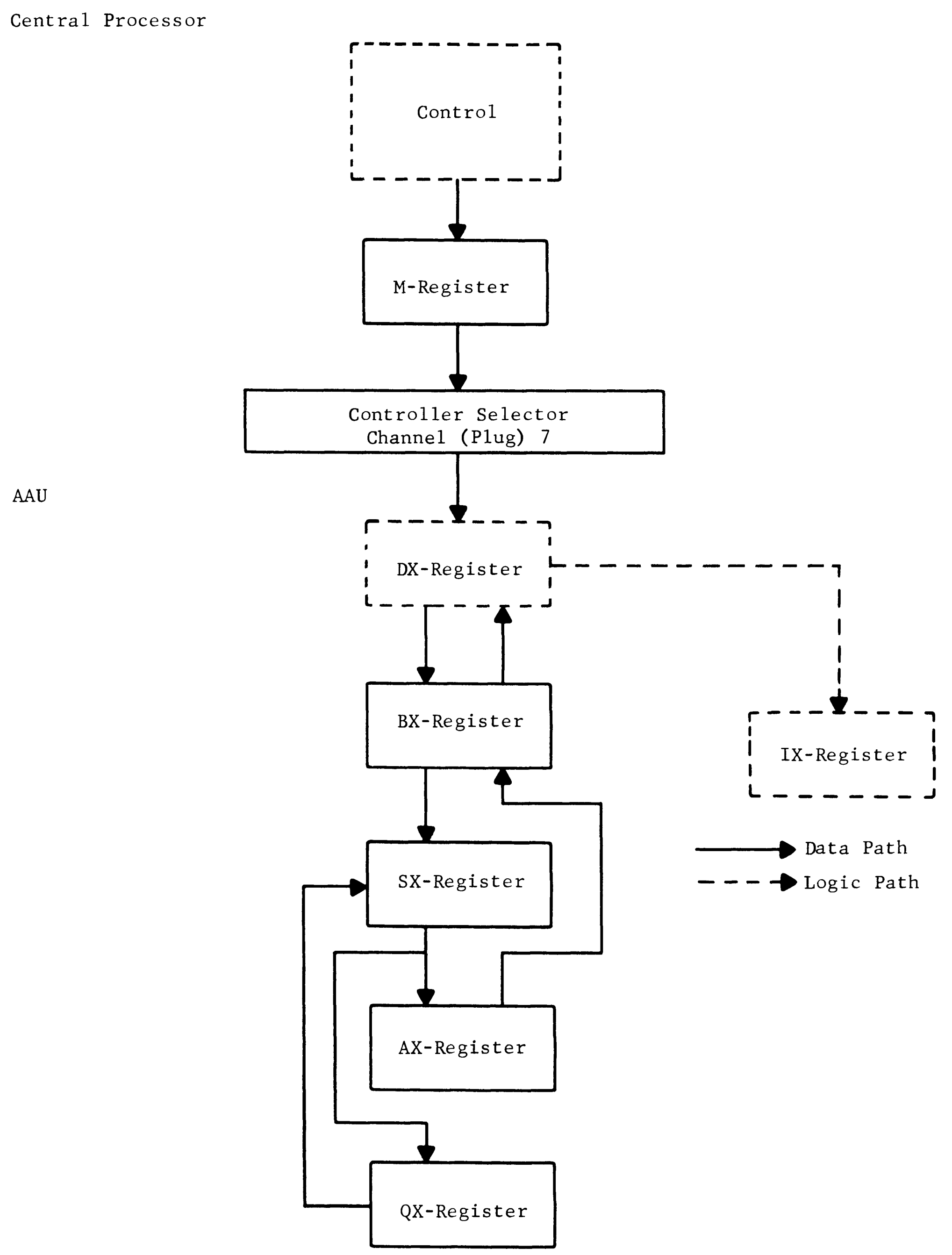


Figure 1. Major Registers and Functional Logic Diagram

## 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, 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, 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 complementing that is required by the AAU.

## 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.

## DX-Register

The DX-register is a buffer between the central processor and the AAU.

# 2. PROGRAMMING

The Auxiliary Arithmetic Unit is a peripheral of the GE-205/215/225 systems. It is connected to the central processor through the controller selector. Address channel (plug) 7 is reserved for the AAU.

## 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 operates independently of the central processor. If an AAU instruction is issued during the AAU operation of an instruction, 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 arithmetic or load/store instructions 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.

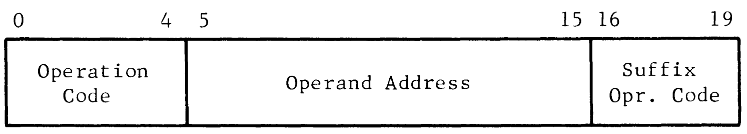
## WORD FORMATS

The formats for instruction and data words for the AAU are compatible within the GE-205/215/225 systems.

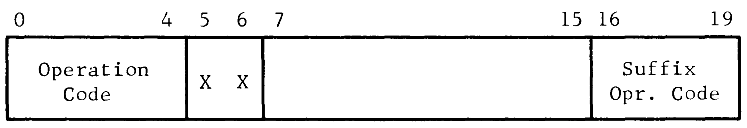
### Instruction Words

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

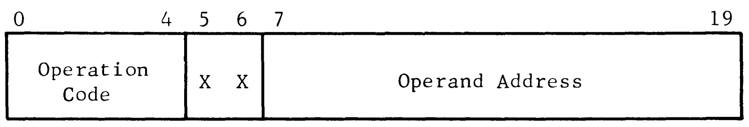
GENERAL INSTRUCTION WORDS



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; the meanings of the words depend on 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. Fixed-point words in memory are in the format shown in Figure 2. 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 2 identical sign bits. Similarly, a fixed-point word in the QX-register also consists of 38 information bits and 2 sign bits.

Bit 1, $S_e$, is the sign bit of the entire word (38 bits), bit 21, $S_m$, 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 3.

$S_e$ = Sign of number (0 = +, 1 = -)

$F_1$, $F_2$,.....$F_{38}$ are bits of the number.

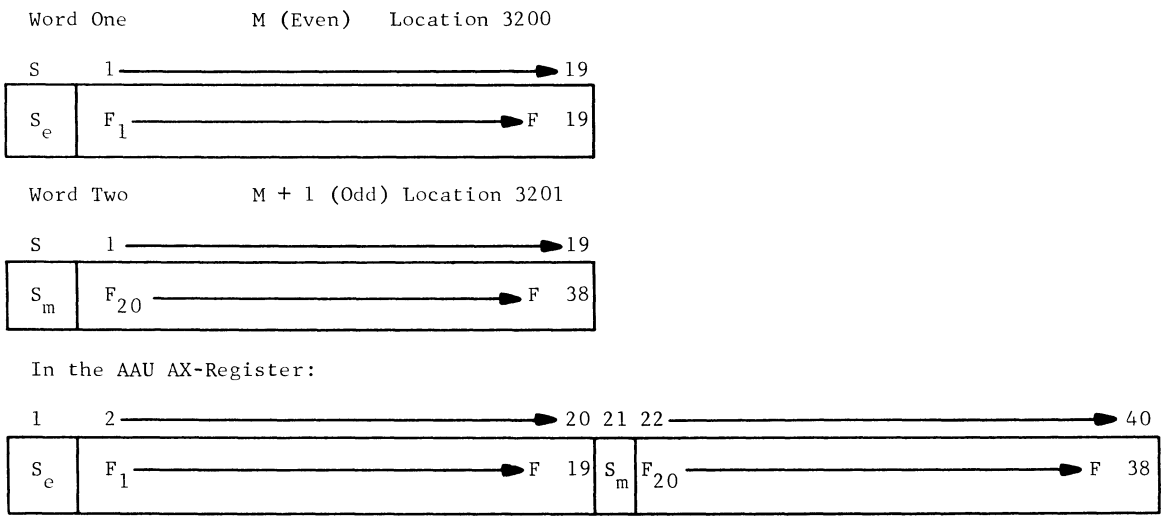
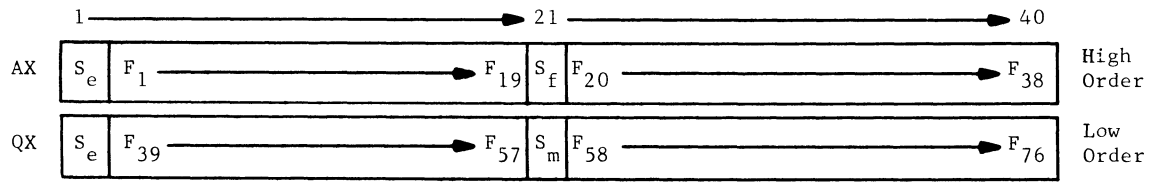


Figure 2. Fixed-Point Data Word Format


1 21 40
AX Se F1 F19 Sf F20 F38 High Order
QX Se F39 F57 Sm F58 F76 Low Order

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

Figure 3. Format of a "Product" of a Fixed-Point Number

An example of fixed-point numbers follows:

| | AX-Register Bit Position | | | | Remarks |
|---|---|---|---|---|---|
| Value | 1 | 2 - 20 | 21 | 22 - 40 | |
| 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 | 2'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.

EXAMPLE: Illegal fixed-point numbers

| AX-register | 1 | 2 - 20 | 21 | 22 - 40 |
|---|---|---|---|---|
| | 1 | 0 . . . . . 0 | 0 | 0 . . . . . 0 |
| | 1 | 0 . . . . . 0 | 1 | 0 . . . . . 0 |

FLOATING-POINT WORDS. Floating-point words consist of two parts, an exponent and a mantissa. There are four basic terms used in floating-point arithmetic operations:

1. Exponent. As used with the AAU, the exponent (or characteristic) constitutes the 9 most significant bits (8 numeric bits and a sign bit) of a double word (see Figure 4). These bits designate to what power of 2 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 4).

   The mantissa is multiplied by 2 raised to the exponential power expressed in bits 1-9 to give the entire word the desired numeric value.

3. Radix Point. In any numbering system the radix point 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 10 in the AX-register) is a 1-bit. For each position shifted left, 1 is subtracted from the exponent.

   Negative numbers are normalized by shifting the mantissa left until its most significant bit (bit 10 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, 1 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 formats shown in Figure 4. The binary point is assumed to be before the first bit (bit 10) of the mantissa. This format produces a binary number with a 30-bit mantissa and a binary characteristic range of

-256 to +255. These are approximately equal to a decimal number with a 9-digit mantissa and a decimal range of $10^{-77}$ to $10^{+77}$. The use of two words 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

$M_1 \ldots M_{30}$ = Bits of the mantissa (fraction)

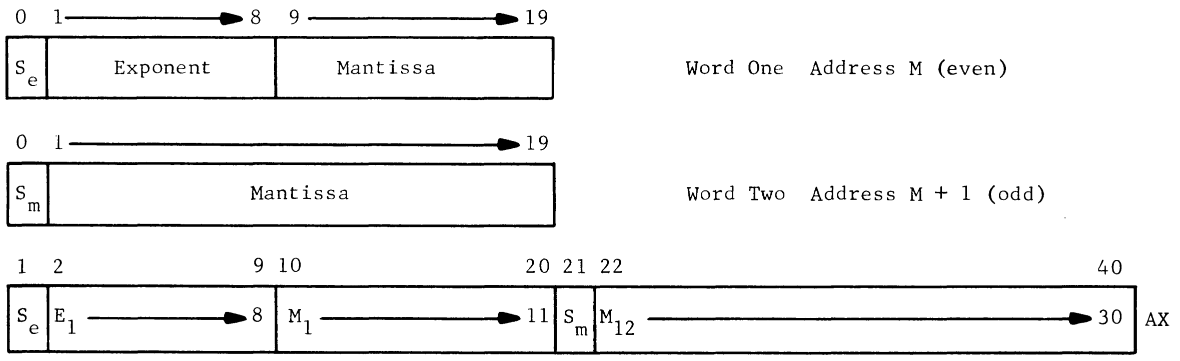


Figure 4. Floating-Point Data Word Format

If the floating-point number is the result of multiplication, addition, or subtraction, the minor (low order) half appears in the QX-register in the format shown in Figure 5. The major (high order) half appears in the AX-register.

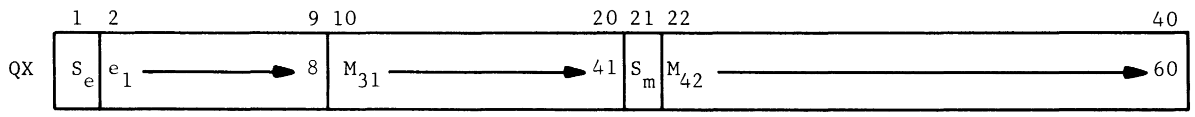


Figure 5. Floating-Point Number

The value of the QX exponent, e, is set equal to the value of the AX exponent, E, minus 30 (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, it uses 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 Position

| | 1 | 2 - 9 | 10 - 20 | 21 | 22 - 40 |
|---|---|---|---|---|---|
| Largest Positive | 0 | 11.......11 | 11.....1. | .0. | .11.....11 |
| +1 | 0 | 00.......01 | 10.....0. | .0. | .00.....00 |
| Smallest Positive | 1 | 00.......00 | 00.....0. | .0. | .00.....01 |
| 0 | 0 | 00.......00 | 00.....0. | .0. | .00.....00 |
| Smallest Negative | 1 | 00.......00 | 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 a negative sign. The "smallest negative" is the number nearest to zero with a 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.

EXAMPLE 1.

AX-Register Bit Positions

| | S | 1 - 8 | 9 - 19 | 20 | 21 - 39 |
|---|---|---|---|---|---|
| Illegal floating-point number | 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 causes the exponent to be reset to zero. The operation then proceeds as though the number were zero.

| | 1 | 2 - 9 | 10 - 20 | 21 | 22 - 40 | |
|---|---|---|---|---|---|---|
| Treated as zero | 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 is treated as $-256_{10}$.

| | 1 | 2 - 9 | 10 | 11 - 20 | 21 | 22 - 40 | |
|---|---|---|---|---|---|---|---|
| $1 \times 2^{\pm 256}$ | 1 | 0.....0 | 1 | 0.......0 | 0 | 0......0 | or any legal mantissa except all zeros |

Subroutines. 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 G-E 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. A common exponent will probably not 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. 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 $+377_8$ $(+255_{10})$.

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 resulting exponent.
- During a right shift 1 and add 1 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 is the number of leading 0's in a positive mantissa or leading 1's in a negative mantissa.

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

Whenever underflow occurs, the AX and QX registers are cleared to zero if the AAU is in the normalized floating-point mode. This assures that zero is always a valid replacement for the true result when underflow occurs. The clearing of AX and QX registers does not cause reset of the underflow.

## INSTRUCTIONS

Instructions for the AAU are divided into four categories, as follows:

1. General instructions:

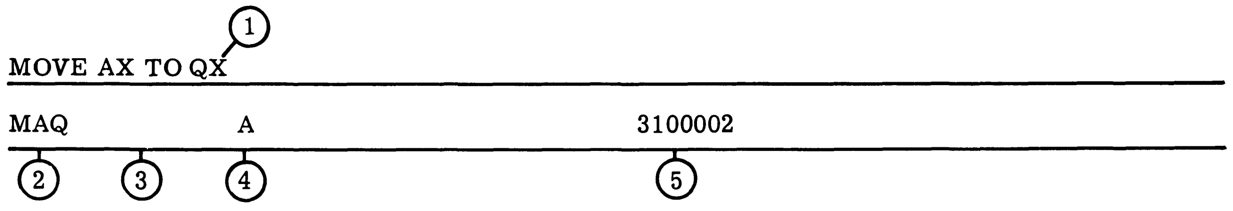
   a. Control (sets mode of operation)
   b. Data transfer (within the AAU)
   c. Reset (underflow/overflow)
   d. Normalization

2. Arithmetic Instructions.
   Arithmetic operations; add, subtract, multiply, and divide.

3. Data Transfer (load/store) between the AAU and central processor.

4. 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 is explained on the following page.

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 instruction affects the AAU registers as it affects 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; all other bits are 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. Control instructions set the mode of operation to be performed upon the data words received.

The mode of operation must be set by a Set Mode instruction before an arithmetic instruction is given to provide 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.

The operator can set the mode of operation of the AAU manually by pressing the desired mode indicator lamp switch.

### 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.

DATA TRANSFER INSTRUCTIONS. AAU 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.

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. Reset instructions are used to reset the OVERFLOW HOLD and UNDERFLOW HOLD indicators.

RESET OVERFLOW HOLD

| | | |
|---|---|---|
| ROV | | 3100004 |

The OVERFLOW HOLD indicator is turned off.

RESET UNDERFLOW HOLD

| | |
|---|---|
| RUN | 3200004 |

The UNDERFLOW HOLD indicator is turned off.

RESET INDICATORS

| | |
|---|---|
| RIN | 3500004 |

The UNDERFLOW HOLD and the OVERFLOW HOLD indicators are turned off.

The OVERFLOW HOLD and UNDERFLOW HOLD indicators will have been set as a result of overflow and underflow conditions.

NORMALIZE INSTRUCTION. This instruction operates correctly only on floating-point data. The instruction itself is not mode dependent.

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 (10) for positive numbers, or a 0-bit is detected in AX (10) for negative numbers. For each bit position shifted, 1 is subtracted from the exponent in AX. Bits from QX (10) shift into position AX (40). 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 6 is an illustration of the initiation of an instruction.

Bits S-4 of the instruction word define 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.

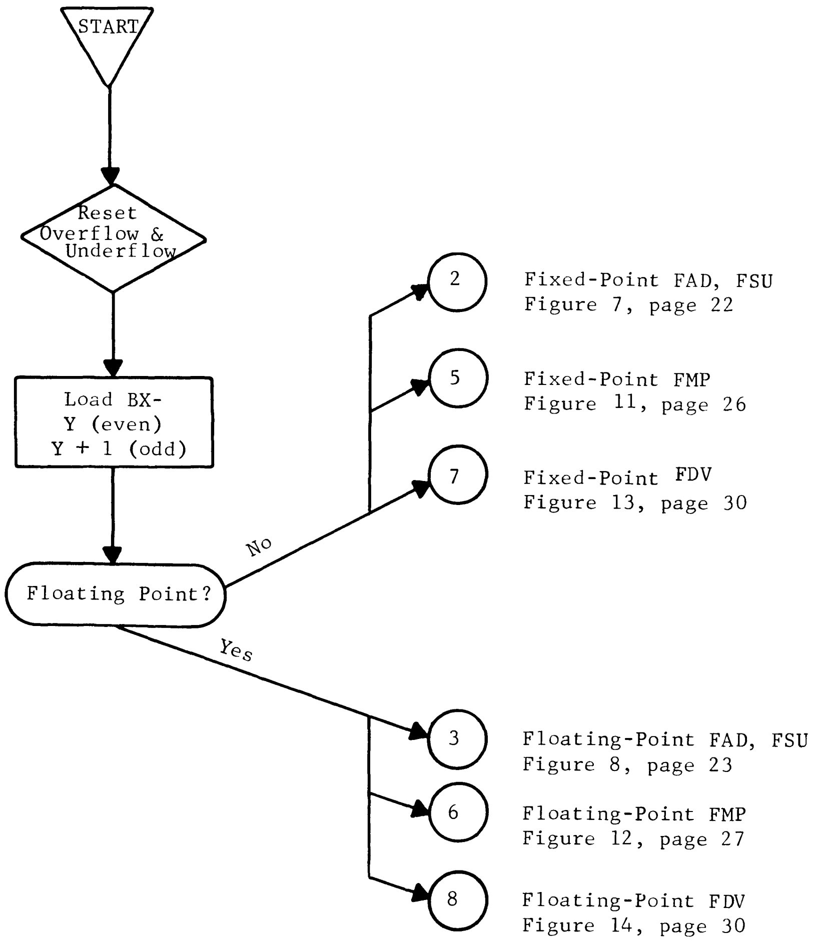


Figure 6. Initiation of Arithmetic Instruction

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-20) appears in the odd location and then BX (21-40) is written over BX (S-20) 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. Arithmetic operations are performed by the AAU upon each operand 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 three 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 1 and 21 (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 the add operation. The AX-register remains in an overflow condition.
- Underflow is set 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.

## Normalized Floating-Point Mode

- The floating-point number in M and M+1 are loaded into the BX-register.
- The QX-register is cleared.
- The AX- and BX-registers are examined for zero content and if either contains zeros, the nonzero number is placed in the AX-register and the contents are normalized. If neither AX nor BX is 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 1 to the exponent for each position shifted, until the exponents are equal. Bits shifted out of AX-3 are shifted into QX-10; bits shifted out of QX-40 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 1 to the exponent.
- AX and QX mantissas are examined for zero content. If they contain zeros, the exponents are cleared and the operation is terminated.
- 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-10 are reintroduced into AX-40; AX and QX are left-shifted together. Zeros are introduced into QX-40.
- The 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 21) is set to the same value as the AX mantissa sign.
- A check is made to determine whether overflow or underflow occurred.
- Overflow is set if the exponent sum in the AX-register exceeds $+255_{10}$. The AX- and QX-registers remain in an overflow condition.
- Underflow is set if the exponent sum in AX is less than $-256_{10}$. The AX- and QX-registers are cleared.

Unnormalized Floating-Point Mode. All the conditions set forth for normalized floating-point mode prevail except that 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 7.

Figures 8 and 9 illustrate the execution of a floating-point add operation.

## 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 1 and 21 (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 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.

### 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.
- If both registers contain legal numbers, they are examined for zero content. If either one contains 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 1 to the exponent for each position shifted, until the exponents are equal. Bits shifted out of AX-40 are shifted into QX-10; bits shifted out of QX-40 are lost.
- When exponents are aligned, the mantissas are algebraically added, and any mantissa overflow 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 1 to the exponent.
- AX and QX mantissas are examined for zero content, and, if they 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 between the AX- and QX-registers is normalized. Leading zeros (for positive mantissas) and one (for negative mantissas) are stripped by left-shifting AX and QX and subtracting from the exponent; bits from QX-10 are reintroduced into AX-40; AX and QX are left-shifted together. Zeros are introduced into QX-40.
- The 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 (21) is set to the same value as the AX mantissa sign.
- A check is made to determine whether 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 if the exponent result in AX is less than $-256_{10}$. The AX-and QX-registers are cleared.

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

Figure 7 illustrates the execution of the FSU instruction during a fixed-point mode of operation.

Figures 8 and 9 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.

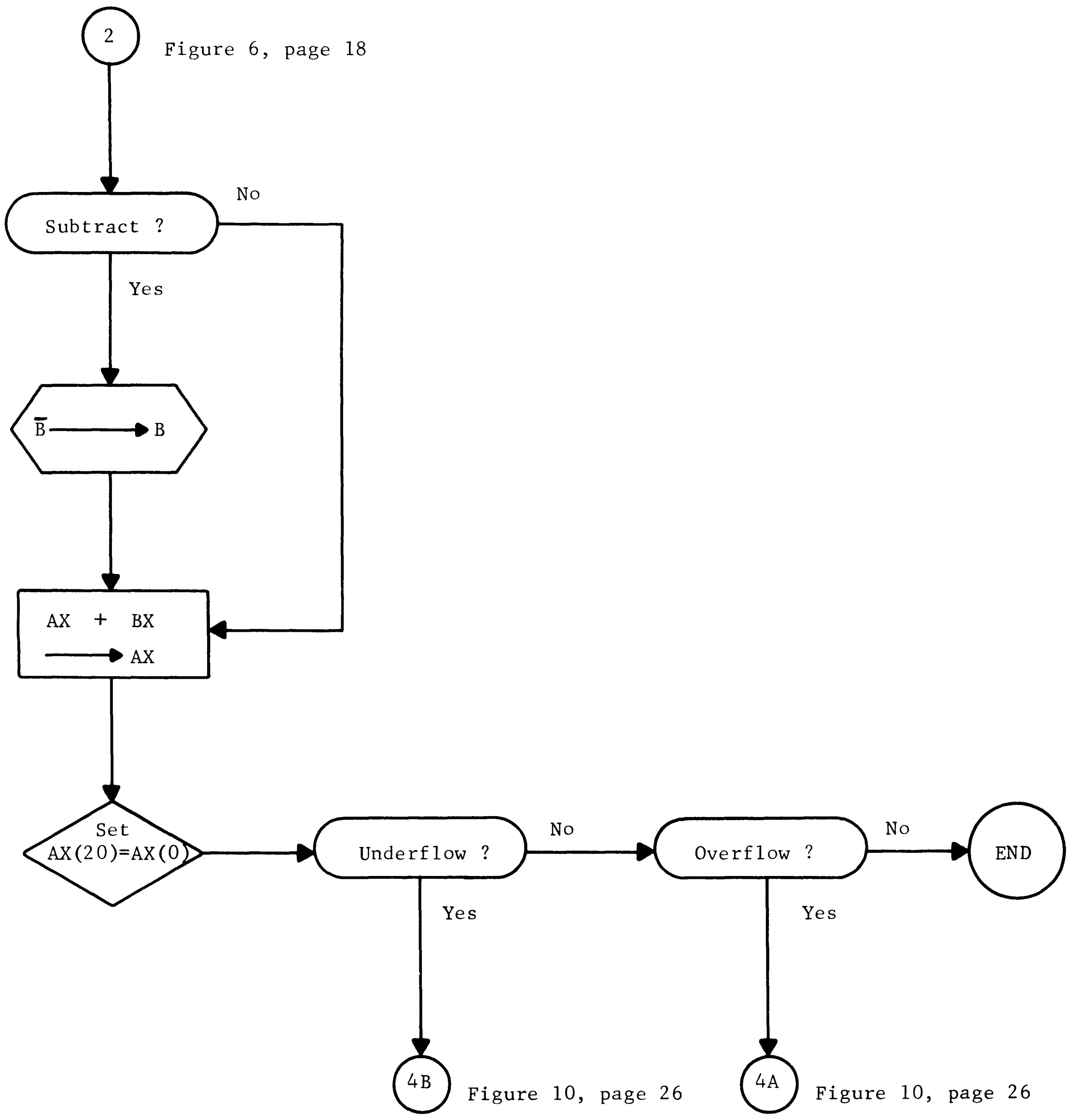


Figure 7. Fixed-Point FAD, FSU

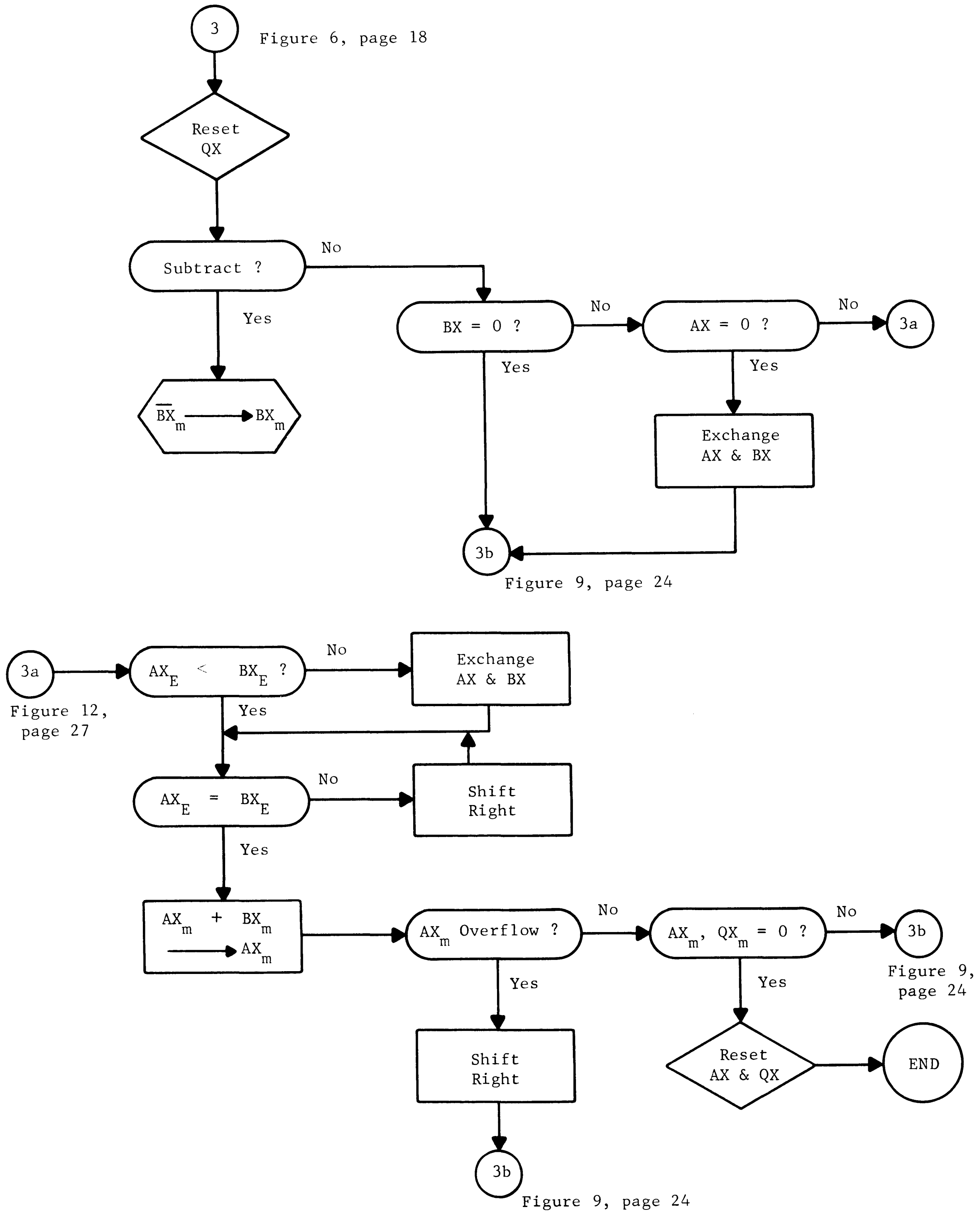


Figure 8. Floating-Point FAD, FSU

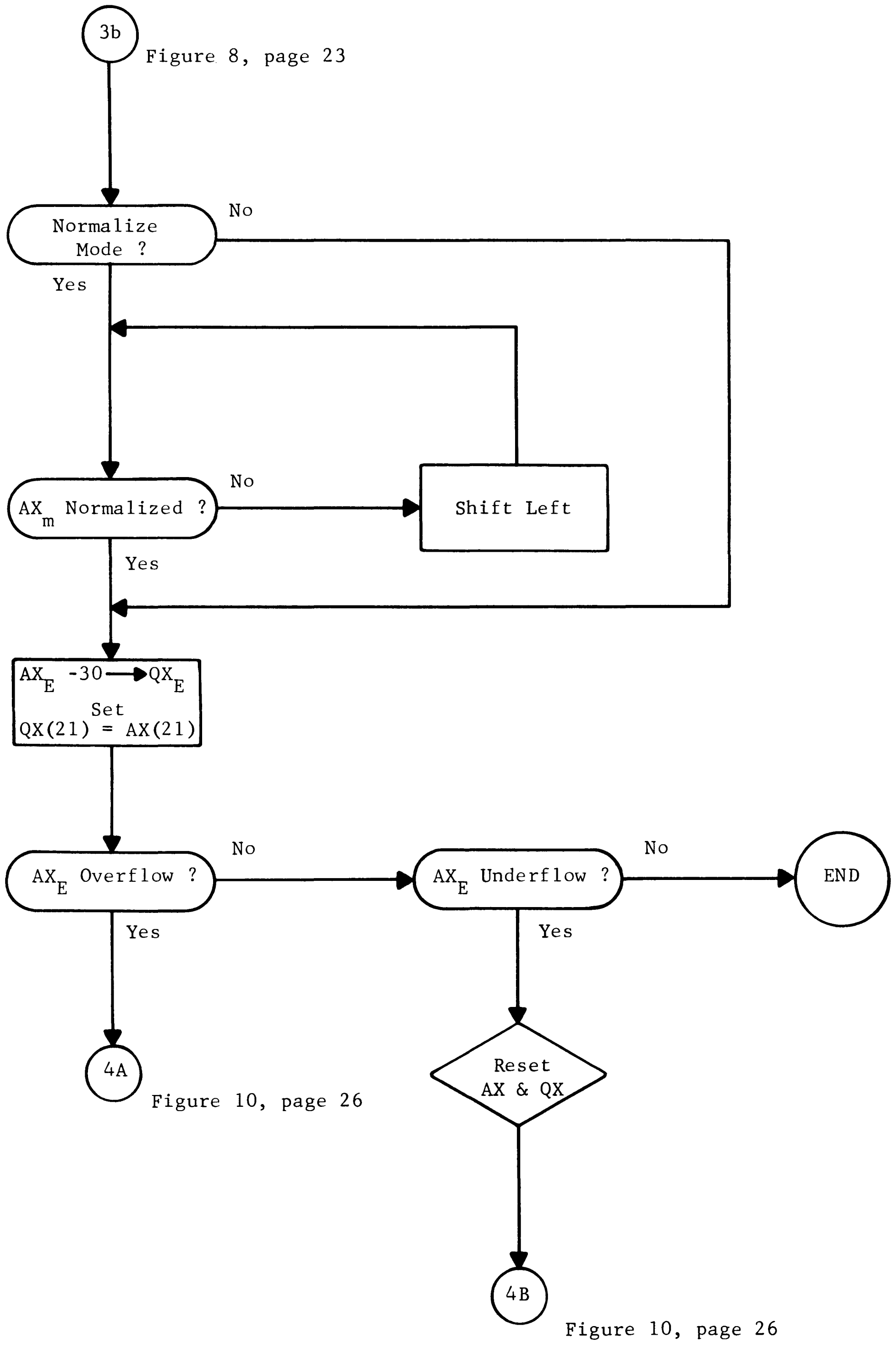


Figure 9. Normalization, Floating-Point Numbers

Fixed-Point Mode

- M and M+1 are loaded into the BX-register. The AX-register is cleared. Bit 21 of the QX-register is set equal to QX-1.
- If both registers contain legal numbers, they are examined for zero content. 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 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.

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 $AX_E$.
- 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 $BX_E$ is stored in $AX_E$.
- The BX and QX mantissas are examined for zero content. If they contain only 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 right-shifted together until a 1-bit is detected in position QX-40. 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 product 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(21) and AX(21) are set to agree.
- A check is made to determine whether overflow or underflow occurred.
- Overflow is set if the product in the AX-register exceeds $+255_{10}$.
- Underflow is set if the exponent product in AX is less than $-256_{10}$. The AX- and QX-registers are cleared.

Unnormalized Floating-Point Mode. All conditions set forth for normalized floating-point mode prevail except that the product in AX- and QX-registers is left in unnormalized form.

Figure 11 illustrates the execution of the FMP instruction during fixed-point operation.

Figure 12 illustrates the execution of the FMP instruction during floating-point operation.

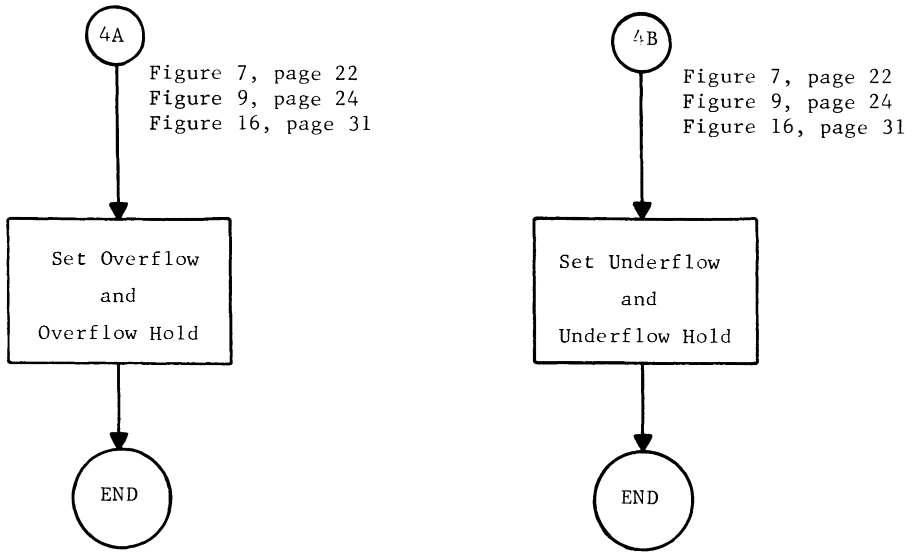


Figure 10. Setting of Underflow, Overflow Indicators

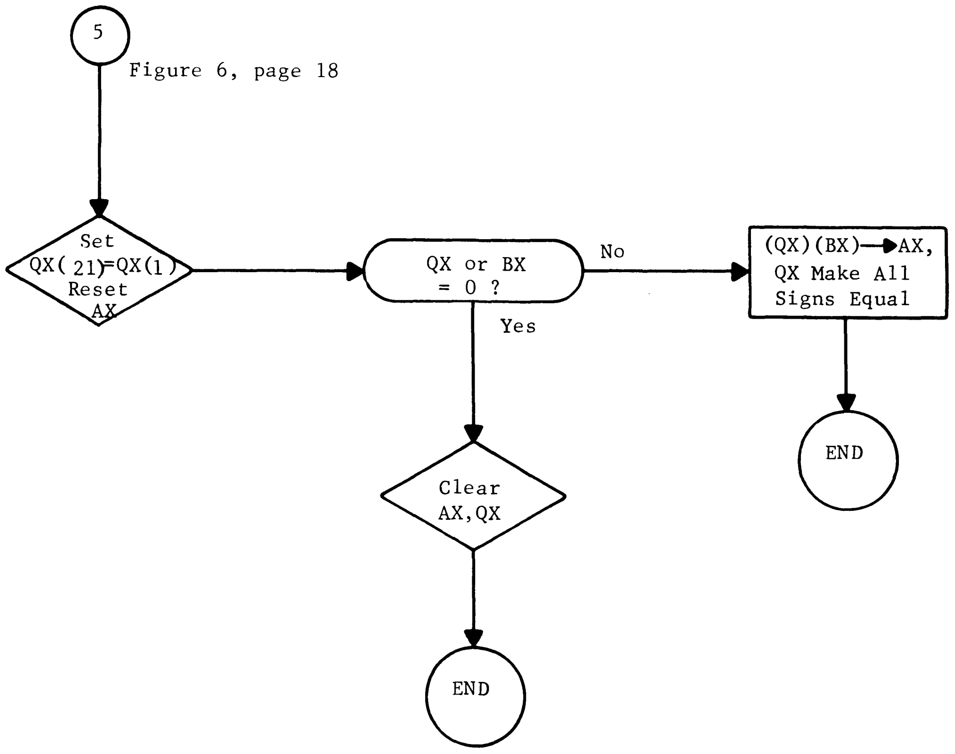

5
Figure 6, page 18
Set QX(21)=QX(1) Reset AX
QX or BX = 0 ?
No
Yes
(QX)(BX)→AX, QX Make All Signs Equal
END
Clear AX,QX
END

Figure 11. Fixed-Point FMP

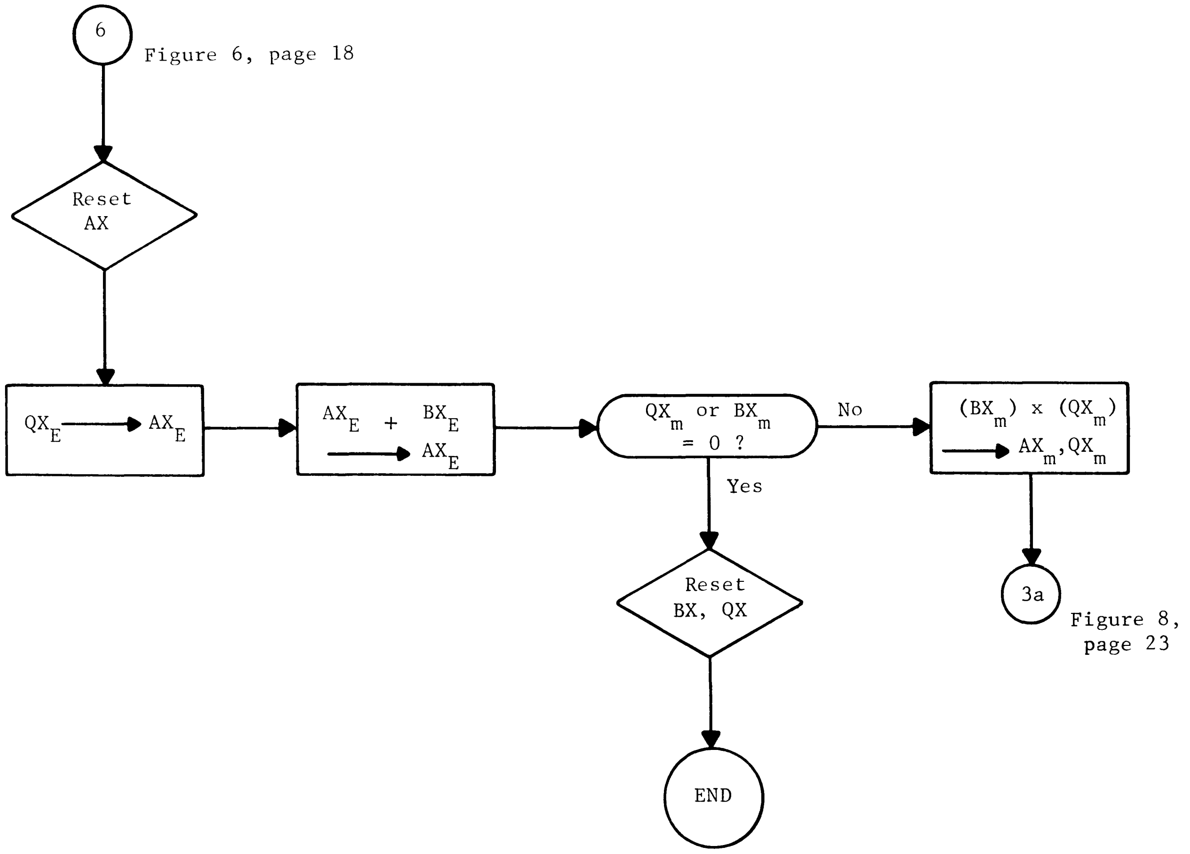

6
Figure 6, page 18
Reset AX
$QX_E \longrightarrow AX_E$
$AX_E + BX_E \longrightarrow AX_E$
$QX_m$ or $BX_m$ = 0 ?
No
Yes
$(BX_m) \times (QX_m) \longrightarrow AX_m, QX_m$
Reset BX, QX
END
3a
Figure 8, page 23

Figure 12. 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-1 determines the sign of the entire number in AX- and QX-registers), the AX- and QX-registers are complemented.
- If the registers contain legal numbers the BX-register is checked to determine whether it contains zeros (this is the divisor). If the divisor in BX is zero, an error condition exists and the OVERFLOW indicator will be set.

- A check is made to determine whether 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 either is true, the instruction is terminated with the OVERFLOW indicator 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.

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-21 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 is transferred 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 exponent of the quotient is formed by algebraically subtracting the divisor exponent from the dividend exponent.

- A check is made to determine whether the mantissa in the BX-register contains zeros. If so, the OVERFLOW indicator is set in the AAU.

- A check is made to determine whether the absolute value of the dividend mantissa is equal to or greater than the absolute value of the divisor mantissa. If either is true, the mantissas of AX and QX are shifted right one place and the exponents are increased by 1.

- Another check is made to determine whether the absolute value of the dividend mantissa is equal to or greater than the absolute value of the divisor mantissa, and, if it is, the OVERFLOW indicator is set in the AAU. If they are not equal to or greater than the absolute value of the divisor mantissa, 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 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.

- Overflow is set if the exponent of the quotient in the AX-register exceeds $+255_{10}$.
- Underflow is set if the resultant AX exponent is less than -256; the AX- and QX-registers are cleared.

Unnormalized Floating-Point Mode. All conditions set forth for normalized floating-point mode prevail except that the quotient in the AX-register is left in unnormalized form.

Figure 13 illustrates the execution of the FDV instruction during fixed-point operation.

Figures 14, 15 and 16 illustrate the execution of the FDV instruction during floating-point operation.

## Data Transfer Instructions

Data transfer instructions 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 the AX-register.
- Contents of M (even) are loaded into AX 1-20.
- Contents of M+1 (odd) are loaded into AX 21-40.
- Contents of M and M+1 are unchanged.
- Contents of M (odd) are loaded into AX 1-20 and 21-40.
- Illegal numbers may be loaded.

### STORE AX REGISTER

| | | |
|---|---|---|
| FST | M X | 33MMMMM |

- Contents of the AX-register replace contents of M and M+1 if M is even.
- Contents of the AX-register are not changed.
- Contents of the AX-register 1-20 are stored in M if M is odd and then the contents of AX 21-40 are written over 1-20 in the same location M.

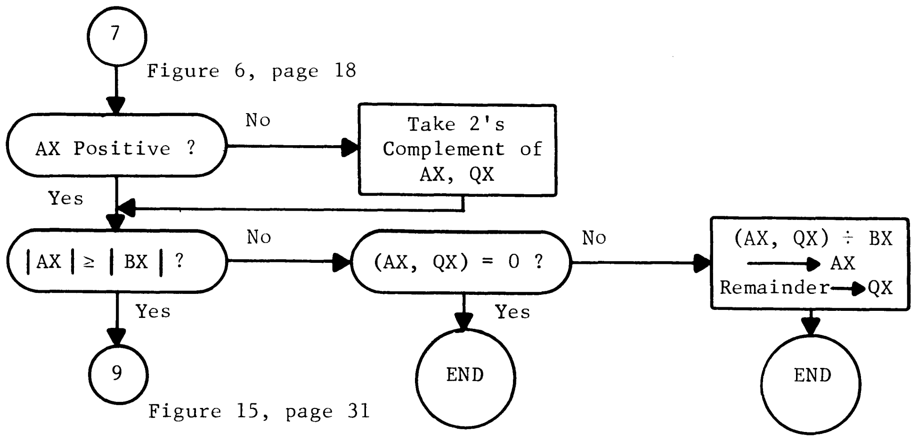


Figure 13. Fixed-Point FDV

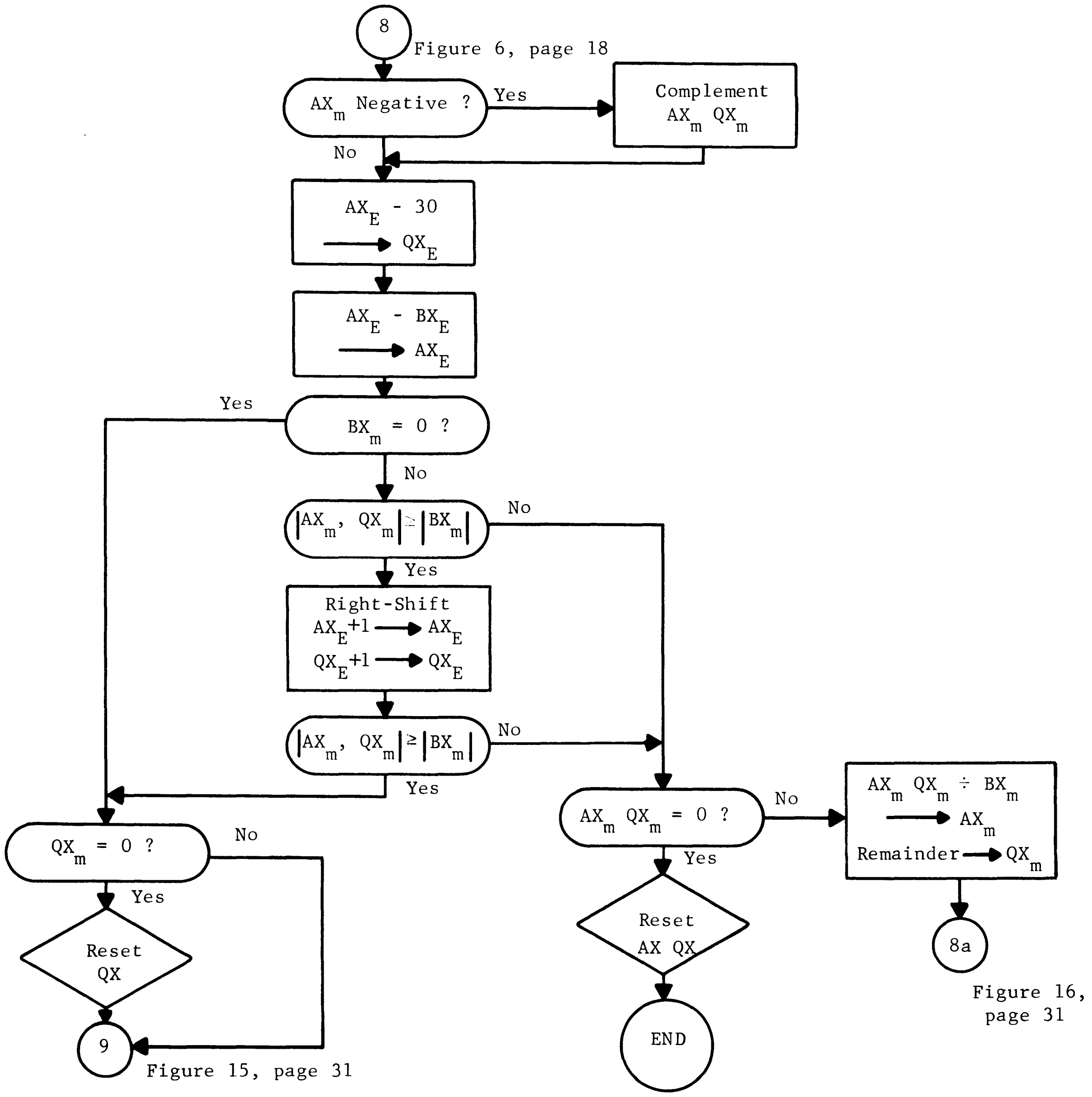


Figure 14. Floating-Point FDV

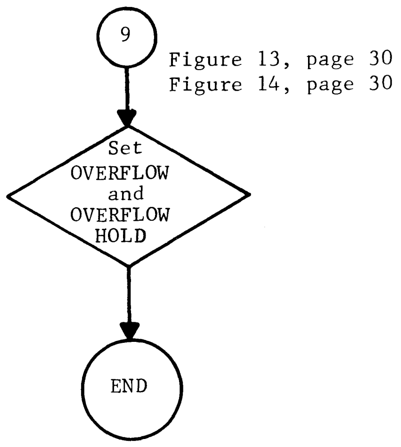


Figure 15. Setting Divide Check Indication

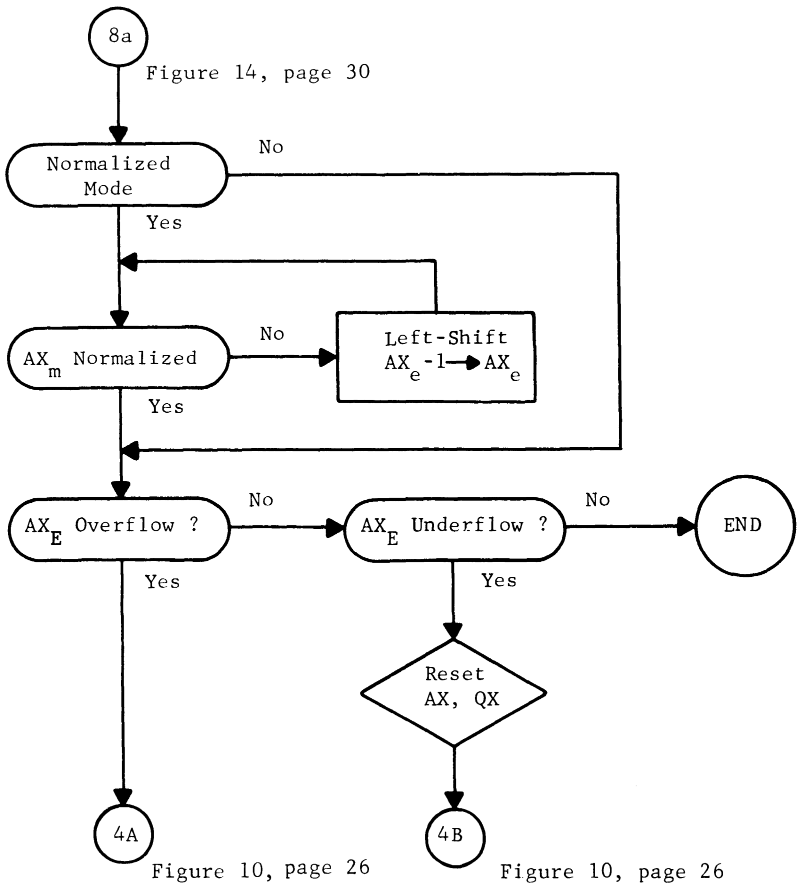


Figure 16. Normalization Floating-Point FDV

## 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 whether it is ready to receive an instruction.

BRANCH ON AAU NOT READY

| | | | |
|---|---|---|---|
| BAR | BAN | 7 | 2516720 |

The AAU is tested to determine whether 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 1, if in the fixed-point mode or for a minus sign in bit position 21, 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 21 (sign of the mantissa) if in the floating-point mode or for a plus sign in bit position 1, 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: The above branches are not tests for the OVERFLOW HOLD indicator.

| BRANCH ON UNDERFLOW | BRANCH ON NO UNDERFLOW |
|---|---|
| BAR BUF 7 2514724 | BAR BNU 7 2516724 |
| The UNDERFLOW indicator is tested for an on condition (indicator is not reset). | The UNDERFLOW indicator is tested for an off condition (indicator is not reset). |

NOTE: The above branches are not tests for the UNDERFLOW HOLD indicator.

| BRANCH ON OVERFLOW HOLD ON | BRANCH ON OVERFLOW HOLD OFF |
|---|---|
| BAR BOO 7 2514725 | BAR BON 7 2516725 |
| The AAU is tested for the OVERFLOW HOLD on. If the indicator is on, it is turned off. | The AAU is tested for the OVERFLOW HOLD off. If the indicator is on, it is turned off. |

| BRANCH ON UNDERFLOW HOLD ON | BRANCH ON UNDERFLOW HOLD OFF |
|---|---|
| BAR BUO 7 2514726 | BAR BUN 7 2516726 |
| The AAU is tested for the UNDERFLOW HOLD on. If the indicator is on, it is turned off. | 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 or UNDERFLOW indicators on (indicators are not reset). | The AAU is tested for either the OVERFLOW or UNDERFLOW indicators off (indicators are not reset). |

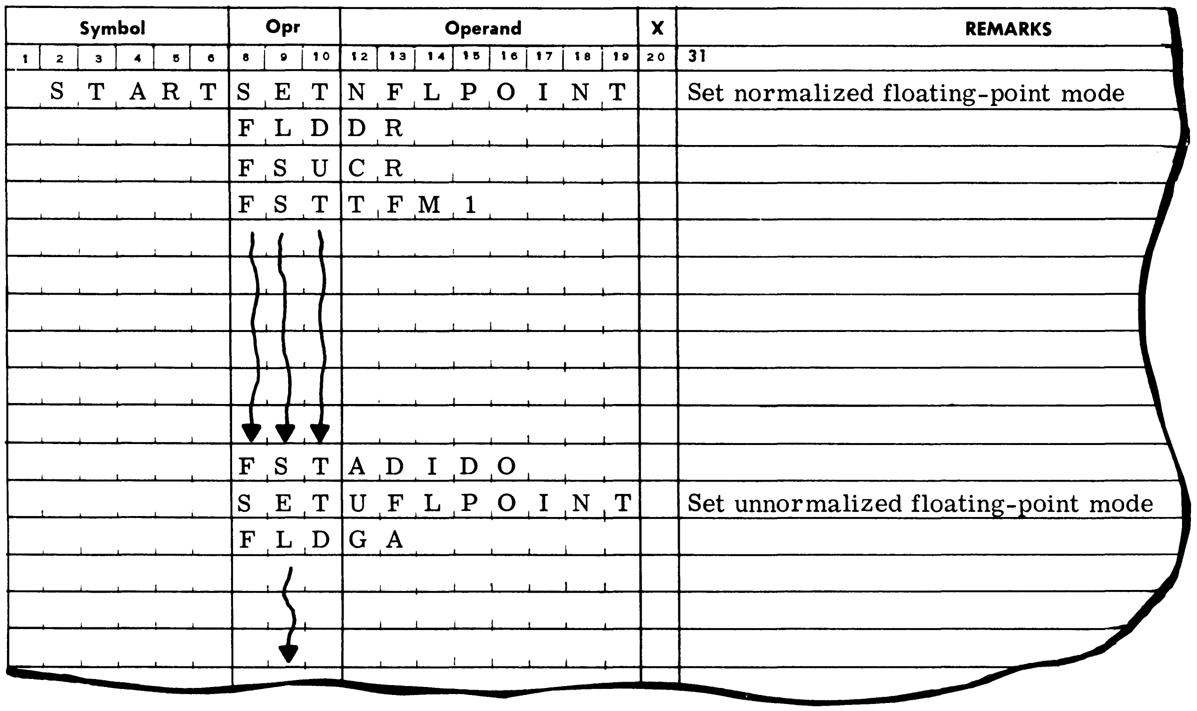
## PROGRAM CONSIDERATIONS

Unlike peripherals whose controllers have 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 17 shows a program which begins calculations in the normalized floating-point mode and then switches to the unnormalized floating-point 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 |
| START | SET | NFLPOINT | | Set normalized floating-point mode |
| | FLD | DR | | |
| | FSU | CR | | |
| | FST | TFM1 | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | FST | ADIDO | | |
| | SET | UFLPOINT | | Set unnormalized floating-point mode |
| | FLD | GA | | |
| | | | | |
| | | | | |
| | | | | |

Figure 17. Setting Mode for Calculation

# 3. OPERATING THE 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 those performed by their counterpart registers, A, B, Q, and I in the central processor.

The 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.

## AAU CONTROLS AND INDICATORS

The AAU operator panel is composed of register displays, mode and alarm indicators and operating switches. A detailed description follows Figure 18 showing the operator panel of the AAU.

### AX-and QX-Register Indicators

The operator can see the contents of the AX- and QX-registers from 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 21. 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 21, is ignored.

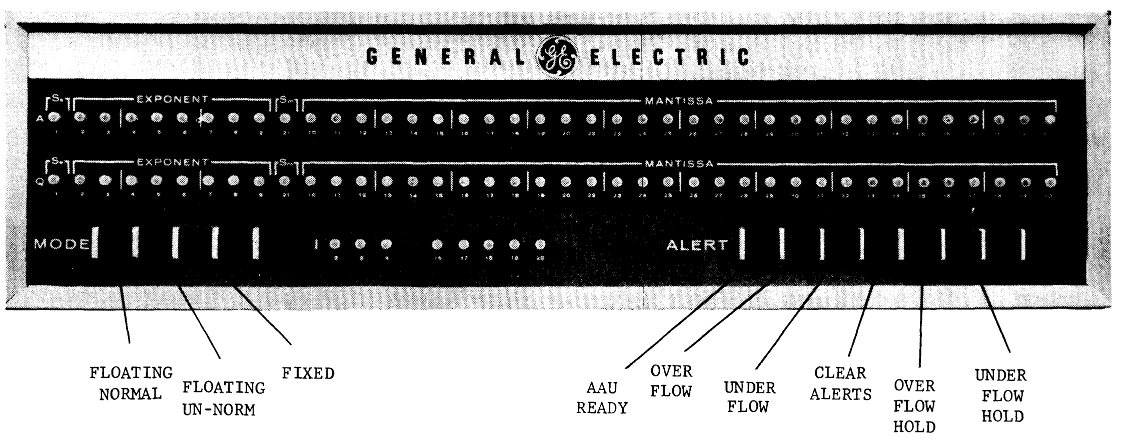


Figure 18. AAU Control and Indicator Panel

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:

| $S_e$ | Exponent | | | | | | | | $S_m$ | Mantissa | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 21 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 1 | 1 | | | 1 | 1 | 1 | | 1 | | 1 | 1 | | | | 1 | | | | 1 | | 1 | 1 | 1 | 1 | | | 1 | | | | | | 1 | 1 | | | | 1 | 1 |
| - | 2 | | | 3 | | | 5 | | + | | 6 | | | 1 | | | 0 | | | 5 | | | 7 | | | 1 | | | 0 | | | 1 | | | 4 | | | 3 | |

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

## Mode Indicators and Switches

There are three mode indicators and switches on the control and indicator panel. When the program specifies the mode, the applicable mode indicator is lit (FLOATING NORMAL, FLOATING UN-NORM, or FIXED). These three indicators are also switches, and can establish the mode, when pressed. However, their use as switches is for maintenance personnel only. When the program makes an unscheduled halt, the operator should note which of the three mode indicators is lit.

## 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, 19 and 20 (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.

## Alert Indicators and Switches

AAU READY INDICATOR. The AAU READY indicator glows 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. The AAU is not in a Ready status when it is busy doing an internal instruction. It becomes Ready when an instruction is completed. The AAU can be tested by the program for Ready or not Ready.

OVERFLOW AND UNDERFLOW INDICATORS. The two conditions of overflow and underflow have indicators on the panel and can be tested by the program. The OVERFLOW and OVERFLOW HOLD indicators are lit as a result of a carry out of bit position one when two positive registers are added, or as a result of an illegal divide. The OVERFLOW indicator is reset (turned off) when the AAU accepts the next instruction which accesses memory or general instruction (XAQ, MAQ, etc.), or when the operator presses the CLEAR ALERTS switch. The UNDERFLOW and UNDERFLOW HOLD indicators are turned on when a carry is missing as a result of the addition of two negative registers. The UNDERFLOW indicator is turned off by the next instruction accepted by the AAU, by general instruction (XAQ, MAQ, etc.), or by pressing the CLEAR ALERTS switch.

CLEAR ALERTS SWITCH. The CLEAR ALERTS switch clears and resets the following.

- The OVERFLOW alarm
- The OVERFLOW HOLD alarm
- The UNDERFLOW alarm
- The UNDERFLOW HOLD alarm

The CLEAR ALERTS switch remains on whenever the d-c power is on.

# APPENDIX

## AAU PROGRAM LIBRARY ROUTINES

### Internal Data Routines

| | Library Number |
|---|---|
| Fixed-Point BCD to Binary--Provides conversion from binary-coded decimal to binary for AAU operations. | CD225C1.002 |
| Floating-Point Binary to BCD Conversion (BDCONA)--AAU--Provides floating-point binary to floating-point BCD conversion with rounding. | CD225C2.006 |
| Floating-Point Binary to BCD Conversion--AAU--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--AAU--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 | $0 \le X < \infty$ |
| $J_0$, J | $0 \le X < \infty$ |
| $K_0$, $K_1$ | $0 < X < \infty$ |
| $Y_0$, $Y_1$ | $0 < X < \infty$ |

| | |
|---|---|
| Complex Arithmetic Package with Trace Option--AAU--Provides a means of handling complex floating-point numbers by means of a package of subroutines. | CD225D1.004 |
| Double Precision Fixed-Point Arctangent--AAU--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 |
| Double Precision Fixed-Point $\text{Log}_{(2,e,10)}x$--AAU--Computes $\log_\alpha x$, where $\alpha$ 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 |
| Double Precision Fixed-Point $\text{Exponential}(2,e,10)^x$--AAU--Computes $\alpha^x$, where $\alpha$ is 2, e, or 10 and x is the independent variable in fixed-point form at a binary point of 8. | CD225D2.028 |

|  | Library Number |
|---|---|
| Double Precision Fixed-Point SIN-COS--AAU--Computes the sine or cosine of a fixed-point argument in radians between -256 and +255. | CD225D2.020 |
| Double Precision Fixed-Point Square Root--AAU--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--AAU--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$_{(2,e,10)}$$^x$ using AAU--Computes $\alpha^x$ where $\alpha$ is 2, e, or 10 and x is the independent variable in normalized floating-point form. | CD225D2.012 |
| Floating-Point Log$_{(2,e,10)}$$^x$ using AAU--Computes log $\alpha^x$ where $\alpha$ is 2, e, or 10 and x is the independent variable in normalized floating-point form. | CD225D2.014 |
| Floating-Point Sine/Cosine--AAU--Computes the sine or cosine of a normalized floating-point argument in radians. | CD225D2.008 |
| Floating-Point Square Root--AAU--Computes the square root of a normalized floating-point number, in the range of 0 to $2^{255}$. | CD225D2.006 |
| Gamma Function--AAU--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--Tenth Order--AAU--Fits polynomials up to the tenth order through points $(X_1, Y_1)$, $(X_2, Y_2)$, .... $(X_n, Y_n)$ by the method of least squares, and calculates residuals and total variance to facilitate choosing the best fit. | CD225D6.002 |
| Least Squares Polynomial Curve Fit Program--AAU--Fits polynomials up to the tenth order through a maximum of 400 points $(X_i, Y_i)$. Standard error, variance, and residuals are provided. | CD225D6.004 |
| Linear Programming--AAU--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 using the AAU--Solves a set of n linear real simultaneous equations in n unknowns whose coefficients are represented in normalized floating-point form and stores the results in the same form. | CD225D4.012 |
| Matrix Transpose--AAU Version--Finds the transpose of a real matrix A(m,n) whose elements are double words and stores the results in the same form in matrix B(n,m). | CD225D4.006 |
| Multiple Linear Regression Program--AAU--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 |

| | Library Number |
|---|---|
| Normalized Floating-Point Matrix Inversion using the AAU--Computes the inverse of an N x N matrix, A, whose elements are real and represented in normalized floating-point form and stores the result, B, in the same form. | CD225D4.010 |
| Normalized Floating-Point Matrix Multiply using the AAU--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 stores the resultant matrix B(m,n) in the same form. | CD225D4.004 |
| Normalized Floating-Point Matrix Add or Subtract using the AAU--Computes the sum or difference of two m x n real matrices, A, B, whose elements are represented in normalized floating-point form and stores the result C in the same form. | CD225D4.002 |
| Normalized Floating-Point Scalar Multiply using the AAU--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 stores the resultant matrix B(m,n) in the same form. | CD225D4.008 |
| Roots of Polynomial using AAU--Calculates all n roots of the polynomial. | CD225D5.002 |
| AAU 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 Subroutine (FLIN)--AAU--Reads cards and converts 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 (OUT)--AAU--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 (PADAR)--AAU--Reads fixed-format decimal cards and converts the fields into BCD, floating-point or fixed-point. Also checks 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

| | |
|---|---|
| GE-225 WIZ System--AAU--Algebraic compiler. | CD225H2.002 |
| GE-225 WIZ-II System--AAU--Algebraic compiler. | CD225H2.004 |

## Assembly Systems

| | |
|---|---|
| GE-225 F1.002 ZOOM--A Macro Assembler. | CD225F1.002 |

## LIST OF INSTRUCTIONS

The abbreviations used in the mnemonics of the operand and symbol X fields of the General Assembly Program are as follows:

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

| Opr | Operand | X | Octal | Description | | Timing in Microseconds | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | GE-205/215 | | GE-225 | |
| | | | | | | Min. | Max. | Min. | Max. |
| BAR | BAN | 7 | 2516720 | Branch on AAU not ready | | 72 | 72 | 36 | 36 |
| BAR | BAR | 7 | 2514720 | Branch on AAU ready | | 72 | 72 | 36 | 36 |
| BAR | BER | 7 | 2514727 | Branch on AAU error | | 72 | 72 | 36 | 36 |
| BAR | BMI | 7 | 2514721 | Branch on AAU minus | | 72 | 72 | 36 | 36 |
| BAR | BNE | 7 | 2516727 | Branch on AAU no error | | 72 | 72 | 36 | 36 |
| BAR | BNO | 7 | 2516723 | Branch on AAU no overflow | | 72 | 72 | 36 | 36 |
| BAR | BNU | 7 | 2516724 | Branch on AAU no underflow | | 72 | 72 | 36 | 36 |
| BAR | BNZ | 7 | 2516722 | Branch on AAU nonzero | | 72 | 72 | 36 | 36 |
| BAR | BOO | 7 | 2514725 | Branch on overflow hold on | | 72 | 72 | 36 | 36 |
| BAR | BON | 7 | 2516725 | Branch on overflow hold off | | 72 | 72 | 36 | 36 |
| BAR | BOV | 7 | 2514723 | Branch on AAU overflow | | 72 | 72 | 36 | 36 |
| BAR | BPL | 7 | 2516721 | Branch on AAU plus | | 72 | 72 | 36 | 36 |
| BAR | BUF | 7 | 2514724 | Branch on AAU underflow | | 72 | 72 | 36 | 36 |
| BAR | BUO | 7 | 2514726 | Branch on underflow hold on | | 72 | 72 | 36 | 36 |
| BAR | BUN | 7 | 2516726 | Branch on underflow hold off | | 72 | 72 | 36 | 36 |
| BAR | BZE | 7 | 2514722 | Branch on AAU zero | | 72 | 72 | 36 | 36 |
| FAD | M | X | 31MMMMM | Add | F | 211.5 | 283.5 | 139.5 | 193.5 |
| | | | | | N | 279 | 898 | 207 | 808 |
| | | | | | U | 256.5 | 463.5 | 184.5 | 373.5 |

| Operand | X | Octal | Description | GE-205/215 Min. | GE-205/215 Max. | GE-225 Min. | GE-225 Max. |
|---|---|---|---|---|---|---|---|
| FDV | X | 36MMMMM | Floating Point Divide | F-1134 | 1424.25 | 1062 | 1298.25 |
| | | | | N- 954 | 1491.75 | 882 | 1365.75 |
| | | | | U- 931.5 | 1266.75 | 859.5 | 1140.75 |
| FLD | X | 3000000 | Floating Point Load | 144 | 144 | 72 | 72 |
| FMP | X | 35MMMMM | Floating Point Multiply | F- 404 | 913.5 | 332 | 823.5 |
| | | | | N- 414 | 1251 | 342 | 1161 |
| | | | | U- 369 | 778.5 | 297 | 688.5 |
| FST | X | 3300000 | Floating Point Store | 144 | 144 | 72 | 72 |
| FSU | X | 32MMMMM | Floating Point Subtract | F- 211.5 | 283.5 | 139.5 | 193.5 |
| | | | | N- 279 | 898 | 207 | 808 |
| | | | | U- 256.5 | 463.5 | 184.5 | 373.5 |
| LAQ | A | 3600002 | Load AX From QX | 94.5 | 139.5 | 58.5 | 85.5 |
| LQA | A | 3200002 | Load QX From AX | 94.5 | 139.5 | 58.5 | 85.5 |
| MAQ | A | 3100002 | Move AX To QX | 94.5 | 139.5 | 58.5 | 85.5 |
| ROV | | 3100004 | Reset Overflow Hold | 94.5 | 139.5 | 58.5 | 85.5 |
| RUN | | 3200004 | Reset Underflow Hold | 94.5 | 139.5 | 58.5 | 85.5 |
| SET FIXPOINT | | 3500010 | Set Fixed Point | 94.5 | 139.5 | 58.5 | 85.5 |
| SET NFLPOINT | | 3100010 | Set Normalized Floating Point | 94.5 | 139.5 | 58.5 | 85.5 |
| SET UFLPOINT | | 3200010 | Set Unnormalized Floating Point | 94.5 | 139.5 | 58.5 | 85.5 |
| XAQ | A | 3500002 | Exchange AX And QX | 162 | 207 | 126 | 153 |

Timing in Microseconds

*Progress Is Our Most Important Product*

GENERAL ELECTRIC

INFORMATION SYSTEMS DIVISION

CPB-325A