1	IDENTIFICATION
1.1	Digital-7-31-F-Sym
1.2	Signed Multiply Subroutine - Single Precision
1.3	February 15, 1965

2. ABSTRACT

This subroutine forms a 34-bit signed product from 17-bit signed multiplier and multiplicand.

3. REQUIREMENTS

3.1 Storage

This subroutine uses 47 (decimal) memory locations.

4. USAGE

4.2 Calling Sequence

The subroutine is called by the JMS instruction. When the JMS is executed to enter the subroutine the multiplier must be in the accumulator (AC). The location following the JMS must contain a LAC with the address of the multiplicand.

The subroutine will return the instruction immediately following the latter location with the least significant part of the product in the AC. The most significant part of the product will be stored in location MP5.

6. DESCRIPTION

Reference to the flowchart (10.1) will illustrate the following discussion.

6.1.1 On entry, the sign of the multiplier is tested, and if negative, the multiplier is made positive.

6.1.2 The multiplicand is obtained and tested for 0. If it is found equal to 0, a jump to the exit is executed. Next the sign of the multiplicand is tested; and if it is found negative, the multiplicand is made positive.

6.1.3 At this point, the contents of the link are as follows:

Sign of Multiplier	Sign of Multiplicand	Links
0	0	0
0	1	1
1	0	1
1	1	0

and represent, therefore, the sign of the product.

6.1.4 The multiply loop proper (tagged MP4) is entered. During this loop, the least significant half of the product shifts into the most significant end of MP5 while the multiplier shifts out the least significant end of MP5 and is lost. Note that the sign of the product is retained in MP5.

6.1.5 The sign of the product is tested. If positive, the subroutine exits. If negative, complementation of the product is performed before the exit.

6.3 Scaling

Upon entry the binary point is assumed to be located between bit positions 0 and 1 in both multiplier and multiplicand. Since there are 17 magnitude bits in each of the two factors, the product will contain 34 magnitude bits.

The product is double signed, i.e., bit positions 0 and 1 of the most significant word of the product both contain the sign. The remaining 16 bits of the most significant word of the product are magnitude bits.

The least significant word of the product is devoted entirely to magnitude.

If the binary point of the factors are as stated above, the binary point of the product will be located between bit positions 1 and 2 in the most significant portion of the product.

On entry, multiplier and multiplicand must be 2's complement binary. After return, the product is contained in two words in 2's complement form.

For more information on binary scaling for fixed-point computers, see Application Note 501.

7. METHOD

7.2 Algorithm

The conventional algorithm is used. The least significant bit of the multiplier is tested. If it is equal to 1, the multiplicand is added to the developing product and this quantity is shifted right. If the least significant bit of the multiplier is 0, no addition is made before the shift. The process is repeated until all the bits of the multiplier in order from least significant to most significant have been processed.

9. EXECUTION TIME

9.1 Minimum

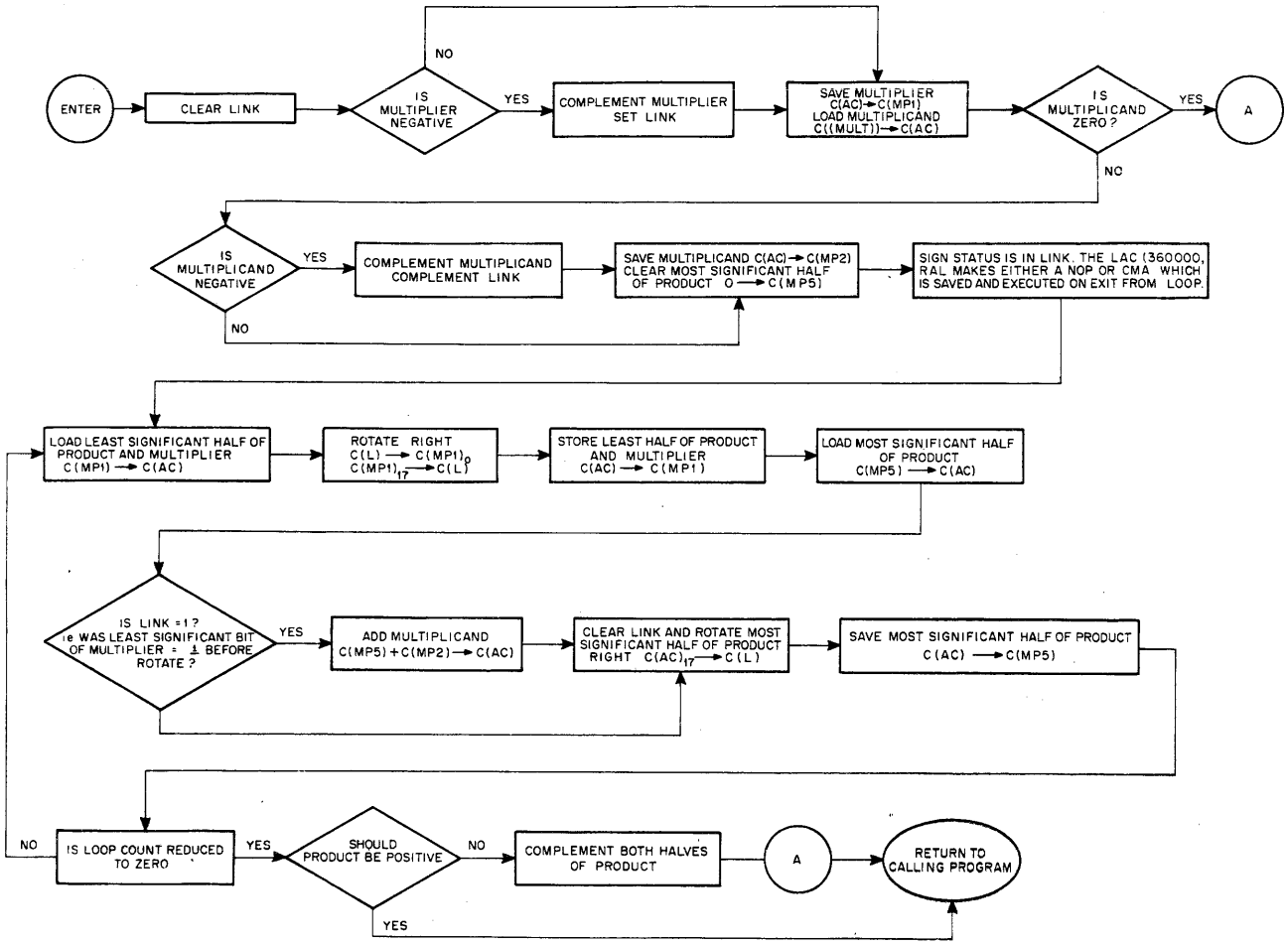
When the subroutine discovers that the multiplicand is 0, the multiplication loop is bypassed. In this case, execution time will be 14 microseconds.

9.2 Maximum

Maximum execution time occurs when the sign of the product is negative and the multiplier consists (in binary) of all ones. The time is approximately 570 μsec.

10. PROGRAM

10.1 Flowchart



10.2 Example

The C(Y) are tested. If C(Y) = 0, C(MP1) = C(MP5) = 0. If C(Y) is not 0, then C(Y) → C(MP2), C(MP5) are cleared and multiplication is carried out as follows:

If $C(MP1)_{17}$ contains a 1, C(MP2) are added to C(MP5). The contents of MP5 and the MP1 are then shifted right one bit. If $C(MP1)_{17} = 0$, the contents of MP5 and those of the MP1 are shifted right one bit.

For this example, assume that the registers MP1, MP5 and MP2 are five bits in length instead of 17. The following sequential steps will occur in a multiply operation. The multiplicand is 9 and the multiplier is 4.

MP5	MP1	Y	Comments
00000	01001	00100	Initial contents of the register MP1, ready to be tested.
00100	01001		C(MP2) + C(MP5) →C(MP5) since $C(MP1)_{17}$ is a 1.
00010	00100		C(MP5, MP1) rotated right one place. $C(MP1)_{17}$ is tested.
00001	00010		No addition, because $C(MP1)_{17}$ is 0. C(MP5, MP2) rotated right one bit and AC_{17} is tested.
00000	10001		No addition $C(MP1)_{17} = 0$, C(MP5, MP1) rotated right one bit. $C(MP1)_{17}$ is tested.
00100	10001		C(MP2) + C(MP5) C(MP5) since $C(MP1)_{17}$ is a 1.
00010	01000		C(MP5, MP1) rotated right.
00001	00100		No addition $C(MP1)_{17} = 0$, C(MP5, MP1) rotated right one bit. Rotation counter indicates that the multiplication is complete, since it has been reduced to 0.

10.3 Program Listing

A listing of the subroutine with MULT located at address 0200 is as follows:

```
        /CALLING SEQUENCE:
                /LAC    MULTIPLIER
                /JMS    MULT
                /LAC    MULTIPLICAND
                /RETURN ;LOW ORDER PRODUCT IN AC
                        /HIGH ORDER PRODUCT IN LOCATION MP5
0200    0000    MULT,       0
0201    7100                DZM MP5     /ZERO OUT PRODUCT AREA
```

```
0202    7510            SNA             /IS MULTIPLIER ZERO?
0203    7061            JMP MPZ         /IF ZERO, RETURN
0204    3250            SPA ! CLL       /TAKE ABSOLUTE VALUE OF MULTIPLIER
0205    3251            CMA ! CML       /SET LINK = 1 IF MULTIPLIER IS NEGATIVE
0206    1600            DAC #MP1
0207    7450            XCT I MULT      /PICK UP MULTIPLICAND
0207    7450            SNA             /IS MULTIPLICAND ZERO
0210    5234            JMP MPZ         /IF ZERO, RETURN
0211    7510            SPA             /IF NON ZERO, TAKE ABSOLUTE VALUE
0212    7061            CMA ! CML       /IF NEGATIVE, COMPLEMENT LINK
0213    3252            DAC #MP2        /LINK HAS SIGN OF PRODUCT
0214    1247            LAC (360000)    /COMPLEMENT ACCUMULATOR IF PRO-
                                        /DUCT IS NEGATIVE
0215    3253            RAL
0216    1250            DAC MPSIGN
0217    7010            LAM -21         /INITIALIZE COUNT TO -17
0220    3250            DAC #MP3
0221    1251    MP4,    LAC MP1
0222    7430            RAR             /ROTATE MULTIPLIER RIGHT ONE BIT
0223    1252            DAC MP1         /LOW ORDER INTO LINK
0224    7110            LAC MP5         /FETCH PRODUCT
0225    3251            SZL ! CLL
0226    2253            TAD MP2         /ADD MULTIPLICAND IF LINK IS 1
0227    5216            RAR             /ROTATE PRODUCT RIGHT ONE BIT
0230    1250            DAC MP5
0231    7010            ISZ MP3         /IS COUNT + 1 = 0?
0232    7430            JMP MP4         /IF NOT, GO TO MP4
0233    5240    MPSIGN, 0               /IF YES COMPLEMENT HIGH ORDER PORTION
0234    3250            DAC MP5         /OF PRODUCT, IF IT IS NEGATIVE
0235    1251            LAC MP1         /RETRIEVE LOW ORDER BIT OF PRODUCT
0236    2200            RAR             /FROM THE LINK
0237    5600            XCT MPSIGN      /PLACE IT INTO THE LOW ORDER PORTION
                                        /OF WORD COMPLEMENT AS ABOVE
0240    7141    MPZ,    ISZ MULT
0241    3250            JMP I MULT      /RETURN
```

STORAGE MAP: (Locations available to the user)
MP5 (C(MP5) = high order product)