

PROGRAM LIBRARY
MATH ROUTINES

PDP-8

PDP-8

PROGRAM LIBRARY

MATH ROUTINES

January, 1968

Order No. DEC-08-FFAA-D from Program Library, Maynard, Mass. Price \$ 1.50

Direct comments concerning this manual to Software Quality Control, Maynard, Mass.

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

Reprinted
September, 1967

Copyright 1968 by Digital Equipment Corporation

CONTENTS

<u>Part</u>		<u>Page</u>
1.	Single-Precision Square Root DEC-08-FMAA-D	1 - 1
2.	Single-Precision Signed Multiply Subroutine DEC-08-FMBA-D	2 - 1
3.	Single-Precision Signed Divide Subroutine DEC-08-FMCA-D	3 - 1
4.	Double-Precision Signed Multiply DEC-08-FMDA-D	4 - 1
5.	Double-Precision Signed Divide Subroutine DEC-08-FMEA-D	5 - 1
6.	Double-Precision Sine Subroutine DEC-08-FMFB-D	6 - 1
7.	Double-Precision Cosine Subroutine DEC-08-FMGB-D	7 - 1
8.	Four-Word Floating-Point Package DEC-08-FMHA-D	8 - 1
9.	Logical Subroutines DEC-08-FMIA-D	9 - 1
10.	Arithmetic Shift Subroutines DEC-08-FMJA-D	10 - 1
11.	Logical Shift Subroutines DEC-08-FMKA-D	11 - 1

1. Single Precision Square Root, DEC-08-FMAA-D

2. ABSTRACT

This subroutine will extract the square root of a single-precision integer. Given an input N ($0 \leq N < 2^{12}$), it will produce an integer K and a remainder R , such that $N = K^2 + R$.

3. REQUIREMENTS

3.1 Storage

This subroutine uses 23 (decimal) memory locations.

3.3 Equipment

Standard PDP-8

4. USAGE

4.1 Loading

The library tape that is supplied is a symbolic tape. It does not begin with an origin setting, although it does end with a dollar sign. The binary tape produced by assembling this tape, or the binary tape produced by assembling this tape with other tapes, is loaded with the Binary Loader.

4.2 Calling Sequence

This subroutine is called with an effective JMS SQRT with the argument in the accumulator. The subroutine returns control to the location following the JMS with the answer in the accumulator and with the remainder in the register tagged SQRT.

6. DESCRIPTION

6.2 Examples and/or Applications

The following program will illustrate the use of this subroutine:

400

```
CLA
TAD  X
JMS  I      SQRTPT
HLT
```

X, 0145 (1101 DECIMAL)
SQRTPT, SQRT

This sample program will halt at location 403 with 0012 (octal) or 10 (decimal) in the accumulator. Register SQRT (address 0222) will contain 0001, the remainder.

7. METHODS

7.2 Algorithm

The algorithm makes use of the fact that the sum of the odd integers is a square:

$$\sum_{K=1}^N (2K-1) = 2 \sum_{K=1}^N K - \sum_{K=1}^N 1 = 2 \left(\frac{N}{2}\right)(N+1) - N = N^2$$

9. EXECUTION TIME

9.4 Timing Equation

If the answer is N , the time for the subroutine is

$$(30 + N (25.5)) \mu\text{sec}$$

10. PROGRAM

10.4 Program Listing


```

/DEC 08-FMAA-LA
/SQUARE ROOT ,..... ENTER WITH SQUARE IN AC
/                               EXITS WITH ROOT IN AC
/                               ODD INTEGER METHOD
0200 0000      SQRI,      0
0201 3222      DCA SQRI      /SAVE INPUT
0202 3226      DCA ROOT      /0 TO ANSWER
0203 1223      TAD SQRI      /-1; FIRST ATTEMPT
0204 3225      SQX,      DCA SQRI
0205 1222      TAD SQRI      /COMPARE INPUT
0206 7100      CLL          /WITH THIS TRY
0207 1225      TAD SQRI
0210 7420      SNL
0211 5217      JMP SQRI      /TEST>INPUT; ALL DONE
0212 2226      ISZ ROOT      /ADD +1 TO ANSWER
0213 3222      DCA SQRI      /INPUT=INPUT-TEST
0214 1225      TAD SQRI
0215 1224      TAD SQRI      /TEST=TEST-2
0216 5204      JMP SQX      /CONTINUE
0217 7200      SQRI,      CLA
0220 1226      TAD ROOT      /FETCH ANSWER
0221 5600      JMP I SQRI    /EXIT
0222 0000      SQRI,      0
0223 7777      SQRI,      -1
0224 7776      SQRI,      -2
0225 0000      SQRI,      0
0226 0000      ROOT,      0

```

PAUSE

\$

THERE ARE NO ERRORS

SYMBOL TABLE

ROOT	0226
SQRI	0225
SQRI	0217
SQRI	0200
SQRI	0222
SQRI	0223
SQRI	0224
SQX	0204

1. Signed Multiply Subroutine - Single Precision, DEC-08-FMBA-D.
2. ABSTRACT
This subroutine forms a 22-bit signed product from 11-bit signed multiplier and multiplicand.
3. REQUIREMENTS
 - 3.1 Storage
This subroutine uses 44 (decimal) memory locations.
4. USAGE
 - 4.1 Loading
The library tape that is supplied is a symbolic tape. It does not begin with an origin setting, although it does end with a dollar sign. The binary tape produced by assembling this tape, or the binary tape produced by assembling this tape with other tapes, is loaded with the Binary Loader.
 - 4.2 Calling Sequence
The subroutine is called by an effective JMS MULT. When the JMS is executed to enter the subroutine, the multiplier must be in the accumulator (AC). The location following the JMS must contain the multiplicand. The subroutine returns to the instruction immediately following the latter location with the most significant part of the product in the AC. The least significant part of the product is stored in location MP1.
6. DESCRIPTION
 - 6.1 Discussion
Reference to the flow chart (11.1) 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 equals 0, a jump to the exit is executed. Next the sign of the multiplicand is tested, and if it is negative, the multiplicand is made positive.
 - 6.1.3 At this point, the content of the link is as follows:

<u>Sign of Multiplier</u>	<u>Sign of Multiplicand</u>	<u>Link</u>
0	0	0
0	1	1
1	0	1
1	1	0

and represents, therefore, the sign of the product.

- 6.1.4 The multiplication loop proper (tagged MP4) is entered. During this loop, the least significant half of the product shifts into the most significant end of MP1, while the multiplier shifts out the least significant end of MP1 and is lost. Note that the sign of the product is retained in MP1.
- 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.

6.2 Examples or Applications

Example (See 11.1 Flow Chart)

The $C(Y)$ are tested. If $C(Y) = 0$, $C(MP1) = C(MP5) = 0$. If $C(Y) \neq 0$, $C(Y) \rightarrow C(MP2)$, $C(MP5)$ are cleared and multiplication is carried out as described below.

If $C(MP1)_{11}$ contains a 1, $C(MP2)$ are added to $C(MP5)$. The contents of $MP5$ and the $MP1$ are then shifted right one bit. If $C(MP1)_{11} = 0$, 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 11. The following sequential steps occur in a multiply operation. The multiplicand is 9 and the multiplier is 4.

<u>MP5</u>	<u>MP1</u>	<u>Y</u>	<u>Comments</u>
00000	01001	00100	Initial contents of the register $MP1$ ready to be tested.
00100	01001		$C(MP2) + C(MP5) \rightarrow C(MP5)$ since $C(MP1)$ is a 1.
00010	00100		$C(MP5, MP1)$ rotated right one place. $C(MP1)_{11}$ is tested.
00001	00010		No addition, because $C(MP1)_{11}$ is 0. $C(MP5, MP2)$ rotated right one bit and AC_{11} is tested.
00000	10001		No addition, $C(MP1)_{11} = 0$, $C(MP5, MP1)$ rotated right one bit. $C(MP1)_{11}$ is tested.
00100	10001		$C(MP2) + C(MP5) \rightarrow C(MP5)$ since $C(MP1)_{11}$ is a 1.
00010	01000		$C(MP5, MP1)$ rotated right.
00001	00100		No addition, $C(MP1)_{11} = 0$, $C(MP5, MP1)$ rotated right one bit. Rotation counter indicates that the multiplication is complete since it has been reduced to 0.

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 11 magnitude bits in each of the two factors, the product contains 22 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 ten 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 points 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 position of the product.

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

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

7. METHOD

7.1 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 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, it bypasses the multiplication loop. In this case, execution time is 25.5 μ sec if the multiplier is positive and 27.0 μ sec if the multiplier is negative.

9.2 Maximum

Maximum execution time occurs when the sign of the product is negative and the multiplier consists (in binary) of all 1s. This time is approximately 350 μ sec.

10. PROGRAM

10.4 Program Listing

/DEC-08-FMBA

/TWO'S COMPLEMENT SINGLE PRECISION MULTIPLY ROUTINE

/RETURN HIGH ORDER PRODUCT IN AC, LOW IN MP1

0200	0000	MULT,	0	
0201	7100		CLL	
0202	7510		SPA	/TEST FOR NEGATIVE MULTIPLIER
0203	7061		CMA CML IAC	
0204	3250		DCA MP1	/STORE MULTIPLIER
0205	3251		DCA MP5	
0206	1600		TAD I MUL1	
0207	7450		SNA	/TEST FOR ZERO MULTIPLICAND
0210	5234		JMP MP5N*2	/JMP IF MULTIPLICAND=0
0211	7510		SPA	/TEST FOR NEGATIVE MULTIPLICAND
0212	7061		CMA CML IAC	
0213	3252		DCA MP2	/STORE MULTIPLICAND
0214	1247		TAD THIR	
0215	3253		DCA MP3	
0216	1250	MP4,	TAD MP1	/MULTIPLY LOOP PROPER
0217	7010		RAR	
0220	3250		DCA MP1	
0221	1251		TAD MP5	
0222	7430		SZL	/TEST IF MULTIPLICAND SHOULD BE ADDED
0223	1252		TAD MP2	
0224	7110		CLL RAR	
0225	3251		DCA MP5	
0226	2253		ISZ MP3	/TEST FOR END OF LOOP
0227	5216		JMP MP4	
0230	1250		TAD MP1	
0231	7010		RAR	
0232	7430	MPSN,	SZL	
0233	5240		JMP COMP	
0234	3250		DCA MP1	
0235	1251		TAD MP5	
0236	2200	MPZ,	ISZ MUL1	/EXIT TO CALLING PROGRAM
0237	5000		JMP I MUL1	
0240	7141	COMP,	CMA CML IAC	/COMPLEMENT PRODUCT
0241	3250		DCA MP1	
0242	1251		TAD MP5	
0243	7040		CMA	
0244	7430		SZL	
0245	7001		IAC	
0246	5236		JMP MPZ	
0247	7764	THIR,	7764	/ELEVEN IN DECIMAL
0250	0000	MP1,	0	
0251	0000	MP5,	0	
0252	0000	MP2,	0	
0253	0000	MP3,	0	

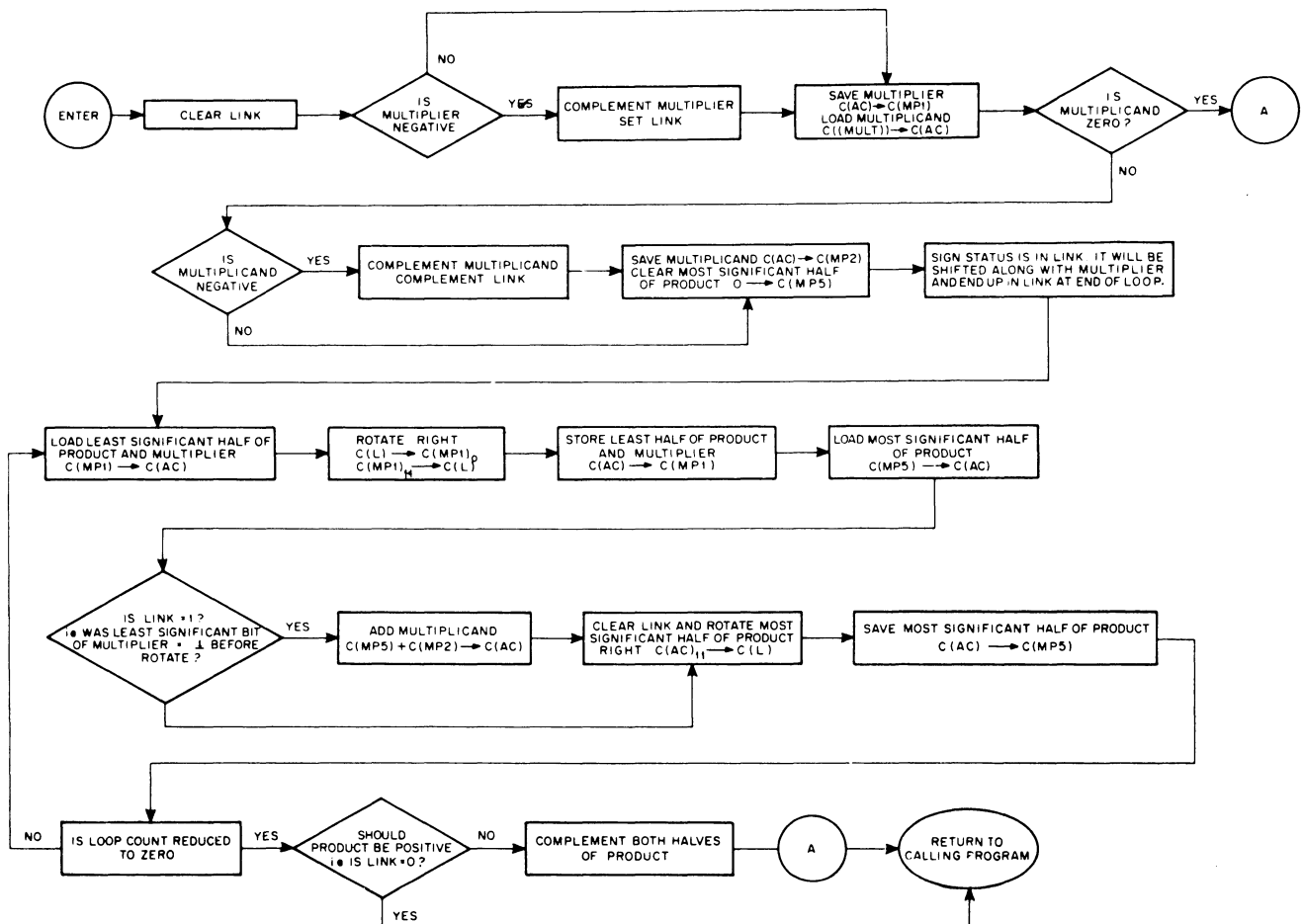
PAUSE

SYMBOL TABLE

COMP	0240
MPSN	0232
MPZ	0236
MP1	0250
MP2	0252
MP3	0253
MP4	0216
MP5	0251
MULI	0200
THIR	0247

11. DIAGRAMS

11.1 Flow Chart



1. Single Precision Signed Divide Subroutine, DEC-08-FMCA-D

2. ABSTRACT

The Single-Precision Divide Subroutine will divide a 12-bit signed divisor into a 24-bit signed dividend to produce a 12-bit signed quotient and a 12-bit signed remainder.

3. REQUIREMENTS

3.1 Storage

This subroutine requires 62 (decimal) memory locations. It is provided in two forms: binary tape assembled with an origin of 0200, and a symbolic tape with no origin setting and ending with a dollar sign.

4. USAGE

4.1 Loading

This subroutine requires 62 (decimal) memory locations. It is provided as a symbolic tape with no origin setting and ending with a dollar sign.

4.2 Calling Sequence

The subroutine is called with an effective JMS DIVIDE. The accumulator contains the high-order bits of the dividend; the location following the JMS contains the low-order bits of the dividend; the location following this contains the divisor; and the subroutine returns to the following location with the quotient in the accumulator and the remainder in C(HDIVND). If a divide error has occurred, C(L) = 1 and the accumulator contains 0, otherwise C(L) = 0.

TAD	HIGH D	/C(AC) = HIGH DIVIDEND
JMS	I DIVDP	/CALL DIVIDE
LOWD		/LOW DIVIDEND
DIVSOR		/DIVISOR
HLT		/C(AC) = QUOTIENT IF L = 0
DIVDP,	DIVIDE	/(0200)
HIGHD,		/HIGH DIVIDEND

4.5 Errors in Usage

There are two types of errors that may be encountered in using the divide subroutine, the first of which is tested by the routine. The divide may be represented as:

$$\frac{(\text{High-Order Dividend}) \cdot 2^{12} + \text{Low-Order Dividend}}{\text{Divisor}}$$

= Quotient, Remainder

or

$$(\text{High-Dividend}) \cdot 2^{12} + \text{Low-Dividend} = (\text{Quotient}) (\text{Divisor}) + \text{Remainder}.$$

Since $(\text{Quotient}) < 3777(8)$, it is possible that a divisor and dividend are so specified that no quotient may be found that satisfies this identity. If High-Order Dividend \geq Quotient, then the divide will not take place and C(L) will be 1. There are cases, however, that are not detected by this test. For example:

$$\begin{array}{r} 1777 \quad 7777 \\ \hline 2000 \end{array}$$

Since $(3777)(2000) + 3777 = 1000 \quad 1777$, there is no possible quotient that when multiplied by the divisor will yield the dividend.

5. RESTRICTIONS

See Section 4.5.

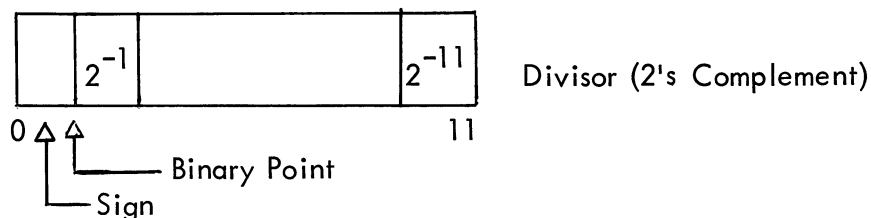
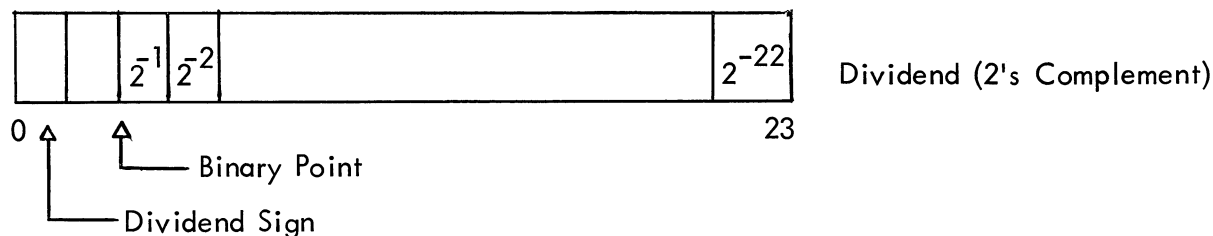
6. DESCRIPTION

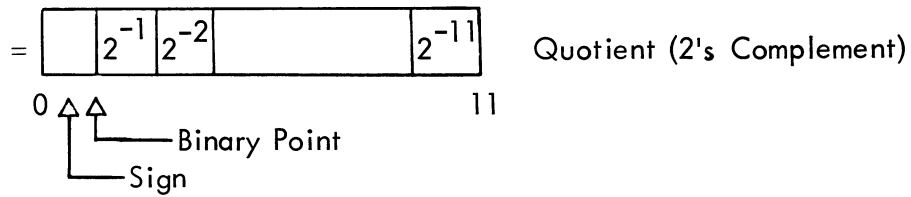
6.1 Discussion

The algorithm works by shifting the dividend left and comparing it with the divisor. If Dividend \geq Divisor then Dividend = Dividend-Divisor, and a bit is set in the quotient. This is repeated the proper number of times. The remainder will have the same sign as the dividend, and the quotient will be signed properly: (Dividend Sign) XOR (Divisor Sign) = (Quotient Sign).

6.3 Scaling

The Single-Precision Divide Subroutine is scaled analogous to the scaling of the Single-Precision Multiply Subroutine (DEC-08-FMBA, previously Digital-8-11-F). It may be thought of as either an integer divide or a fractional divide.





If High-Order Dividend = HD
 Low-Order Dividend = LD
 Quotient = Q
 Remainder = R
 Divisor = D

$$\frac{HD \cdot 2^{12} + LD}{D} = Q, R$$

so that $Q \cdot D + R = (HD) \cdot 2^{12} + (LD)$

or

$$\frac{(HD \cdot 2^{12} + LD) \cdot 2^{-11}}{D \cdot 2^{-11}} = Q \cdot 2^{-11}, R \cdot 2^{-11}$$

Examples:

(a)
$$\begin{array}{r} 000\ 000\ 000\ 000\ 000\ 000\ 000\ 111 \\ \underline{000\ 000\ 000\ 011} \end{array} = 000\ 000\ 000\ 010$$

Remainder = 000 000 000 001

$$\frac{7}{3} = 2, 1$$

(b)
$$\begin{array}{r} 000\ 100\ 000\ 000\ 000\ 000\ 000\ 000 \\ \underline{010\ 000\ 000\ 000} \end{array} = 010\ 000\ 000\ 000$$

Remainder = 000 000 000 000

$$\frac{\frac{1}{4}}{\frac{1}{2}} = \frac{1}{2}$$

7. METHODS (See Above)

9. EXECUTION TIME

9.1 Minimum 58.5 μ sec (Divide Check)

9.2	Maximum	478.5 μ sec
9.3	Average	\approx 460 μ sec
10.	PROGRAM	
10.4	Program Listing	

```

/DEC-08-FMCA-LA
/SIGNED SINGLE PRECISION DIVIDE SUBROUTINE
/CALLING SEQUENCE:
/      C(AC) CONTAINS HIGH ORDER DIVIDEND
/      JMS DIVIDE
/      LOW ORDER DIVIDEND
/      DIVISOR
/      RETURN: C(AC)=QUOTIENT; REMAINDER IN HDIVND
/IF HIGH ORDER DIVIDEND IS EQUAL TO OR GREATER
/THAN THE DIVISOR; NO DIVISION TAKES PLACE AND C(L)=1

```

/PAGE 1

0200	0000	DIVIDE,	0	
0201	7100		CLL	
0202	7510		SPA	/DIVIDEND<0?
0203	7060		CMA CML	/YES COMPLEMENT AND SET C(L)
0204	3267		UCA HDIVND	/HIGH ORDER DIVIDEND
0205	7420		SNL	
0206	7040		CMA	
0207	3272		UCA SDVND	/SET DIVIDEND SIGN SWITCH
0210	1600		TAU I DIVIDE	/FETCH LOW ORDER DIVIDEND
0211	7430		SZL	
0212	7141		CMA CLL IAC	/YES: COMPLEMENT
0213	3270		UCA LDIVND	/LOW ORDER DIVIDEND
0214	7430		SZL	/CARRY?
0215	2267		ISZ HDIVND	/YES
0216	2200		ISZ DIVIDE	
0217	1600		TAU I DIVIDE	/FETCH DIVISOR
0220	7100		CLL	
0221	7500		SMA	
0222	7061		CMA CML IAC	/NEGATE IT
0223	3271		UCA DIVSOR	/SAVE DIVISOR
0224	7420		SNL	/WAS IT <0?
0225	7040		CMA	/YES: AC=-1
0226	1272		TAU SDVND	
0227	3273		UCA ANSWER	/ANSWER SIGN SWITCH
0230	7100		CLL	
0231	1271		TAU DIVSOR	/COMPARE DIVISOR
0232	1267		TAU HDIVND	/WITH DIVIDEND
0233	2200		ISZ DIVIDE	
0234	7630		SZL CLA	/OVER FLOW?
0235	5600		JMP I DIVIDE	/YES: DIVISOR<DIVIDEND

/PAGE 2

0236 1275
0237 3274
0240 5251

TAU M13
UCA DIVCNT
JMP DV2

/13 SHIFTS

/DIVIDE LOUP

0241 1267
0242 7004
0243 3267
0244 1267
0245 1271
0246 7430
0247 3267
0250 7200
0251 1270
0252 7004
0253 3270
0254 2274
0255 5241
0256 1267
0257 2272
0260 7041
0261 3267
0262 1270
0263 2273
0264 7041
0265 7100
0266 5600

DV3, TAU HDIVND
KAL
UCA HDIVND
TAU HDIVND
TAU DIVSOR
SZL
UCA HDIVND
CLA
DV2, TAU LDIVND
KAL
UCA LDIVND
ISZ DIVCNT
JMP DV3
TAU HDIVND
ISZ SDVND
CMA IAC
UCA HDIVND
TAU LDIVND
ISZ SNSWER
CMA IAC
CLL
JMP I DIVIDE

/DIVIDEND LEFT SHIFT
/COMPARE DIVISOR/DIVIDEND
/REMAINDER AFTER SUBTRACT
/QUOTIENT BITS
/ENTER HERE
/DONE 12?
/NO: CONTINUE
/REMAINDER
/DIVIDEND<0?
/YES
/QUOTIENT
/ANSWER<0?
/YES: NEGATE
/EXIT

0267 0000
0270 0000
0271 0000
0272 0000
0273 0000
0274 0000
0275 7763

HDIVND, 0
LDIVND, 0
DIVSOR, 0
SDVND, 0
SNSWER, 0
DIVCNT, 0
M13, -15

/-13(10)

5

SYMBOL TABLE

DIVCNT 0274
DIVIDE 0200
DIVSOR 0271
DV2 0251
DV3 0241
HDIVND 0267
LDIVND 0270
M13 0275
SDVND 0272
SNSWER 0273

1. Signed Double Precision Multiply, DEC-08-FMDA-D

2. ABSTRACT

This subroutine forms a 46-bit signed product from the 23-bit signed multiplier and multiplicand.

3. REQUIREMENTS

3.1 Storage

This subroutine uses 125 (decimal) memory locations.

4. USAGE

4.2 Calling Sequence

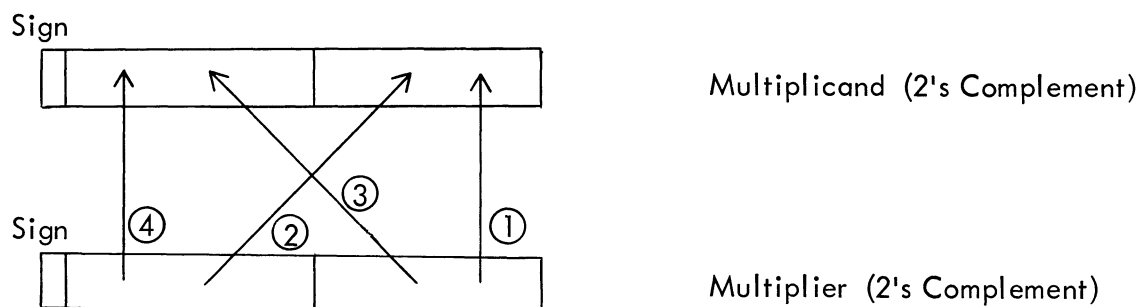
The signed double precision multiply routine is called by an effective JMS DMUL. The two locations following the JMS must contain the address of the high-order multiplicand and the address of the high-order multiplier respectively.

The subroutine will return to the instruction immediately following the latter location, with the most significant portion of the answer in the accumulator. The low order portions of the answer will be in registers (from high to low) B, C, and D.

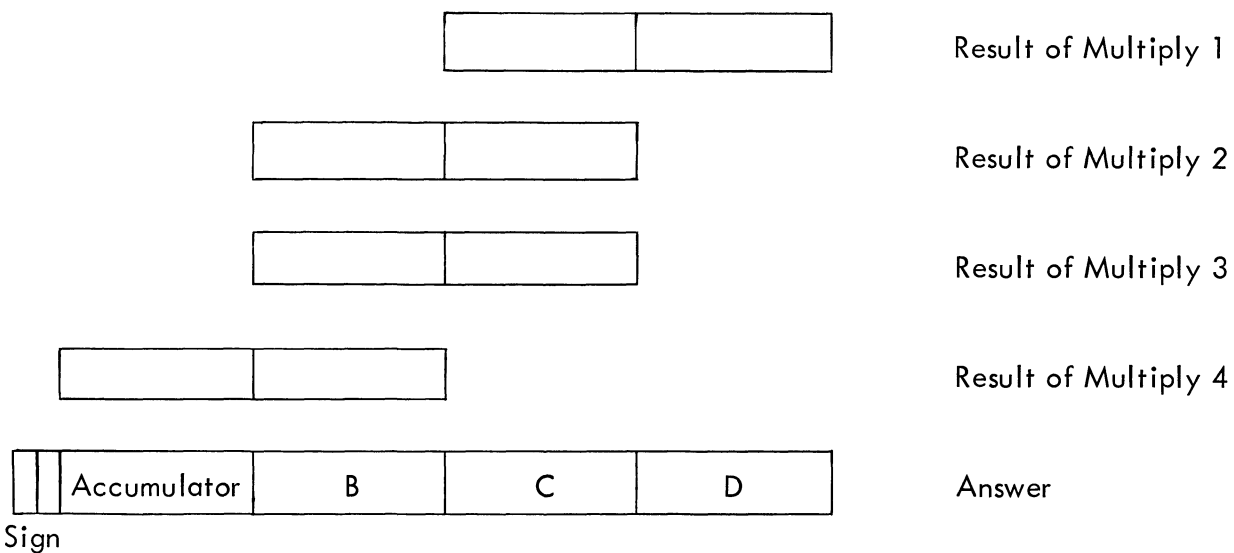
6. DESCRIPTION

6.1 Discussion

The double precision multiply routine calls a single precision multiply routine four times after the absolute values of the multiplier and multiplicand have been taken.



The results are added:



6.2 Examples

To multiply two double precision numbers which are located in registers tagged

X and Y:

```

0400
      JMS  I  DMULTP
      X
      Y
      HLT
X,    0
      0
Y,    0
      0
      DMULTP,  DMUL

```

If X and Y contained:

X	0000	0012	6000	0000
Y	0000	0012	3000	0000

The answers would be:

0000	0000	0000	0144	7200	0000	0000	0000
AC	B	C	D	AC	B	C	D

For further examples see the Double Precision Sine Routine, DEC-08-FMFA formerly Digital-8-16-F.

6.3 Scaling

Since there are 23 magnitude bits in both the multiplier and the multiplicand, the product will contain 46 magnitude bits. These are right justified in the AC and B, C, and D registers. Since the answer is in 2's complement form, the two sign bits are equal (redundant).

The multiply routine may be thought of as an integer multiplication, as a fraction multiplication, or as any combination of these. When the double precision multiply routine is given two 23-bit numbers, it produces a 46-bit product that is right justified. If the scaling is

(XXXX XXX.X) (XXXX XXXX.)

the scaling of the answer will be

XXXX XXXX XXXX XXX.X

The operands and the answer are in 2's complement form. Since only 46 bits of product may be produced and since the answer is right-justified, the two "sign" bits (0 and 1) are redundant.

7. METHODS

7.1 See the Single Precision Multiply Routine write-up, DEC-08-FMBA formerly Digital-8-11-F.

9. EXECUTION TIME

The execution time is a function of the number of 1's in the operands.

The maximum execution time is 1.605 msec. Average time will be around 1.4 msec.

10. PROGRAM

The subroutine occupies approximately one memory page and may be located on any page. The symbolic library tape does not start with an origin setting, but does end with a dollar sign.

10.4 Program Listing

```

/DEC-08-FMDA-LA
/SIGNED DOUBLE PRECISION MULTIPLY ROUTINE
/CALLING SEQUENCE:
/      JMS DMUL
/      ADDRESS OF MULTIPLICAND(HIGH ORDER)
/      ADDRESS OF MULTIPLIER(HIGH ORDER)
/      RETURN, HIGH ORDER PRODUCT IN AC
/      NEXT HIGH TO LOW IN B,C,D

```

```

/PAGE 1
DMUL,

```

0200	0000	0	
0201	7300	CLL CLA	
0202	1333	TAD RESI	/-2
0203	3332	DCA SIGNSW	/SET SIGN SWITCH
0204	4306	JMS TSGN	/FETCH AND SET SIGN
0205	1337	TAD MLTH	/RESULT IN MLTH,MLTL
0206	3334	DCA MULIH	/HIGH ORDER MULTIPLICAND
0207	1336	TAD MLTL	
0210	3335	DCA MULIL	/LOW ORDER MULTIPLICAND
0211	4306	JMS TSGN	/FETCH AND SET SIGN
0212	1335	TAD MULIL	/LOW ORDER MULTIPLICAND
0213	3301	DCA MP2	
0214	1336	TAD MLTL	/LOW ORDER MULTIPLIER
0215	4344	JMS MP4	/MULTIPLY
0216	3343	DCA D	/LOW ORDER
0217	1373	TAD MP5	
0220	3342	DCA C	/HIGH ORDER
0221	1334	TAD MULIH	/HIGH ORDER MULTIPLICAND
0222	3301	DCA MP2	
0223	1336	TAD MLTL	/LOW ORDER MULTIPLIER
0224	4344	JMS MP4	/MULTIPLY
0225	1342	TAD C	
0226	3342	DCA C	
0227	7004	RAL	/GET CARRY
0230	1373	TAD MP5	
0231	3341	DCA B	
0232	7004	RAL	/GET CARRY
0233	3340	DCA A	
0234	1335	TAD MULIL	/LOW ORDER MULTIPLICAND
0235	3301	DCA MP2	
0236	1337	TAD MLTH	/HIGH ORDER MULTIPLIER
0237	4344	JMS MP4	/MULTIPLY
0240	1342	TAD C	
0241	3342	DCA C	/ADD

/PAGE 2

0242	7004	RAL	/GET CARRY
0243	1373	TAD MP5	
0244	1341	TAD B	
0245	3341	DCA B	
0246	7004	RAL	/GET CARRY
0247	1340	TAD A	
0250	3340	DCA A	/ADD
0251	1334	TAD MLTH	/HIGH ORDER MULTIPLICAND
0252	3301	DCA MP2	
0253	1337	TAD MLTH	/HIGH ORDER MULTIPLIER
0254	4344	JMS MP4	
0255	1341	TAD B	
0256	3341	DCA B	
0257	7004	RAL	
0260	1373	TAD MP5	
0261	1340	TAD A	
0262	2332	ISZ SIGNSW	/ANSWER <0??
0263	5600	JMP I DMUL	/NO: EXIT
0264	3340	DCA A	/YES
0265	1343	TAD D	
0266	7141	CMA CLL IAC	/NEGATE
0267	3343	DCA D	
0270	1342	TAD C	/NEGATE
0271	4301	JMS COM	
0272	3342	DCA C	
0273	1341	TAD B	
0274	4301	JMS COM	/NEGATE
0275	3341	DCA B	
0276	1340	TAD A	
0277	4301	JMS COM	
0300	5600	JMP I DMUL	/EXIT
0301	0000	MP2,	
0302	7040	COM,	
0303	7430		
0304	7101		
0305	5701		

		/PAGE 3		
		MP1,		
		TSIGN,	0	
0306	0000		TAD I DMUL	/FETCH ADDRESS
0307	1600		DCA ADDR5	
0310	3340		TAD I ADDR5	/FETCH HIGH ORDER
0311	1/40		CLL	
0312	7100		SPA	/IS IT <0?
0313	7510		CMA CML	/YES: COMPLEMENT, SET LINK
0314	7060		DCA MLTH	
0315	3337		ISZ ADDR5	
0316	2340		TAD I ADDR5	/FETCH LOW ORDER
0317	1/40		SZL	/WAS IT <0?
0320	7430		ISZ SIGNSW	/YES, ADD 1 TO SWITCH
0321	2332		NOP	
0322	7000		SZL	
0323	7430		CMA CLL IAC	/COMPLEMENT, CLEAR LINK
0324	7141		DCA MLTL	
0325	3336		SZL	/CARRY?
0326	7430		ISZ MLTH	/YES
0327	2337		ISZ DMUL	
0330	2200		JMP I TSIGN	/EXIT ROUTINE
0331	5706			
0332	0000	SIGNSW,	0	
0333	7776	RESI,	-2	
0334	0000	MULIH,	0	
0335	0000	MULIL,	0	
0336	0000	MLTL,	0	
0337	0000	MLTH,	0	
		ADDR5,		
0340	0000	A,	0	
0341	0000	B,	0	
0342	0000	C,	0	
0343	0000	D,	0	

/PAGE 4

0344	0000	MP4,	0	/UNSIGNED MULTIPLY
0345	3306		DCA MP1	
0346	3373		DCA MP5	
0347	1374		TAD M12	/COUNT 12 BITS
0350	3372		DCA MP3	
0351	7100		CLL	
0352	1306		TAD MP1	/CARRY GOES INTO
0353	7010		RAR	/LEFT OF MP1
0354	3306		DCA MP1	/TEST MULTIPLIER BIT
0355	1373		TAD MP5	
0356	7420		SNL	/A 1?
0357	5362		JMP ,+3	/NO: DON'T ADD
0360	7100		CLL	/YES: ADD
0361	1301		TAD MP2	
0362	7010		RAR	
0363	3373		DCA MP5	
0364	2372		ISZ MP3	/DONE 12 BITS?
0365	5352		JMP MP4+6	/NO: CARRY IS IN C(L)
0366	1306		TAD MP1	/YES: DONE
0367	7010		RAR	
0370	7100		CLL	
0371	5744		JMP I MP4	/EXIT
0372	0000	MP3,	0	
0373	0000	MP5,	0	
0374	7764	M12,	-14	

PAUSE

SYMBOL TABLE

A	0340
ADDRS	0340
B	0341
C	0342
CUM	0301
D	0343
DMUL	0200
MLTH	0337
MLTL	0336
MP1	0306
MP2	0301
MP3	0372
MP4	0344
MP5	0373
MULIH	0334
MULIL	0335
M12	0374
RESI	0333
SIGNSW	0332
TSIGN	0306

1. Double Precision Signed Divide Subroutine, DEC-08-FMEA-D.

2. ABSTRACT

The Double-Precision Divide Subroutine will divide a 24-bit signed divisor into a 48-bit signed dividend to produce a 24-bit signed quotient and an unsigned remainder.

3. REQUIREMENTS

3.1 Storage

This subroutine requires 105 (decimal) memory locations. It is provided in two forms: a binary tape assembled with an origin of 0200 and, a symbolic tape with no origin setting.

4. USAGE

4.1 Loading

The subroutine is loaded with the Binary Loader (Digital-8-2-U). The symbolic is either assembled with the user program or separately with the proper origin setting.

4.2 Calling Sequence

The subroutine is called with an effective JMS DUBDIV with the address of the high-order word of the dividend (address of the dividend) in the accumulator, followed by the address of the high-order word of the divisor (address of the divisor). Control returns to the calling program at the address of the JMS plus 2.

	TAD	HIGH	
	JMS	I	DDIVP
	LOW		
	HLT		
	:		
	:		
DDIVP,	DUBDIV		
HIGH,	· + 1	/ADDRESS OF DIVIDEND	
	0	/DIVIDEND	
	0		
	0		
	0		
LOW,	0	/DIVISOR	
	0		

The high-order quotient is returned in the accumulator and the remaining bits of the answer are found as follows:

C(DIVND4) = Low-order quotient
C(DIVND1) = High-order remainder
C(DIVND2) = Low-order remainder

The quotient is signed, while the remainder is left unsigned.

4.5 Errors in Usage

Since the division process may be represented as:

$$\frac{\text{Dividend}}{\text{Divisor}} = \text{Quotient, Remainder}$$

such that:

$$\text{Dividend} = (\text{Quotient}) (\text{Divisor}) + \text{Remainder}$$

It is possible to specify a dividend and a divisor such that the quotient cannot be contained within the word size (in this case, 23 bits). If this is true, the results will be nonvalid. This condition is not tested by the Double-Precision Divide Subroutine. (For a more complete description, see DEC-08-FMCA, formerly Digital-8-12-F, Section 4.5.)

5. RESTRICTIONS

See Section 4.5.

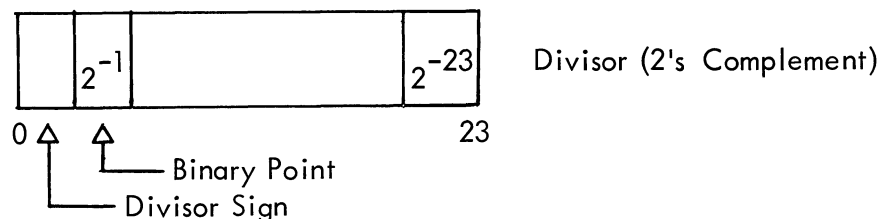
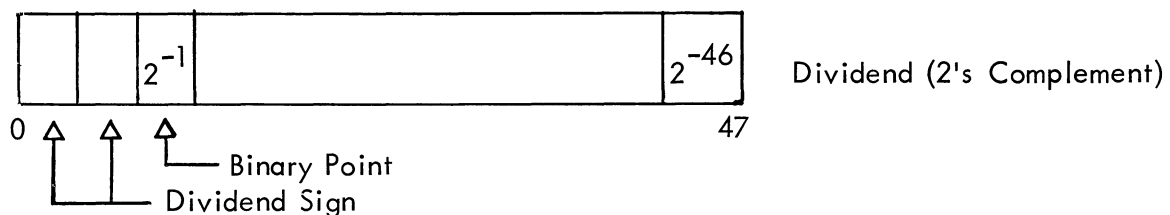
6. DESCRIPTION

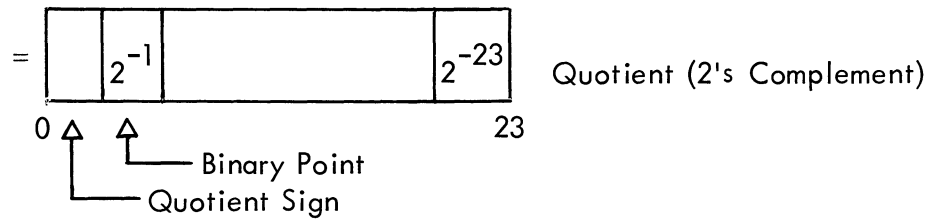
6.1 Discussion

See DEC-08-FMCA, Section 6.1.

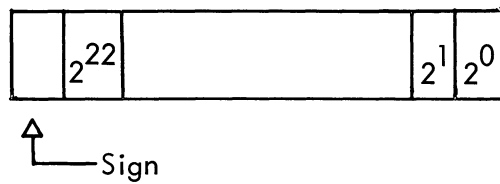
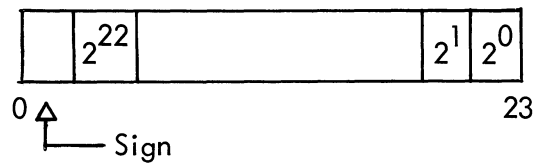
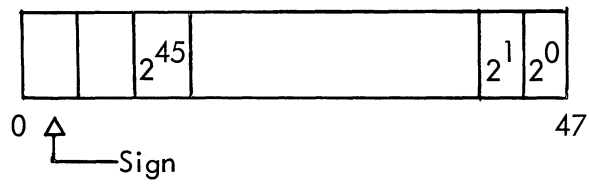
6.3 Scaling

The Double-Precision Divide Subroutine is scaled analogous to the scaling of the Double-Precision Multiply Subroutine (DEC-08-FMDA, formerly Digital-8-13-F). It may be considered either an integer divide or a fractional divide.





or



9. EXECUTION TIME

9.1	Minimum	1.424 msec
9.2	Maximum	1.705 msec
9.3	Average	1.65 msec

10. PROGRAM

10.4 Program Listing

```

/DEC-08-FMEA-LA
/DOUBLE PRECISION DIVIDE SUBROUTINE
/CALLING SEQUENCE:
/      C(AC)=ADDRESS OF HIGH ORDER DIVIDEND
/      JMS DUBDIV
/      ADDRESS OF HIGH ORDER DIVISOR
/      RETURN: C(AC)=HIGH ORDER QUOTIENT
/              C(DIVND4)=LOW ORDER QUOTIENT
/              C(DIVND1)=HIGH ORDER REMAINDER
/              C(DIVND2)=LOW ORDER REMAINDER
/IF DIVISOR<DIVIDEND; RESULTS UNSPECIFIED

```

/PAGE 1

0200	0000	DUBDIV, 0	
0201	3331	DCA ADDR5	/DIVIDEND ADDRESS
0202	1343	TAD REST	/-2
0203	3340	DCA SIGNSW	/SET SIGN SWITCH
0204	1731	TAD I ADDR5	/HIGH-ORDER DIVIDEND
0205	3332	DCA DIVND1	
0206	2331	ISZ ADDR5	
0207	1731	TAD I ADDR5	/DIVIDEND
0210	3333	DCA DIVND2	
0211	2331	ISZ ADDR5	
0212	1731	TAD I ADDR5	/DIVIDEND
0213	3334	DCA DIVND3	
0214	2331	ISZ ADDR5	
0215	1731	TAD I ADDR5	/DIVIDEND
0216	3335	DCA DIVND4	
0217	1332	TAD DIVND1	/DIVIDEND<0?
0220	7700	SMA CLA	
0221	5237	JMP DIVG01	/NO: CONTINUE
0222	2340	ISZ SIGNSW	/YES: ADD 1 TO SWITCH
0223	1335	TAD DIVND4	
0224	7141	CMA IAC CLL	/NEGATE DIVIDEND
0225	3335	DCA DIVND4	
0226	1334	TAD DIVND3	
0227	4344	JMS COM	
0230	3334	DCA DIVND3	
0231	1333	TAD DIVND2	
0232	4344	JMS COM	
0233	3333	DCA DIVND2	
0234	1332	TAD DIVND1	
0235	4344	JMS COM	
0236	3332	DCA DIVND1	

/PAGE 2

0237	1600	/FETCH DIVISOR	
0240	2200	DIVG01, TAD I DUBDIV	
0241	3331	ISZ DUBDIV	
0242	1/31	DCA ADDR5	/ADDRESS OF DIVISOR
0243	7100	TAD I ADDR5	/HIGH ORDER DIVISOR
0244	7500	CLL	
0245	7060	SMA	/DIVISOR>0?
0246	3336	CMA CML	/YES:NEGATE AND SET C(L)
0247	2331	DCA HDIVSR	
0250	1/31	ISZ ADDR5	
0251	7420	TAD I ADDR5	/LOW ORDER DIVISOR
0252	2340	SNL	
0253	7000	ISZ SIGNSW	/ADD 1 TO SIGN SWITCH
0254	7430	NUP	
0255	7141	SZL	
0256	3337	CMA IAC CLL	/COMPLEMENT
0257	7430	DCA LDIVSR	/LOW ORDER DIVISOR
0260	2330	SZL	/CARRY?
0261	1342	ISZ HDIVSR	/YES
0262	3341	TAD M25	
0263	7100	DCA DIVCNT	/SET DIVIDE COUNT=24
0264	5307	CLL	
		JMP DIV2	

/PAGE 3

0265	1333	DIV3,	TAD DIVND2	/SHIFT HIGH DIVIDEND
0266	7004		RAL	/LEFT
0267	3333		DCA DIVND2	
0270	1332		TAD DIVND1	
0271	7004		RAL	
0272	3332		DCA DIVND1	
0273	1333		TAD DIVND2	/COMPARE DIVISOR;
0274	1337		TAD LUIVSR	/WITH DIVISOR
0275	3331		DCA ADDR5	
0276	7004		RAL	/GET CARRY
0277	1332		TAD DIVND1	
0300	1336		TAD HDIVSR	
0301	7420		SNL	
0302	5306		JMP DIV2_1	
0303	3332		DCA DIVND1	
0304	1331		TAD ADDR5	
0305	3333		DCA DIVND2	
0306	7200		CLA	
0307	1335	DIV2,	TAD DIVND4	/ROTATE LOW ORDER
0310	7004		RAL	/WORDS LEFT
0311	3335		DCA DIVND4	
0312	1334		TAD DIVND3	/QUOTIENT BITS
0313	7004		RAL	
0314	3334		DCA DIVND3	/ENTER FROM C(L)
0315	2341		ISZ DIVCN1	/DONE 24?
0316	5265		JMP DIV3	/NO: CONTINUE
0317	2340		ISZ SIGNSW	/ANSWER<0?
0320	5327		JMP OUT	/NO: EXIT
0321	1335		TAD DIVND4	/YES
0322	7141		CMA CLL IAC	
0323	3335		DCA DIVND4	
0324	1334		TAD DIVND3	
0325	4344		JMS COM	
0326	5600		JMP I DUBUIV	
0327	1334	OUT,	TAD DIVND3	
0330	5600		JMP I DUBUIV	

/PAGE 4

0331	0000	ADDRS,	0	
0332	0000	DIVND1,	0	
0333	0000	DIVND2,	0	
0334	0000	DIVND3,	0	
0335	0000	DIVND4,	0	
0336	0000	HUIVSR,	0	
0337	0000	LUIVSR,	0	
0340	0000	SIGNSW,	0	
0341	0000	DIVCN1,	0	
0342	7747	M25,	-31	/-25(10)
0343	7776	RES1,	-2	
0344	0000	COM,	0	
0345	7040		CMA	
0346	7430		S&L	
0347	7101		CLL IAC	
0350	5744		JMP I COM	
PAUSE				

SYMBOL TABLE

ADDRS	0331
COM	0344
DIVCN1	0341
DIVG01	0237
DIVND1	0332
DIVND2	0333
DIVND3	0334
DIVND4	0335
DIV2	0307
DIV3	0265
DUBDIV	0200
HUIVSR	0336
LUIVSR	0337
M25	0342
OUT	0327
RES1	0343
SIGNSW	0340

12. REFERENCES

See DEC-08-FMDA, formerly Digital-8-13-F.

1. Double-Precision Sine Subroutine, DEC-08-FMFB-D.

2. ABSTRACT

The Double-Precision Sine Subroutine will evaluate the function $\text{Sin}(X)$ for $-4 < X < 4$ (X is in radians). The argument is a double-precision word, 2 bits representing the integer part and 21 bits representing the fractional part. The result is a 23-bit signed fraction $-1 < \text{Sin}(X) < 1$.

3. REQUIREMENTS

3.1 Storage

This subroutine uses 248 (decimal) memory locations.

3.2 Subprograms and/or Subroutines

The Double-Precision Multiply Subroutine (DEC-08-FMDA, formerly Digital-8-13-F) or EAE Version (Digital-8-23-F).

4. USAGE

4.2 Calling Sequence

The Double-Precision Sine Subroutine is called by an effective JMS DSIN followed by the address of the high-order word of the argument. Control returns to the calling program at the address of argument address + 1 with $C(AC) = 0$, $C(L) = 0$ and with the answer in registers ARG, ARG + 1. For example:

	JMS	I	DSINP
	ARGMNT		
	HLT		
	:		
	:		
DSINP,	DSIN		
ARGMNT,	1000		
	0000		

6. DESCRIPTION

6.1 Discussion

The input to the sine subroutine is considered to be in radians within the range $-4 < X < 4$. The subroutine is able to call itself recursively and does so when reducing the range of the argument to the first quadrant. The following identities are used:

if	$X = 0$	$\text{Sin}(0) = 0$
if	$X < 0$,	$\text{Sin}(-X) = -\text{Sin}(X)$ (recursive call)
if	$X < \pi$,	$\text{Sin}(X) = -\text{Sin}(X - \pi)$ (recursive call)

if $X > \pi/2$ $\text{Sin}(X) = -\text{Sin}(X - \pi)$ (recursive call)
 if $X = \pi/2$ $\text{Sin}(\pi/2) = 1$
 for $0 < X < \pi/2$,

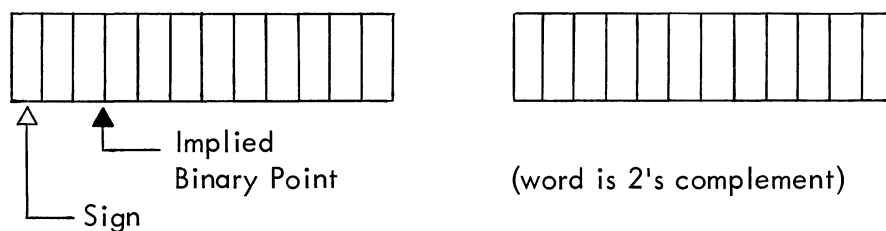
$F = \frac{2X}{\pi}$ so that $0 < F < 1$, then:

$$\text{Sin}(X) = F(C_1 + C_3F^2 + C_5F^4 + C_7F^6 + C_9F^8)$$

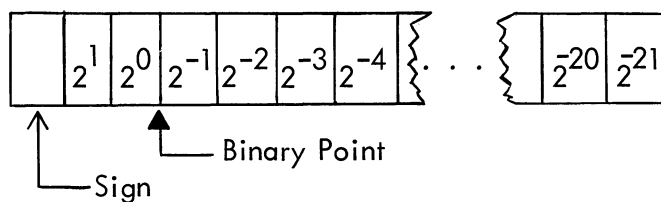
6.3

Scaling

The scaling for the argument is:



The binary weightings of the argument may be represented as:



Thus, 1.5 radians would be:

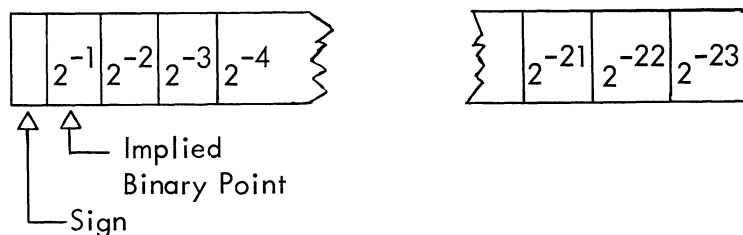
001 100 000 000 000 000 000 000

and

-1.5 radians would be:

110 100 000 000 000 000 000 000

The answer is a 23-bit signed fraction (2's complement) with the following binary weightings:



Thus if the answer were 0.75(10), it would appear as follows:

ARG	011	000	000	000
ARG+1	000	000	000	000

If the answer were -0.75(10), it would appear as:

ARG	101	000	000	000
ARG+1	000	000	000	000

7. METHODS

7.2 Algorithm

See Section 6.1.

9. EXECUTION TIME

9.1	Minimum	When the argument is a multiple of π : 70 μ sec
9.2	Maximum	Without EAE: 10.6 msec With EAE: 2.78 msec
9.3	Average	Without EAE: 10.4 msec With EAE: 2.6 msec.

10. PROGRAM

10.1 Core Map

The Double-Precision Sine Subroutine, as listed, was assembled starting at 0400 (8). It assumes that the Double-Precision Multiply Subroutine (DEC-08-FMDA, formerly Digital-8-13-F) is in core starting at 0200. If the multiply subroutine is placed elsewhere, the pointers on page 1 of the program should be changed.

10.4

Program Listing

```

                                /DEC-08-FMFB-PA
                                /DOUBLE PRECISION SINE
                                /POINTERS TO DEC-08-FMUA
0200    DMUL=200
0341    B=341
0342    C=342
0400    *400

0400    0000    DSIN,    0
0401    1600    TAD I DSIN        /ADDRESS OF ARGUMENT
0402    3351    DCA TEMP
0403    1751    TAD I TEMP        /HIGH ORDER
0404    3347    DCA X2
0405    2351    ISZ TEMP
0406    1751    TAD I TEMP        /LOW ORDER
0407    3350    DCA X2+1
0410    2200    ISZ DSIN        /FIX EXIT
0411    1200    TAD DSIN        /SAVE ON PUSHDOWN LIST
0412    3763    DCA I PUSH
0413    2363    ISZ PUSH
0414    1347    TAD X2 /CHECK FOR ZERO
0415    7640    SZA CLA
0416    5233    JMP NEG
0417    1350    TAD X2+1
0420    7640    SZA CLA
0421    5233    JMP NEG /NO
0422    7200    CLA
0423    3754    DCA I PNT3        /SIN(0)=0
0424    3755    DCA I PNT3+1
0425    7240    XIT1,    CLA CMA /EXIT
0426    1363    TAD PUSH
0427    3363    DCA PUSH
0430    1763    TAD I PUSH
0431    3351    DCA TEMP
0432    5751    JMP I TEMP
0433    1347    NEG,    TAD X2 /CHECK FOR NEGATIVE X
0434    7700    SMA CLA
0435    5261    JMP POS
0436    1350    TAD X2+1        /SIN(-X)=-SIN(X)
0437    7141    CLL CMA IAC
0440    3350    DCA X2+1
0441    1347    TAD X2
0442    7040    CMA
0443    7430    SZL
0444    7001    IAC
0445    3347    DCA X2

```

/DEC-08-FMFB-PA
/PAGE 2

0446	4200	JMS DSIN	/RECURSIVE CALL FOR SINE
0447	0547	X2	
0450	1755	XIT2, TAD I PNT3+1	/NEGATE THE ANSWER
0451	7141	CLL CMA IAC	
0452	3755	DCA I PNT3+1	
0453	1754	TAD I PNT3	
0454	7040	CMA	
0455	7430	S&L	
0456	7001	IAC	
0457	3754	DCA I PNT3	
0460	5225	JMP XIT1	
0461	7100	POS, CLL /IS X<PI?	
0462	1350	TAD X2+1	
0463	1360	TAD MPI+1	
0464	3351	DCA TEMP	
0465	7004	RAL /CARRY	
0466	1347	TAD X2	
0467	1357	TAD MPI	
0470	7510	SPA	
0471	5300	JMP PCHK	
0472	3347	DCA X2 /SIN(X)=-SIN(X-PI)	
0473	1351	TAD TEMP	
0474	3350	DCA X2+1	
0475	4200	JMS DSIN	
0476	0547	X2	
0477	5250	JMP XIT2	
0500	7300	PCHK, CLA CLL /IS X<PI/2?	
0501	1350	TAD X2+1	
0502	1362	TAD MPI0+1	
0503	3351	DCA TEMP	
0504	7004	RAL	
0505	1347	TAD X2	
0506	1361	TAD MPI0	
0507	7510	SPA	
0510	5337	JMP ALG	
0511	7440	S&A	
0512	5324	JMP P2NG	
0513	1351	TAD TEMP	
0514	7440	S&A	
0515	5324	JMP P2NG	
0516	7140	CMA CLL /SIN(PI/2)=1	
0517	7010	RAR	
0520	3754	DCA I PNT3	
0521	7040	CMA	
0522	3755	DCA I PNT3+1	
0523	5225	JMP XIT1	

```

/DEC-08-FMFB-PA
/PAGE 3
0524 7300 P2NG, CLL CLA
0525 1350 TAD X2+1
0526 1360 TAD MPI+1 /SIN(X)=-SIN(X-PI)
0527 3350 DCA X2+1
0530 7004 RAL
0531 1347 TAD X2
0532 1357 TAD MPI
0533 3347 DCA X2
0534 4200 JMS DSIN /RECURSIVE CALL FOR SINE
0535 0547 X2
0536 5250 JMP XIT2
0537 7200 ALG, CLA /ALIGN SCALING FOR ALGORITHM
0540 1350 TAD X2+1
0541 7104 CLL RAL
0542 3753 DCA I PNT2+1
0543 1347 TAD X2
0544 7004 RAL
0545 3752 DCA I PNT2
0546 5756 JMP I PNT4

/SYMBOLS AND CONSTANTS FOR THIS PAGE
0547 0000 X2, 0
0550 0000 0
0551 0000 TEMP, 0
0552 0743 PNT2, X
0553 0744 X+1
0554 0741 PNT3, ARG
0555 0742 ARG+1
0556 0600 PNT4, DALG
0557 4667 MPI, 4667 /-(PI)
0560 4023 4023
0561 6333 MPI0, 6333 /-(PI/2)
0562 6012 6012
0563 0564 PUSH, PUSH+1 / POINTER FOR PUSHDOWN LIST

```

```

/DEC-08-FMFB-PA
/PAGE 4
*DSIN+200
0600 0600
0600 4736
0601 0743
0602 0755
0603 4277
0604 4277
0605 4312
0606 0743
0607 4736
0610 0743
0611 0743
0612 4277
0613 4312
0614 0737
0615 1353
0616 3345
0617 1354
0620 3346
0621 3341
0622 3342
0623 7100
0624 1745
0625 1342
0626 3342
0627 2345
0630 7004
0631 1341
0632 1745
0633 3341
0634 2345
0635 4736
0636 0741
0637 0737
0640 4277
0641 4312
0642 0741
0643 2346
0644 5223
0645 7100
0646 1341
0647 7510
0650 7020
0651 7010
0652 3341
0653 1342
0654 7010
0655 3342

DALG, JMS I DMTG /FORM (2/PI)*ARG
X
TOPI
JMS SCAL /GET RID OF EXTRA SIGN BIT
JMS SCAL /SCALING = 0 NOW
JMS ROUND
X
JMS I DMTG /GET X*X
X
X
JMS SCAL /GET RID OF EXTRA SIGN BIT
JMS ROUND
XSQR
TAD FYX /INI
DCA PNT / T
TAD FOUR / I
DCA CHK / A
DCA ARG / L
DCA ARG+1 / IZE
LOOP, CLL
TAD I PNT
TAD ARG+1
DCA ARG+1
ISZ PNT
RAL
TAD ARG
TAD I PNT
DCA ARG
ISZ PNT /INCREMENT POINTER FOR NEXT
JMS I DMTG
ARG
XSQR
JMS SCAL /GET RID OF SIGN BIT
JMS ROUND
ARG
ISZ CHK
JMP LOOP
CLL
TAD ARG /SHIFT ARG 1 PLACE
SPA
CML
RAR
DCA ARG
TAD ARG+1
RAR
DCA ARG+1

```

0656	7100	CLL	/ADD IN LAST CONSTANT
0657	1360	TAD C1+1	
0660	1342	TAD ARG+1	
0661	3342	DCA ARG+1	
0662	7004	RAL	/CARRY
0663	1341	TAD ARG	
0664	1357	TAD C1	
0665	3341	DCA ARG	
0666	4736	JMS I DMTG	
0667	0741	ARG	
0670	0743	X	
0671	4277	JMS SCAL	/PUT SCALING BACK TO ZERO
0672	4277	JMS SCAL	/GET RID OF SIGN BIT
0673	4312	JMS ROUND	
0674	0741	ARG	
0675	5676	JMP I OUT	
0676	0425	OUT, XIT1	
0677	0000	SCAL, 0	/ROUTINE TO ADJUST SCALING
0700	3350	DCA TEM2	
0701	1752	TAD I CTG	
0702	7104	CLL RAL	
0703	3752	DCA I CTG	
0704	1751	TAD I BTG	
0705	7004	RAL	
0706	3751	DCA I BTG	
0707	1350	TAD TEM2	
0710	7004	RAL	
0711	5677	JMP I SCAL	
0712	0000	ROUND, 0	
0713	3347	DCA TEM1	
0714	1712	TAD I ROUND	/ADDRESS OF HIGH ORDER
0715	2312	ISZ ROUND	
0716	3350	DCA TEM2	
0717	1347	TAD TEM1	
0720	3750	DCA I TEM2	
0721	1350	TAD TEM2	
0722	7001	IAC	
0723	3347	DCA TEM1	
0724	1751	TAD I BTG	
0725	3747	DCA I TEM1	
0726	1752	TAD I CTG	
0727	7710	SPA CLA /BIT 0=1??	
0730	5712	JMP I ROUND	/NO: EXIT
0731	2747	ISZ I TEM1	/YES: ROUND
0732	5712	JMP I ROUND	
0733	2750	ISZ I TEM2	/CARRY
0734	7000	NOP	/RETURN SKIP OR NOT!!
0735	5712	JMP I ROUND	

		/DEC-08-FMFB-PA	
		/PAGE 6	
		/SYMBOLS AND CONSTANTS	
0736	0200	DMTG,	DMUL
0737	0000	XSQR,	0
0740	0000		0
0741	0000	ARG,	0
0742	0000		0
0743	0000	X,	0
0744	0000		0
0745	0000	PNT,	0
0746	0000	CHK,	0
0747	0000	TEM1,	0
0750	0000	TEM2,	0
0751	0341	BTG,	B
0752	0342	CTG,	C
0753	0761	FYX,	C9
0754	7774	FOUR,	-4
0755	2427	TOP1,	2427
0756	6303		6303
0757	3110	C1,	3110
0760	3755		3755
0761	2367	C9,	2367
0762	0000		0000
0763	3331	C7,	3331
0764	7766		7766
0765	1505	C5,	1505
0766	0243		0243
0767	0420	C3,	0420
0770	5325		5325
			\$

/2/PI

/C3-C9 STORED IN BACKWARDS ORDER

SYMBOL TABLE

ALG	0537
ARG	0741
B	0341
BTG	0751
C	0342
CHK	0746
CTG	0752
C1	0757
C3	0767
C5	0765
C7	0763
C9	0761
DALG	0600
DMTG	0736
DMUL	0200
DSIN	0400
FOUR	0754
FYX	0753
LOOP	0623
MPI	0557
MPIO	0561
NEG	0433
OUT	0676
PCHK	0500
PNT	0745
PNT2	0552
PNT3	0554
PNT4	0556
POS	0461
PUSH	0563
P2NG	0524
ROUND	0712
SCAL	0677
TEMP	0551
TEM1	0747
TEM2	0750
TUPI	0755
X	0743
XIT1	0425
XIT2	0450
XSQR	0737
X2	0547

1. Double-Precision Cosine Subroutine, DEC-08-FMGB-D

2 ABSTRACT

This subroutine will form the cosine of a double-precision argument (in radians). The input range is $-4 < X < 4$.

3. REQUIREMENTS

3.1 Storage

This subroutine requires 64 (decimal) memory locations.

3.2 Subprograms and/or Subroutines

This subroutine requires the Double-Precision Sine Subroutine (DEC-08-FMFB-D). The symbolic tape contains definitions that are used as intercommunication registers to the sine subroutine. If the sine subroutine is moved, these "pointers" must be changed.

3.3 Equipment

Standard PDP-8.

4. USAGE

4.1 Loading

The library tape that is supplied is a symbolic tape. It begins with an absolute origin setting and ends with a dollar sign. The binary tape produced by assembling this tape, or the binary tape produced by assembling this tape with other tapes, is loaded with the Binary Loader.

4.2 Calling Sequence

The Double-Precision Cosine Subroutine is called in a manner that is identical to the way in which the Double-Precision Sine Subroutine is called. For more complete information, see DEC-08-FMFB-D

5. RESTRICTIONS

See DEC-08-FMFB-D

6. DESCRIPTION

6.1 Discussion

The Double-Precision Cosine Subroutine uses the following identities:

If $X < 0$; $\text{COS}(-X) = \text{COS}(X)$

Then $\text{SIN}(\pi/2 - X) = \text{COS}(X)$

range. This insures that the argument presented to the sine subroutine is in the proper

6.3 Scaling

See DEC-08-FMFB-D

7. METHODS

See DEC-08-FMFB-D

8. FORMAT

See DEC-08-FMFB-D

9. EXECUTION TIME

9.1 Minimum

The minimum time occurs when the argument is 0. In this case, time = 55.5 μ sec.

9.3 Average

In general, the Double-Precision Cosine Subroutine takes from 75 μ sec to 93 μ sec longer than the Double-Precision Sine Subroutine (see DEC-08-FMFB-D).

10. PROGRAM

10.4 Program Listing

```

/DEC-08-FMGB-PA
/DOUBLE PRECISION COSINE SUBROUTINE
/CALLS DEC-08-FMFA
/POINTERS TO DEC-08-FMFB FOLLOW
ARG=741
DSIN=400

```

0741		
0400		
1000	1000	
1000	0000	*1000
1001	1600	DCOS, 0
1002	3262	TAD I DCOS /FETCH ADDRESS OF
1003	1662	DCA ADDRSS /ARGUMENT
1004	3256	TAD I ADDRSS /FETCH HIGH ORDER
1005	2262	DCA EX /ARGUMENT
1006	1662	ISZ ADDRSS /INCREMENT ADDRESS POINTER
1007	3257	TAD I ADDRSS /FETCH LOW ORDER
1010	1256	DCA EX+1 /ARGUMENT
1011	7640	TAD EX /IS ARGUMENT EQUAL
1012	5224	SZA CLA /TO ZERO
1013	1257	JMP TSIGNN /NO: TEST THE SIGN
1014	7640	TAD EX+1 /TEST LOW ORDER BITS
1015	5224	SZA CLA /FOR ZERO
1016	7040	JMP TSIGNN /NOT EQUAL TO ZERO
1017	7010	CMA
1020	3660	RAR
1021	7040	DCA I ARGPNT
1022	3661	CMA
1023	5254	DCA I ARGPNT+1 /SET ANSWER TO 1
1024	1256	JMP EXIT
1025	7700	TSIGNN, TAD EX /SEE IF X>0
1026	5237	SMA CLA
1027	1257	JMP ARGPOS /ARGUMENT IS >0
1030	7141	TAD EX+1 /ARGUMENT IS <0
1031	3257	CLL CMA IAC /NEGATE IT
1032	1256	DCA EX+1
1033	7040	TAD EX
1034	7430	CMA
1035	7001	SZL
1036	3256	IAC
		DCA EX

1037	7300	ARGPOS, CLL CLA	
1040	1257	TAD EX+1	
1041	7041	CMA IAC	
1042	1265	TAD PIOT+1	/SUBTRACT X FROM
1043	3257	DCA EX+1	/PI/2
1044	1256	TAD EX	
1045	7040	CMA	
1046	7430	SZL	
1047	7001	IAC	
1050	1264	TAD PIOT	
1051	3256	DCA EX	
1052	4663	JMS I DSINPT	/CALL SINE SUBROUTINE
1053	1056	EX	/ARGUMENT ADDRESS
1054	2200	EXIT, ISZ DCOS	/RETURN TO CALL+1
1055	5600	JMP I DCOS	/ANSWER IN ARG,ARG+1
1056	0000	EX, 0	
1057	0000	0	
1060	0741	ARGPNT, ARG	
1061	0742	ARG+1	
1062	0000	ADDRSS, 0	
1063	0400	DSINPT, DSIN	
1064	1444	PIOT, 1444	
1065	1767	1767	

PAUSE

SYMBOL TABLE

ADDRSS	1062
ARG	0741
ARGPNT	1060
ARGPOS	1037
DCOS	1000
DSIN	0400
DSINPT	1063
EX	1056
EXIT	1054
PIOT	1064
TSIGNN	1024

12. REFERENCES

12.1 Other Library Programs

See Digital-8-16-F for further explanation of the calling sequence, timing, scaling, and algorithm.

1. Four-Word Floating-Point Package, DEC-08-FMHA-D.

2. ABSTRACT

This program is almost identical to the 3-word Floating-Point Package (Digital-8-5-S) except that accuracy is carried to 35 bits, and 4 12-bit words are used for storage.

3. REQUIREMENTS

3.1 Storage

This program occupies registers 7; 40-61; 5600-7577 (octal).

4. USAGE

4.1 Loading

Binary Loader (Digital-8-2-U) or DECtape System.

4.2 Calling Sequence

Identical to Digital-8-5-S.

5. RESTRICTIONS

See Digital-8-5-S.

6. DESCRIPTION

The floating accumulator resides in memory locations 44, 45, 46, and 47. The instructions FGET, FPUT use 4-word arguments (11-bit exponent + sign; 35-bit mantissa + sign). The 4-word package contains all operations except for square root (0002) and square (0001).

7. METHODS

See Digital-8-5-S.

8. FORMAT (Not Applicable)

9. EXECUTION TIME

9.3 Average

Execution times are very difficult to estimate as they greatly depend upon the data on which the floating-point package is operating. Generally speaking:

FADD	=	382 μ sec + 42(N) where N is the number of shifts to align binary points.
FSUB	=	FADD time + 42 μ sec
FDIV	=	3.4 msec (approximately)

FMPY = 3.3 msec (approximately)
 FGET = 156 μ sec
 FPUT = 172 μ sec
 FNOR = $168 + N(42)$ μ sec where N is number of shifts;
 +84 μ sec if argument <0.
 FEXT = 140.5 μ sec

10. PROGRAM

10.4 Program Listing

/4 WORD FLOATING POINT
 /ARITHMETIC INTERPRETER
 /PAGE 1

*40

0040	0000	EX1,	0
0041	0000	HIGH1,	0
0042	0000	MID1,	0
0043	0000	LOW1,	0
0044	0000	EXP,	0
0045	0000	HORDER,	0
0046	0000	MIDDL,	0
0047	0000	LORDER,	0
0050	0000	OVER2,	0

0051	0000	OVER1,	0
------	------	--------	---

*61

0061	0000	FLAG,	0	/ARITHMETIC ERROR FLAG
------	------	-------	---	------------------------

*5500

5600	0000	FPNT,	0	
5601	7300	CLA CLL		
5602	3051	DCA OVER1		
5603	3050	DCA OVER2		
5604	1600	TAD I FPNT		/GET INSTRUCTION
5605	3257	DCA JUMP		
5606	1257	TAD JUMP		
5607	0265	AND PAGENO		/PAGE 0??
5610	7650	SNA CLA		
5611	5214	JMP .+3		/YES
5612	1267	TAD MASK5		/NO - GET PAGE BITS
5613	0200	AND FPNT		
5614	3262	DCA ADDRS		
5615	1270	TAD MASK7		/GET 7 BIT ADDRESS
5616	0257	AND JUMP		
5617	1262	TAD ADDRS		
5620	3262	DCA ADDRS		

5621	1266		TAD INDRCT	/BIT3=1??
5622	0257		AND JUMP	
5623	7650		SNA CLA	
5624	5227		JMP LOOP01	
5625	1662		TAD I ADDRS	/YES - DEFER
5626	3262		DCA ADDRS	
5627	2200	LOOP01,	ISZ FPNT	
5630	1662		TAD I ADDRS	
5631	3040		DCA EXI	/EXPONENT
5632	1262		TAD ADDRS	
5633	3263		DCA SAVE	
5634	2263		ISZ SAVE	
5635	1663		TAD I SAVE	/HIGH ORDER
5636	3041		DCA HIGH1	
5637	2263		ISZ SAVE	
5640	1663		TAD I SAVE	
5641	3042		DCA MID1	/MIDDLE BITS
5642	2263		ISZ SAVE	
5643	1663		TAD I SAVE	
5644	3043		DCA LOW1	/LOWER BITS
5645	1257		TAD JUMP	
5646	7106		CLL RTL	
5647	7006		RTL	
5650	0264		AND MASK3	/LOOK-UP ON TABLE
5651	1271		TAD TABLE	
5652	3260		DCA JUMP2	
5653	1660		TAD I JUMP2	
5654	3260		DCA JUMP2	
5655	4660		JMS I JUMP2	/EXECUTE
5656	5201		JMP FPNT+1	/GET NEXT
5657	0000	JUMP,	0	
5660	0000	JUMP2,	0	
5661	0000	GO2,	0	
5662	0000	ADDRS,	0	
5663	0000	SAVE,	0	
5664	0017	MASK3,	0017	
5665	0200	PAGENO,	0200	
5666	0400	INDRCT,	0400	
5667	7600	MASK5,	7600	
5670	0177	MASK7,	0177	
5671	5672	TABLE,	.+1	
5672	5714		EXIT	
5673	6000		FLAD	
5674	6026		FLSU	
5675	6367		FLMY	
5676	6600		FLDV	
5677	5702		FLGT	
5700	5733		FLPT	
5701	6200		FNORM	

		/FLOATING GET=5000	
5702	0000	FLGT,	0
5703	1040		TAD EX1
5704	3044		DCA EXP
5705	1041		TAD HIGH1
5706	3045		DCA HORDER
5707	1042		TAD MID1
5710	3046		DCA MIDDLE
5711	1043		TAD LOW1
5712	3047		DCA LORDER
5713	5201		JMP FPNT+1
		/FLOATING EXIT OR SUBROUTINE=00XX	
5714	0000	EXIT,	0
5715	1257		TAD JUMP
5716	0264		AND MASK3
5717	7450		SNA
5720	5600		JMP I FPNT
5721	1350		TAD TABLE6
5722	3260		DCA JUMP2
5723	1660		TAD I JUMP2
5724	3260		DCA JUMP2
5725	1200		TAD FPNT
5726	3261		DCA G02
5727	4660		JMS I JUMP2
5730	1261		TAD G02
5731	3200		DCA FPNT
5732	5201		JMP FPNT+1
		/FLOATING PUT=6000	
5733	0000	FLPT,	0
5734	1044		TAD EXP
5735	3662		DCA I ADDRS
5736	1045		TAD HORDER
5737	2262		ISZ ADDRS
5740	3662		DCA I ADDRS
5741	1046		TAD MIDDLE
5742	2262		ISZ ADDRS
5743	3662		DCA I ADDRS
5744	1047		TAD LORDER
5745	2262		ISZ ADDRS
5746	3662		DCA I ADDRS
5747	5201		JMP FPNT+1
5750	5750	TABLE6,	.
5751	5770		EXIT6
5752	5770		EXIT6
5753	5770		EXIT6
5754	5770		EXIT6
5755	5770		EXIT6
5756	5770		EXIT6

/BITS 8-11=0??

/YES:FEXT

/NO:LOOKUP BITS 8-11

/ON SUBROUTINE TABLE

/SAVE PSEUDO PC

/RESTORE PSEUDO PC

/RETURN

/SUBROUTINE TABLE
/ABSOLUTE ADDRESSES
/OF SUBROUTINES

/EXIT6=DUMMY OR NOP

5757	5770	EXIT6
5760	5770	EXIT6
5761	5770	EXIT6
5762	5770	EXIT6
5763	5770	EXIT6
5764	5770	EXIT6
5765	5770	EXIT6
5766	5770	EXIT6
5767	5770	EXIT6

5770	0000	EXIT6,	0
5771	5770	JMP I	EXIT6

/FLOATING ADD=1000

*6000

6000	0000	FLAD,	0	
6001	4231	JMS ALIGN		/ALIGN WORDS
6002	5600	JMP I FLAD		/NO ALIGNMENT
6003	4312	JMS SCALE		
6004	7300	CLA CLL		/TRIPLE ADDITION
6005	1051	TAD OVER1		
6006	1050	TAD OVER2		
6007	3050	DCA OVER2		
6010	7004	RAL		/CARRY
6011	1043	TAD LOW1		
6012	1047	TAD LORDER		
6013	3047	DCA LORDER		
6014	7004	RAL		
6015	1042	TAD MID1		
6016	1046	TAD MIDDLE		
6017	3046	DCA MIDDLE		
6020	7004	RAL		
6021	1041	TAD HIGH1		
6022	1045	TAD HORDER		
6023	3045	DCA HORDER		
6024	4705	JMS I NORMAL		
6025	5600	JMP I FLAD		

/FLOATING SUBTRACT=2000

6026	0000	FLSU,	0	
6027	4706	JMS I OPINS		/NEGATE OPERAND
6030	5201	JMP FLSUX		/ADD

/ALIGN BINARY POINTS

6031	0000	ALIGN,	0	
6032	1045	TAD HORDER		
6033	7640	SZA CLA		
6034	5240	JMP .+4		

6035	1040		TAD EX1	/C(FAC)=0
6036	3044		DCA EXP	
6037	5272		JMP DONE	
6040	1041		TAD HIGH1	
6041	7650		SNA CLA	
6042	5631		JMP I ALIGN	/OPERAND=0
6043	1040		TAD EX1	
6044	7041		CMA IAC	
6045	1044		TAD EXP	
6046	7450		SNA	
6047	5272		JMP DONE	/EXPONENTS EQUAL - EXIT
6050	7500		SMA	
6051	7041		CMA IAC	
6052	3304		DCA AMOUNT	/NUMBER OF PLACES
6053	1304		TAD AMOUNT	
6054	1307		TAD TEST1	
6055	7710		SPA CLA	
6056	5274		JMP NOGO	/NO SHIFTING POSSIBLE
6057	1040		TAD EX1	
6060	7041		CMA IAC	
6061	1044		TAD EXP	
6062	7004		RAL	
6063	7620		SNL CLA	
6064	1310		TAD TCON1	/SHIFT OPERAND RIGHT
6065	1311		TAD TCON2	/SHIFT FAC RIGHT
6066	3303		DCA POINT	
6067	4703		JMS I POINT	
6070	2304		ISZ AMOUNT	
6071	5267		JMP .-2	
6072	2231	DONE,	ISZ ALIGN	
6073	5631		JMP I ALIGN	
6074	1040	NOGO,	TAD EX1	
6075	7041		CMA IAC	
6076	1044		TAD EXP	
6077	7700		SMA CLA	
6100	5631		JMP I ALIGN	
6101	5702		JMP I .+1	
6102	5703		FLGT+1	
6103	0000	POINT,	0	
6104	0000	AMOUNT,	0	
6105	6200	NORMAL,	FNORM	
6106	6306	OPMINS,	OPNEG	
6107	0045	TEST1,	0045	
6110	0023	TCON1,	SHFTOP-SHFTAC	
6111	6116	TCON2,	SHFTAC	

		/SCALE BOTH RIGHT	
6112	0000	SCALE,	0
6113	4341		JMS SHFTOP
6114	4316		JMS SHFTAC
6115	5712		JMP I SCALE
		/SCALE FLOATING AC RIGHT	
6116	0000	SHFTAC,	0
6117	7300		CLA CLL
6120	1045		TAD HORDER
6121	7510		SPA
6122	7020		CML
6123	7010		RAR
6124	3045		DCA HORDER
6125	1046		TAD MIDDLE
6126	7010		RAR
6127	3046		DCA MIDDLE
6130	1047		TAD LORDER
6131	7010		RAR
6132	3047		DCA LORDER
6133	1050		TAD OVER2
6134	7010		RAR
6135	3050		DCA OVER2
6136	2044		ISZ EXP
6137	7000		NOP
6140	5716		JMP I SHFTAC
		/SCALE OPERAND RIGHT	
6141	0000	SHFTOP,	0
6142	7300		CLA CLL
6143	1041		TAD HIGH1
6144	7510		SPA
6145	7020		CML
6146	7010		RAR
6147	3041		DCA HIGH1
6150	1042		TAD MID1
6151	7010		RAR
6152	3042		DCA MID1
6153	1043		TAD LOW1
6154	7010		RAR
6155	3043		DCA LOW1
6156	1051		TAD OVER1
6157	7010		RAR
6160	3051		DCA OVER1
6161	2040		ISZ EX1
6162	7000		NOP
6163	5741		JMP I SHFTOP
6164	4200	FLSUX,	JMS FLAD
6165	5626		JMP I FLUX

/NORMALIZE FLOATING ACCUMULATOR

*6200

6200	0000	FNORM,	0	
6201	7300		CLA CLL	
6202	3361		DCA MP1	/0 # OF SHIFTS
6203	3363		DCA MP3	/RESET SWITCH
6204	1045		TAD HORDER	
6205	7510		SPA	/INPUT<0
6206	2363		ISZ MP3	/YES-SET SWITCH
6207	7640		SZA CLA	/FAC=0?
6210	5224		JMP G06	/NO
6211	1046		TAD MIDDLE	
6212	7640		SZA CLA	
6213	5224		JMP G06	
6214	1047		TAD LORDER	
6215	7640		SZA CLA	
6216	5224		JMP G06	/NO
6217	1050		TAD OVER2	
6220	7640		SZA CLA	
6221	5224		JMP G06	/NO
6222	3044		DCA EXP	/YES
6223	5600		JMP I FNORM	/EXIT
6224	1363	G06,	TAD MP3	
6225	7640		SZA CLA	/WAS INPUT <0
6226	4261		JMS ACNEG	/YES
6227	1045	SHIFT,	TAD HORDER	
6230	7104		CLL RAL	
6231	7710		SPA CLA	/TOO FAR?
6232	5251		JMP NOREXT	/YES:EXIT ROUTINE
6233	1050		TAD OVER2	/NO
6234	7104		CLL RAL	
6235	3050		DCA OVER2	/SHIFT LEFT
6236	1047		TAD LORDER	
6237	7004		RAL	
6240	3047		DCA LORDER	
6241	1046		TAD MIDDLE	
6242	7004		RAL	
6243	3046		DCA MIDDLE	
6244	1045		TAD HORDER	
6245	7004		RAL	
6246	3045		DCA HORDER	
6247	2361		ISZ MP1	/ADD 1 TO COUNT
6250	5227		JMP SHIFT	/CONTINUE
6251	1361	NOREXT,	TAD MP1	/SUBTRACT COUNT FROM
6252	7041		CMA IAC	/EXPONENT
6253	1044		TAD EXP	
6254	3044		DCA EXP	
6255	1363		TAD MP3	/WAS INPUT<0??
6256	7640		SZA CLA	
6257	4261		JMS ACNEG	/YES
6260	5600		JMP I FNORM	/EXIT

		/NEGATE FLOATING AC
6261	0000	ACNEG, 0
6262	7300	CLA CLL
6263	1050	TAD OVER2
6264	7041	CMA IAC
6265	3050	DCA OVER2
6266	1047	TAD LORDER
6267	7040	CMA
6270	7430	SZL
6271	7101	CLL IAC
6272	3047	DCA LORDER
6273	1046	TAD MIDDLE
6274	7040	CMA
6275	7430	SZL
6276	7101	CLL IAC
6277	3046	DCA MIDDLE
6300	1045	TAD HORDER
6301	7040	CMA
6302	7430	SZL
6303	7101	CLL IAC
6304	3045	DCA HORDER
6305	5661	JMP I ACNEG

/NEGATE OPERAND

6306	0000	OPNEG, 0
6307	7300	CLA CLL
6310	1051	TAD OVER1
6311	7041	CMA IAC
6312	3051	DCA OVER1
6313	1043	TAD LOW1
6314	7040	CMA
6315	7430	SZL
6316	7101	CLL IAC
6317	3043	DCA LOW1
6320	1042	TAD MID1
6321	7040	CMA
6322	7430	SZL
6323	7101	CLL IAC
6324	3042	DCA MID1
6325	1041	TAD HIGH1
6326	7040	CMA
6327	7430	SZL
6330	7101	CLL IAC
6331	3041	DCA HIGH1
6332	5706	JMP I OPNEG

6333	0000	MULTIP,	0
6334	3361		DCA MP1
6335	3364		DCA MPSCON
6336	1365		TAD THIR
6337	3363		DCA MP3
6340	7100		CLL
6341	1361		TAD MP1
6342	7010		RAR
6343	3361		DCA MP1
6344	1364		TAD MPSCON
6345	7420		SNL
6346	5351		JMP .+3
6347	7100		CLL
6350	1362		TAD MP2CON
6351	7010		RAR
6352	3364		DCA MPSCON
6353	2363		ISZ MP3
6354	5341		JMP MULTIP+6
6355	1361		TAD MP1
6356	7010		RAR
6357	7100		CLL
6360	5733		JMP I MULTIP
6361	0000	MP1,	0
6362	0000	MP2CON,	0
6363	0000	MP3,	0
6364	0000	MPSCON,	0
6365	7764	THIR,	-14
6366	6400	FMULT1,	FMULT
6367	0000	FLMY,	0
6370	4766		JMS I FMULT1
6371	4200		JMS FNORM
6372	3050		DCA OVER2
6373	2777		ISZ I SIGN1
6374	5767		JMP I FLMY
6375	4261		JMS ACNEG
6376	5767		JMP I FLMY
6377	6750	SIGN1,	SGNTST

*6400

/FLOATING MULTIPLY

/(A*2↑24+B*2↑12+C)*(D*2↑24+E*2↑12+F)

6400	0000	FMULT,	0
6401	7201		CLA IAC
6402	1040		TAD EX1
6403	1044		TAD EXP
6404	3044		DCA EXP
6405	1377		TAD SMACLA
6406	3772		DCA I SGNSW
6407	4773		JMS I SIGNP

/ADD EXPONENTS

/SET UP SIGN ROUTINE

/GO THERE

6410	1045	TAD LOW1	
6411	3775	DCA I MP2	
6412	1047	TAD LORDER	/C*F
6413	4774	JMS I DMULT	
6414	7200	CLA	
6415	1776	TAD I MP5	
6416	3371	DCA MUL5	
6417	1046	TAD MIDDLE	
6420	3775	DCA I MP2	
6421	1043	TAD LOW1	/B*F
6422	4774	JMS I DMULT	
6423	1371	TAD MUL5	
6424	3371	DCA MUL5	
6425	7004	RAL	
6426	1776	TAD I MP5	
6427	3370	DCA MUL4	
6430	7004	RAL	
6431	3367	DCA MUL3	
6432	1042	TAD MID1	
6433	3775	DCA I MP2	
6434	1047	TAD LORDER	/C*E
6435	4774	JMS I DMULT	
6436	1371	TAD MUL5	
6437	3371	DCA MUL5	
6440	7004	RAL	
6441	1370	TAD MUL4	
6442	1776	TAD I MP5	
6443	3370	DCA MUL4	
6444	7004	RAL	
6445	1367	TAD MUL3	
6446	3367	DCA MUL3	
6447	1045	TAD HORDER	
6450	3775	DCA I MP2	
6451	1043	TAD LOW1	/A*F
6452	4774	JMS I DMULT	
6453	1370	TAD MUL4	
6454	3370	DCA MUL4	
6455	7004	RAL	
6456	1367	TAD MUL3	
6457	1776	TAD I MP5	
6460	3367	DCA MUL3	
6461	7004	RAL	
6462	3366	DCA MUL2	
6463	1041	TAD HIGH1	
6464	3775	DCA I MP2	
6465	1047	TAD LORDER	/D*C
6466	4774	JMS I DMULT	
6467	1370	TAD MUL4	
6470	3370	DCA MUL4	
6471	7004	RAL	

6472	1367	TAD MUL3	
6473	1776	TAD I MP5	
6474	3367	DCA MUL3	
6475	7004	RAL	
6476	1366	TAD MUL2	
6477	3366	DCA MUL2	
6500	1046	TAD MIDDLE	
6501	3775	DCA I MP2	
6502	1042	TAD MID1	/B*D
6503	4774	JMS I DMULT	
6504	1370	TAD MUL4	
6505	3370	DCA MUL4	
6506	7004	RAL	
6507	1367	TAD MUL3	
6510	1776	TAD I MP5	
6511	3367	DCA MUL3	
6512	7004	RAL	
6513	1366	TAD MUL2	
6514	3366	DCA MUL2	
6515	1045	TAD HORDER	
6516	3775	DCA I MP2	
6517	1042	TAD MID1	/A*E
6520	4774	JMS I DMULT	
6521	1367	TAD MUL3	
6522	3367	DCA MUL3	
6523	7004	RAL	
6524	1366	TAD MUL2	
6525	1776	TAD I MP5	
6526	3366	DCA MUL2	
6527	7004	RAL	
6530	3365	DCA MUL1	
6531	1041	TAD HIGH1	
6532	3775	DCA I MP2	
6533	1046	TAD MIDDLE	/B*D
6534	4774	JMS I DMULT	
6535	1367	TAD MUL3	
6536	3367	DCA MUL3	
6537	7004	RAL	
6540	1366	TAD MUL2	
6541	1776	TAD I MP5	
6542	3366	DCA MUL2	
6543	7004	RAL	
6544	1365	TAD MUL1	
6545	3365	DCA MUL1	
6546	1045	TAD HORDER	
6547	3775	DCA I MP2	
6550	1041	TAD HIGH1	/A*D
6551	4774	JMS I DMULT	
6552	1366	TAD MUL2	

6553	3046		DCA MIDDLE
6554	7004		RAL
6555	1365		TAD MUL1
6556	1776		TAD I MP5
6557	3045		DCA HORDER
6560	1367		TAD MUL3
6561	3047		DCA LORDER
6562	1370		TAD MUL4
6563	3050		DCA OVER2
6564	5600		JMP I FMULT
6565	0000	MUL1,	0
6566	0000	MUL2,	0
6567	0000	MUL3,	0
6570	0000	MUL4,	0
6571	0000	MUL5,	0
6572	6740	SGNSW,	SGNSWT
6573	6727	SIGNP,	SIGNCL
6574	6333	DMULT,	MULTIP
6575	6362	MP2,	MP2CON
6576	6364	MP5,	MP5CON
6577	7700	SMA CLA,	SMA CLA

/FLOATING DIVIDE=4000

*6600

6600	0000	FLDV,	0
6601	1040		TAD EX1
6602	7041		CMA IAC
6603	1044		TAD EXP
6604	7001		IAC
6605	3044		DCA EXP
6606	1326		TAD SPACLA
6607	3340		DCA SGNSWT
6610	4327		JMS SIGNCL
6611	1041		TAD HIGH1
6612	7650		SNA CLA
6613	5303		JMP DVER
6614	7300		CLA CLL
6615	3320		DCA QUOL
6616	3321		DCA QUOH
6617	1325		TAD MIF
6620	3324		DCA DIVCNT
6621	5233		JMP DVX
6622	1047	DV3,	TAD LORDER
6623	7004		RAL
6624	3047		DCA LORDER
6625	1046		TAD MIDDLE
6626	7004		RAL

/SUBTRACT EXPONENTS

/SET UP SIGNS

/DIVISOR=0?

/YES - ERROR

6627	3046		DCA MIDDLE	
6630	1045		TAD HORDER	
6631	7004		RAL	
6632	3045		DCA HORDER	
6633	1043	DVX,	TAD LOW1	/PARTIAL SUBTRACT
6634	1047		TAD LORDER	
6635	3322		DCA DTEM1	
6636	7004		RAL	
6637	1042		TAD MID1	
6640	1046		TAD MIDDLE	
6641	3323		DCA DTEM2	
6642	7004		RAL	
6643	1041		TAD HIGH1	
6644	1045		TAD HORDER	
6645	7420		SNL	/DIVISOR<DIVIDEND?
6646	5254		JMP DV2-1	/NO
6647	3045		DCA HORDER	/YES:C(L)=QUOTIENT BIT
6650	1323		TAD DTEM2	
6651	3046		DCA MIDDLE	
6652	1322		TAD DTEM1	
6653	3047		DCA LORDER	
6654	7200		CLA	
6655	1320	DV2,	TAD QUOL	/SHIFT BIT INTO
6656	7004		RAL	/QUOTIENT
6657	3320		DCA QUOL	
6660	1321		TAD QUOH	
6661	7004		RAL	
6662	3321		DCA QUOH	
6663	1050		TAD OVER2	
6664	7004		RAL	
6665	3050		DCA OVER2	
6666	2324		ISZ DIVCNT	/DONE?
6667	5222		JMP DV3	/NO
6670	1320		TAD QUOL	
6671	3047		DCA LORDER	
6672	1321		TAD QUOH	
6673	3046		DCA MIDDLE	
6674	1050		TAD OVER2	
6675	3045		DCA HORDER	
6676	3050		DCA OVER2	
6677	4717		JMS I NORMIT	
6700	2350	DEXIT,	ISZ SGNST	
6701	4746		JMS I FACNEG	
6702	5600		JMP I FLDV	

6703	7240	DVER,	CLA CMA	/DIVIDE ERROR
6704	3047		DCA LORDER	
6705	7240		CLA CMA	
6706	3046		DCA MIDDLE	
6707	7040		CMA	
6710	7110		CLL RAR	
6711	3045		DCA HORDER	
6712	1045		TAD HORDER	
6713	3044		DCA EXP	
6714	2061		ISZ FLAG	
6715	7000		NOP	
6716	5300		JMP DEXIT	

6717	6200	NORMIT,	FNORM	
6720	0000	QUOL,	0	
6721	0000	QUOH,	0	
6722	0000	DTEM1,	0	
6723	0000	DTEM2,	0	
6724	0000	DIVCNT,	0	
6725	7735	MIF,	-45	/STEP COUNT
6726	7710	SPACLA,	SPA CLA	

/TEST SIGN SUBROUTINE

6727	0000	SIGNCL,	0	
6730	1351		TAD RESTOR	
6731	3350		DCA SGNST	
6732	1045		TAD HORDER	
6733	7700		SMA CLA	
6734	5337		JMP .+3	
6735	4746		JMS I FACNEG	
6736	2350		ISZ SGNST	
6737	1041		TAD HIGH1	
6740	7700	SGNSWT,	SMA CLA	/OR SPA CLA
6741	5727		JMP I SIGNCL	
6742	4747		JMS I OPNEGS	
6743	2350		ISZ SGNST	
6744	7000		NOP	
6745	5727		JMP I SIGNCL	
6746	6261	FACNEG,	ACNEG	
6747	6306	OPNEGS,	OPNEG	
6750	0000	SGNST,	0	
6751	7776	RESTOR,	-2	

ACNEG	6261
ADDRS	5662
ALIGN	6031
AMOUNT	6104
DEXIT	6700
DIVCNT	6724
DMULT	6574
DONE	6072
DTM1	6722
DTM2	6723
DVER	6703
DVX	6633
DV2	6655
DV3	6622
EXIT	5714
EXIT6	5770
EXP	0044
EX1	0040
FACNEG	6746
FLAD	6000
FLAG	0061
FLDV	6600
FLGT	5702
FLMY	6367
FLPT	5733
FLSU	6026
FMULT	6400
FMULT1	6366
FNORM	6200
FPNT	5600
G02	5661
G06	6224
HIGH1	0041
HORDER	0045
INDRCT	5666
JUMP	5657
JUMP2	5660
LOOP01	5627
LORDER	0047
LOW1	0043
MASK3	5664
MASK5	5667
MASK7	5670
MIDDLE	0046
MID1	0042
MIF	6725

MPSCON	6364
MP1	6361
MP2	6575
MP2CON	6362
MP3	6363
MP5	6576
MULTIP	6333
MUL1	6565
MUL2	6566
MUL3	6567
MUL4	6570
MUL5	6571
NOGO	6074
NOREXT	6251
NORMAL	6105
NORMIT	6717
OPMINS	6106
OPNEG	6306
OPNEGS	6747
OVER1	0051
OVER2	0050
PAGENO	5665
POINT	6103
QUOH	6721
QUOL	6720
RESTOR	6751
SAVE	5663
SCALE	6112
SGNSW	6572
SGNSWT	6740
SGNTST	6750
SHFTAC	6116
SHFTOP	6141
SHIFT	6227
SIGNCL	6727
SIGNP	6573
SIGNI	6377
SMACLA	6577
SPACLA	6726
TABLE	5671
TABLE6	5750
TCO1	6110
TCO2	6111
TEST1	6107
THIR	6365

/4/17/65-HB-DEC
 /4 WORD
 /FLOATING POINT I/O ROUTINES
 /REQUIRES FLOATING POINT INTERPRETER
 /ENTRY AT 0007

		*7		
0007	5600	FPNT,	5600	
		*44		
0044	0000	EXPONT,	0	
0045	0000	HORDER,	0	
0046	0000	MIDDL,	0	
0047	0000	LORDER,	0	
		*52		
0052	0000	FPAC1,	0	
0053	0000		0	
0054	0000		0	
0055	0000		0	
0056	7777	SWIT1,	7777	/IF = 0, NO CR-LF AFTER OUTPUT
0057	7777	SWIT2,	7777	/IF = 0, NO LF AFTER CR IN INPUT
0060	0000	CHAR,	0	/CONTAINS LAST CHARACTER READ
0061	0000	DSWIT,	0	/= 0 IF NO CONVERSION TOOK PLACE
		*6767		
6767	0000	PRCHAR,	0	
6770	1057		TAD SWIT2	
6771	7650		SNA CLA	
6772	5767		JMP I PRCHAR	
6773	1377		TAD LFED	
6774	4776		JMS I OPUT	
6775	5767		JMP I PRCHAR	
6776	7345	OPUT,	OUT	
6777	0212	LFED,	0212	

/DOUBLE PRECISION DECIMAL-BINARY
 /INPUT AND CONVERSION
 *7000

7000	0000	DECONV,	0	
7001	7200		CLA	/INITIALIZE MANTISSA
7002	3045		DCA HORDER	
7003	3046		DCA MIDDL	
7004	3047		DCA LORDER	
7005	3266		DCA SIGN	
7006	3267		DCA DNUMBR	
7007	4350		JMS INPUT	

7010	1340		TAD PLUS	/TEST FOR SIGN
7011	7450		SNA	
7012	5220		JMP DECON	
7013	1337		TAD MINUS	
7014	7440		SZA	
7015	5221		JMP .+4	
7016	7240		CLA CMA	
7017	3266		DCA SIGN	/IF-, SET SWITCH
7020	4350	DECON,	JMS INPUT	
7021	7200		CLA	
7022	1060		TAD CHAR	/IS IT A DIGIT
7023	1341		TAD MIN9	
7024	7500		SMA	
7025	5600		JMP I DECONV	/NO
7026	1342		TAD PLUS12	
7027	7510		SPA	
7030	5600		JMP I DECONV	/NO
7031	3265		DCA DIGIT	/YES
7032	1045		TAD HORDER	
7033	0343		AND MASK	/OVERFLOW?
7034	7440		SZA	
7035	5220		JMP DECON	/YES-IGNORE
7036	2061		ISZ DSWIT	
7037	2267		ISZ DNUMBR	/INDEX NUMBER OF DIGITS
7040	4242		JMS MULT10	
7041	5220		JMP DECON	/CONTINUE
7042	0000	MULT10,	0	/ROUTINE TO MULTIPLY
7043	1047		TAD LORDER	/DOUBLE PRECISION WORD
7044	3043		DCA 43	/BY TEN (DECIMAL)
7045	1046		TAD MIDDLE	
7046	3042		DCA 42	
7047	1045		TAD HORDER	/REMAIN=REMAINDER
7050	3041		DCA 41	
7051	3040		DCA 40	
7052	4270		JMS MULT2	/CALL SUBROUTINE TO
7053	4270		JMS MULT2	/MULTIPLY BY TWO
7054	4307		JMS DUBLAD	/CALL DOUBLE ADD
7055	4270		JMS MULT2	
7056	1265		TAD DIGIT	/ADD LAST DIGIT RECEIVED
7057	3043		DCA 43	
7060	3042		DCA 42	
7061	3041		DCA 41	
7062	4307		JMS DUBLAD	
7063	1040		TAD 40	/EXIT WITH REMAINDER
7064	5642		JMP I MULT10	/IN AC
7065	0000	DIGIT,	0	/STORAGE FOR DIGIT
7066	0000	SIGN,	0	/=0 IF PLUS: =7777 IF MINUS
7067	0000	DNUMBR,	0	/=NUMBER OF DIGITS
7070	0000	MULT2,	0	/MULTIPLY LORDER, HORDER BY 2

7071	7300		CLA CLL	
7072	1047		TAD LORDER	
7073	7004		RAL	
7074	3047		DCA LORDER	
7075	1046		TAD MIDDLE	
7076	7004		RAL	
7077	3046		DCA MIDDLE	
7100	1045		TAD HORDER	
7101	7004		RAL	
7102	3045		DCA HORDER	
7103	1040		TAD 40	
7104	7004		RAL	
7105	3040		DCA 40	
7106	5670		JMP I MULT2	
7107	0000	DUBLAD,	0	/DOUBLE PRECISION ADDITION
7110	7300		CLA CLL	
7111	1047		TAD LORDER	
7112	1043		TAD 43	
7113	3047		DCA LORDER	
7114	7004		RAL	
7115	1046		TAD MIDDLE	
7116	1042		TAD 42	
7117	3046		DCA MIDDLE	
7120	7004		RAL	
7121	1045		TAD HORDER	
7122	1041		TAD 41	
7123	3045		DCA HORDER	
7124	7004		RAL	
7125	1040		TAD 40	
7126	3040		DCA 40	
7127	5707		JMP I DUBLAD	
7130	0000	MSIGN,	0	/ROUTINE TO FORM
7131	7300		CLA CLL	/2'S COMPLEMENT
7132	2266		1SZ SIGN	/IF C(SIGN)=7777
7133	5730		JMP I MSIGN	
7134	4736		JMS I .+2	
7135	5730		JMP I MSIGN	
7136	6261		6261	/ACNEG IN INTERPRETER
7137	7776	MINUS,	253-255	/TEST FOR SIGN
7140	7525	PLUS,	-253	
7141	7506	MIN9,	-272	/TEST FOR DIGIT
7142	0012	PLUS12,	272-260	
7143	7600	MASK,	7600	/TEST FOR OVERFLOW
7144	7775	C.10,	7775	
7145	3146		3146	
			7146 3146	3146
7147	3147		3147	

```

/INPUT A CHARACTER, IF CR, TEST
/INPUT SWITCH TO SEE IF LF SHOULD
/BE TYPED. IF RUBOUT, RESTART INPUT
7150 0000 INPUT, 0 /INPUT A CHARACTER
7151 7200 CLA
7152 6031 KSF
7153 5352 JMP .-1
7154 6036 KRB
7155 3060 DCA CHAR
7156 1060 TAD CHAR
7157 4774 JMS I OUTPUT
7160 1060 TAD CHAR
7161 7450 SNA
7162 5351 JMP INPUT+1 /IGNORE BLANKS
7163 1376 TAD MRBOUT
7164 7450 SNA
7165 5775 JMP I RESTRT /RUBOUT-RESTART INPUT
7166 1377 TAD MINCR
7167 7650 SNA CLA
7170 4773 JMS I PRINT /CR - SEE IF TO BE FOLLOWED
7171 1060 TAD CHAR /BY LF
7172 5750 JMP I INPUT /EXIT ROUTINE

7173 6767 PRINT, PRCHAR
7174 7345 OUTPUT, OUT
7175 7401 RESTRT, FLINTP+1
7176 7401 MRBOUT, -377
7177 0162 MINCR, 377-215

```

/FLOATING OUTPUT "E" FORMAT

```

/USES: TSF
/      JMP .-1
/      TLS

```

*7200

```

7200 0000 FLOUTP, 0
7201 4217 JMS FOUTCN /CONVERT MANTISSA AND OUTPUT
7202 1324 TAD BEXP
7203 3044 DCA EXPONT
7204 1343 TAD CHE
7205 4345 JMS OUT
7206 4737 JMS I FEXPPT /CONVERT EXPONENT AND OUTPUT
7207 1056 TAD SWIT1 /PRINT CR-LF?
7210 7650 SNA CLA
7211 5600 JMP I FLOUTP /NO-EXIT
7212 1341 TAD CARRTN /YES
7213 4345 JMS OUT
7214 1342 TAD LNFEED
7215 4345 JMS OUT
7216 5600 JMP I FLOUTP /EXIT

```


/THIS WHOLE SUBROUTINE MAY BE ALTERED TO BUFFER
/THE OUTPUT DIGITS : CHANGE JMS OUTDG TO DCA I 10, ETC.

7217	0000	FOUTCN,	0	
7220	7300		CLA CLL	
7221	1045		TAD HORDER	/NUMBER>0??
7222	7710		SPA CLA	
7223	7220		CLA CML	/NO SET LINK
7224	1327		TAD SPLUS	/YES
7225	7430		SZL	
7226	1330		TAD SMINUS	/NO
7227	4345		JMS OUT	
7230	4353		JMS OUTDG	/OUTPUT "0"
7231	1331		TAD PERIOD	
7232	4345		JMS OUT	/OUTPUT "."
7233	7300		CLA CLL	
7234	1045		TAD HORDER	
7235	7700		SMA CLA	
7236	5242		JMP FG01	
7237	7040		CMA	/NUMBER IS NEGATIVE
7240	3733		DCA I SNPT	/NEGATE
7241	4732		JMS I MSNPT	
7242	7240	FG01,	CLA CMA	/SUBTRACT 1 FROM BINARY EXPON
7243	1044		TAD EXPONT	/COMPENSATE AT FG04
7244	3044		DCA EXPONT	
7245	3324		DCA BEXP	/INITIALIZE DECIMAL EXPONENT
7246	1044	FG02,	TAD EXPONT	/IS -4<EXPONENT<-1
7247	7500		SMA	
7250	5263		JMP FG03	/TOO LARGE: MULTIPLY BY 1/10
7251	1326		TAD FOUR	
7252	7700		SMA CLA	
7253	5270		JMP FG04	
7254	4407		JMS I FPNT	/TOO SMALL-TIMES TEN
7255	3740		FMPY I TENPT	/TEN
7256	0000		FEXT	
7257	7240		CLA CMA	
7260	1324		TAD BEXP	
7261	3324		DCA BEXP	
7262	5246		JMP FG02	
7263	4407	FG03,	JMS I FPNT	
7264	3744		FMPY I PRC.10	/ONE TENTH
7265	0000		FEXT	
7266	2324		ISZ BEXP	
7267	5246		JMP FG02	

7270	3734	FG04,	DCA I DPT	/MULTIPLY BY TWO
7271	4736		JMS I M2PT	/IE.SHIFT LEFT
7272	4735		JMS I M10PT	/MULTIPLY BY TEN
7273	7410		SKP	
7274	4360	FG05A,	JMS DIVTWO	/COMPENSATE FOR
7275	2044		ISZ EXPONT	/BINARY EXPONENT
7276	5274		JMP FG05A	
7277	7450		SNA	/IS FIRST DIGIT A ZERO
7300	5311		JMP FG07	/YES, IGNORE
7301	4353	FG06,	JMS OUTDG	/MULTIPLICATIONS YIELD
7302	1325		TAD MINUS7	/DECIMAL DIGITS AS HIGH
7303	3044		DCA EXPONT	/ORDER REMAINDERS
7304	4735	FG06A,	JMS I M10PT	/IE. .672X10=6+.72.. ETC
7305	4353		JMS OUTDG	
7306	2044		ISZ EXPONT	/7 DIGITS OUTPUT??
7307	5304		JMP FG06A	/NO: CONTINUE
7310	5617		JMP I FOUTCN	/YES:EXIT
7311	7240	FG07,	CLA CMA	/IGNORE FIRST DIGIT
7312	1324		TAD BEXP	/SUBTRACT 1 FROM
7313	3324		DCA BEXP	/DECIMAL EXPONENT
7314	1045		TAD HORDER	
7315	7640		SZA CLA	
7316	5322		JMP .+4	/IS MANTISSA ZERO?
7317	1047		TAD LORDER	
7320	7650		SNA CLA	
7321	3324		DCA BEXP	/YES:EXP=0
7322	7240		CLA CMA	
7323	5302		JMP FG06+1	
7324	0000	BEXP,	0	/CONTAINS DECIMAL EXPONENT
7325	7767	MINUS7,	-11	/NUMBER OF DIGITS OUTPUT
7326	0004	FOUR,	0004	
7327	0253	SPLUS,	253	
7330	0002	SMINUS,	255-253	
7331	0256	PERIOD,	256	
7332	7130	MSNPT,	MSIGN	
7333	7066	SNPT,	SIGN	/POINTERS
7334	7065	DPT,	DIGIT	
7335	7042	M10PT,	MULT10	
7336	7070	M2PT,	MULT2	
7337	7523	FEXPPT,	FEXC	
7340	7504	TENPT,	TEN	
7341	0215	CARRTN,	0215	
7342	0212	LNFEED,	0212	
7343	0305	CHE,	305	
7344	7144	PRC.10,	C.10	

7345	0000	OUT,	0	/OUTPUT ONE ASCII CHARACTER
7346	6041		TSF	
7347	5346		JMP .-1	
7350	6046		TLS	
7351	7200		CLA	
7352	5745		JMP I OUT	
7353	0000	OUTDG,	0	/OUTPUT ONE DIGIT
7354	1357		TAD C260	
7355	4345		JMS OUT	
7356	5753		JMP I OUTDG	
7357	0260	C260,	0260	
7360	0000	DIVTWO,	0	/DIVIDE BY TWO IE.
7361	7110		CLL RAR	/ROTATE RIGHT
7362	3345		DCA OUT	/TEMPORARY STORAGE
7363	1045		TAD HORDER	
7364	7010		RAR	
7365	3045		DCA HORDER	
7366	1046		TAD MIDDLE	
7367	7010		RAR	
7370	3046		DCA MIDDLE	
7371	1047		TAD LORDER	
7372	7010		RAR	
7373	3047		DCA LORDER	
7374	1345		TAD OUT	
7375	5760		JMP I DIVTWO	
		/FLOATING POINT INPUT		
		*7400		
7400	0000	FLINTP,	0	
7401	7240		CLA CMA	/INITIALIZE "PERIOD SWITCH"
7402	3314		DCA PRSW	
7403	3061		DCA DSWIT	
7404	4717		JMS I DPCVPT	/7777 = NO PERIOD
7405	7200		CLA	
7406	1060		TAD CHAR	
7407	1313		TAD PER	
7410	7640		SZA CLA	
7411	5220		JMP FIG01	/PERIOD FOUND
7412	1314		TAD PRSW	/SECOND PERIOD
7413	7650		SNA CLA	/YES, TERMINATE
7414	5222		JMP FIG02	/NO - SET NUMBER OF DIGITS TO 0
7415	3722		DCA I DPN	/SET PERIOD SWITCH TO 0
7416	3314		DCA PRSW	/CONVERT REST OF STRING
7417	5720		JMP I DPCSPT	

7420	1314	FIG01,	TAD PRSW	/PERIOD READ IN PREVIOUSLY?
7421	7650		SNA CLA	
7422	1722	FIG02,	TAD I DPN	/YES:-NUMBER OF DIGITS IN SER
7423	7041		CMA IAC	/NO
7424	3315		DCA SEXP	
7425	4721		JMS I MSGNPT	/TEST SIGN
7426	1312	FIG03,	TAD C43	
7427	3044		DCA EXPONT	
7430	4407		JMS I FPNT	/NORMALIZE F.P. NUMBER
7431	7000		FNOR	
7432	6052		FPUT FPAC1	/SAVE NUMBER
7433	0000		FEXT	
7434	1060		TAD CHAR	
7435	1311		TAD MINUSE	
7436	7640		SZA CLA	/"E" READ IN?
7437	5252		JMP ENDFI	/NO
7440	4717		JMS I DPCVPT	/YES - CONVERT DECIMAL EXPONE
7441	4721		JMS I MSGNPT	/TEST SIGN
7442	1045		TAD HORDER	/EXPONENT TOO LARGE??
7443	7510		SPA	
7444	7001		IAC	
7445	7640		SZA CLA	
7446	5277		JMP EXCESS	/YES
7447	1047		TAD LORDER	/NO:DECIMAL POINT IS
7450	1315		TAD SEXP	/C(SEXP)PLACES TO RIGHT
7451	3315		DCA SEXP	/OF LAST DIGIT

/END OF FLOATING POINT INPUT
/COMPENSATE FOR DECIMAL EXPONENTS

7452	4407	ENDFI,	JMS I FPNT	/RESTORE MANTISSA
7453	5052		FGET FPAC1	
7454	0000		FEXT	
7455	1315		TAD SEXP	
7456	7450		SNA	
7457	5600		JMP I FLINTP	
7460	7700		SMA CLA	
7461	5270		JMP FIG04	
7462	4407		JMS I FPNT	/ . IS TO THE LEFT:
7463	3710		FMPY I PC.10	/TIMES .1000
7464	0000		FEXT	
7465	2315		ISZ SEXP	
7466	5255		JMP ENDFI+3	
7467	5600		JMP I FLINTP	

7470	4407	FIG04,	JMS I FPNT	/. IS TO THE RIGHT: /MULTIPLY BY 10
7471	3304		FMPY TEN	
7472	0000		FEXT	
7473	7240		CLA CMA	
7474	1315		TAD SEXP	
7475	3315		DCA SEXP	
7476	5255		JMP ENDFI+3	
7477	1316	EXCESS,	TAD C3777	
7500	3044		DCA EXPONT	
7501	1316		TAD C3777	
7502	3045		DCA HORDER	
7503	5600		JMP I FLINTP	
7504	0004	TEN,	0004	
7505	2400		2400	
7506	0000		0000	
7507	0000		0000	
7510	7144	PC.10,	C.10	/ .10
7511	7473	MINUSE,	-305	
7512	0043	C43,	0043	
7513	7522	PER,	-256	
7514	0000	PRSW,	0	
7515	0000	SEXP,	0	/CONTAINS DECIMAL EXPONENT
7516	3777	C3777,	3777	
7517	7000	DPCVPT,	DECONV	
7520	7020	DPCSPT,	DECON	
7521	7130	MSGNPT,	MSIGN	
7522	7067	DPN,	DNUMBR	
/OUTPUT THE EXPONENT				
7523	0000	FEXC,	0	
7524	7300		CLA CLL	
7525	1044		TAD EXPONT	
7526	7510		SPA	
7527	7061		CMA IAC CML	
7530	3044		DCA EXPONT	
7531	1367		TAD C253	
7532	7430		SZL	
7533	1370		TAD C255	
7534	4775		JMS I DGPT	
7535	3045		DCA HORDER	
7536	1044		TAD EXPONT	
7537	2045		ISZ HORDER	
7540	1371		TAD M144	
7541	7500		SMA	
7542	5337		JMP .-3	

7543	1372	TAD C144
7544	3044	DCA EXPONT
7545	7040	CMA
7546	1045	TAD HORDER
7547	7440	SZA
7550	4775	JMS I DGPT
7551	3045	DCA HORDER
7552	1044	TAD EXPONT
7553	2045	ISZ HORDER
7554	1373	TAD M12
7555	7500	SMA
7556	5353	JMP .-3
7557	1374	TAD C12
7560	3047	DCA LORDER
7561	7240	CLA CMA
7562	1045	TAD HORDER
7563	4775	JMS I DGPT
7564	1047	TAD LORDER
7565	4775	JMS I DGPT
7566	5723	JMP I FEXC

7567	7773	C253,	0253-260
7570	0002	C255,	255-253
7571	7634	M144,	7634
7572	0144	C144,	0144
7573	7766	M12,	7766
7574	0012	C12,	0012
7575	7353	DGPT,	OUTDG

BEXP	7324
CARRTN	7341
CHAR	0060
CHE	7343
C.10	7144
C12	7574
C144	7572
C253	7567
C255	7570
C260	7357
C3777	7516
C43	7512
DECON	7020
DECONV	7000
DGPT	7575
DIGIT	7065
DIVTWO	7360
DNUMBR	7067
DPCSPT	7520

DPCVPT	7517	MINUSE	7511
DPN	7522	MINUS7	7325
DPI	7334	MIN9	7141
DSWIT	0061	MRBOUT	7176
DJBLAD	7107	MSGNPT	7521
ENDFI	7452	MSIGN	7130
EXCESS	7477	MSNPT	7332
EXPONT	0044	MULT10	7042
FEAC	7523	MULT2	7070
FEXPPT	7337	M10PT	7335
FG01	7242	M12	7573
FG02	7246	M144	7571
FG03	7263	M2PT	7336
FG04	7270	OPUT	6776
FG05A	7274	OUT	7345
FG06	7301	OUTDG	7353
FG06A	7304	OUTPUT	7174
FG07	7311	PC.10	7510
FIG01	7420	PER	7513
FIG02	7422	PERIOD	7331
FIG03	7426	PLUS	7140
FIG04	7470	PLUS12	7142
FLINTP	7400	PRCHAR	6767
FLOUTP	7200	PRC.10	7344
FOUR	7326	PRINT	7173
FOUTCN	7217	PRSW	7514
FPAC1	0052	RESTRT	7175
FPNT	0007	SEXP	7515
HORDER	0045	SIGN	7066
INPUT	7150	SMINUS	7330
LFED	6777	SNPT	7333
LNFEED	7342	SPLUS	7327
LORDER	0047	SWIT1	0056
MASK	7143	SWIT2	0057
MIDDL	0046	TEN	7504
MINCR	7177	TENPT	7340
MINUS	7137		

11. DIAGRAMS (Not Applicable)

12 REFERENCES

See Digital-8-5-S.

1. Logical Subroutines, DEC-08-FMIA-D.

2. ABSTRACT

Subroutines for performing the logical operations of inclusive and exclusive OR are presented as a package.

3. REQUIREMENTS

3.1 Storage

Inclusive OR requires 12 (decimal) core locations. Exclusive OR requires 14 (decimal) locations.

3.3 Equipment

Basic PDP-8

4. USAGE

4.1 Loading

The subroutines may be placed in memory by means of the Binary Loader. See Digital-8-2-U-Rim for a complete description of this loader and its use.

4.2 Calling Sequence

Both subroutines are called by a JMS instruction with one argument in the accumulator. The location following the calling JMS contains the address of the second argument. Both subroutines return to the location following that containing the latter address with the result in the AC.

6. DESCRIPTION

6.1 Discussion

These subroutines supplement the AND and CMA hardware instructions in the performance of logical operations. Note that the result of the exclusive OR is the complement of the logical operation termed the "biconditional."

6.2 Examples

Truth tables for these functions are as follows. Depending on the values of corresponding bits in A and B, the associated bit of the result conforms to the following truth tables:

<u>AND</u>			<u>Inclusive OR</u>			<u>Exclusive OR</u>			<u>Biconditional</u>		
A	B	Result	A	B	Result	A	B	Result	A	B	Result
0	0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	1	1	0	1	1	0	1	0
1	0	0	1	0	1	1	0	1	1	0	0
1	1	1	1	1	1	1	1	0	1	1	1

Or for complete data words

	<u>Inclusive OR</u>
A	011 010 111 001
B	010 110 101 100
Result	<u>011 110 111 101</u>

	<u>Exclusive OR</u>
A	011 010 111 001
B	010 110 101 101
Result	<u>001 100 010 100</u>

9. EXECUTION TIME

9.2 Maximum

Execution time is actually fixed for these subroutines. Inclusive OR requires precisely 32.0 microseconds. Exclusive OR requires exactly 46.0 microseconds.

10. PROGRAM

10.4 Program Listing

A listing of both subroutines with INCOR stored in 0200 is as follows:

```

/LOGICAL SUBROUTINES
/ENTER WITH A IN AC
/ADDRESS OF B FOLLOWS CALLING JMS
/RETURN WITH RESULT IN AC TO
/LOCATION FOLLOWING THAT HOLDING ADDRESS

0200      0000      INCOR,      0      /INCLUSIVE OR
0201      3226      DCA      TEMPY1
0202      1600      TAD I INCOR
0203      3227      DCA      TEMPY2
0204      1627      TAD I TEMPY2
0205      7040      CMA
0206      0226      AND      TEMPY1

```

0207	1627		TAD I TEMPY2	
0210	2200		ISZ INCOR	
0211	5600		JMP I INCOR	
0212	0000	EXCOR,	0	/EXCLUSIVE OR
0213	3226		DCA TEMPY1	
0214	1612		TAD I EXCOR	
0215	3227		DCA TEMPY2	
0216	1226		TAD TEMPY1	
0217	0627		AND I TEMPY2	
0220	7041		CIA	
0221	7104		CLL RAL	
0222	1226		TAD TEMPY1	
0223	1627		TAD I TEMPY2	
0224	2212		ISZ EXCOR	
0225	5612		JMP I EXCOR	
0226	0000	TEMPY1,	0	
0227	0000	TEMPY2,	0	

1. Arithmetic Shift Subroutines, DEC-08-FMJA-D.

2. ABSTRACT

Four basic subroutines, shift right and shift left each at both single and double precision, are presented as a package. These are arithmetic shifts.

3. REQUIREMENTS

3.1 Storage

Core storage required for these subroutines is as follows in decimal:

	<u>Shift Left</u>	<u>Shift Right</u>
Single Precision	12	15
Double Precision	24	27

3.3 Equipment

Basic PDP-8

4. USAGE

4.1 Loading

These subroutines may be loaded using the Binary Loader. See Digital-8-2-U-Rim for a complete description of this loader.

4.2 Calling Sequence

All four subroutines are called with $-N$ (the 2's complement form of N) in the accumulator. N is a binary integer specifying the number of bit positions the data words are to be shifted.

In the location following the calling JMS instruction is an address which in the case of the single-precision subroutines is the address of the data to be shifted. In the case of the double-precision subroutines, this address is that of the most significant portion of the data. The least significant portion of the data must be located in the address following that of the most significant portion.

These subroutines will return to the address following that of the calling JMS plus two. Upon exit, the AC will hold the shifted data in the case of single-precision shifts. In the case of double-precision shifts, the AC will hold the most significant portion of the result while the least significant portion of the result will be stored in location LSH.

4.5 Errors

It is possible by specifying too large an N to shift data completely out of a computer word or words in the case of single-precision shifts or double-precision shifts, respectively. These subroutines do not test for this eventuality.

6. DESCRIPTION

6.1 Discussion

These subroutines are arithmetic shift subroutines. By this is meant that in the case of any shift, bits shifted "out" of the register are lost. In the case of left shifts, bits moving into the least significant bit position are always 0. In the case of right shifts, bits moving into the most significant bit position (the sign) bits are 0 if the original data was positive but are 1 if the original data was negative.

6.2 Examples

The following examples illustrate the nature of the single-precision shift process. In each example, a shift of four bits is shown:

		<u>Right</u>	<u>Left</u>
Positive	Data	000 010 100 100	000 000 111 101
	Result	000 000 001 010	001 111 010 000
Negative	Data	111 111 010 100	111 110 000 101
	Result	111 111 111 101	100 001 010 000

6.3 Scaling

Shift right and shift left operations are the fundamental means by which numerical data is scaled in fixed-point computers.

For more information on numerical binary scaling for fixed-point computers, see Application Note 801.

9. EXECUTION TIMES

9.3 Timing Equations

Time needed for a given shift may be calculated from the following equations.

9.3.1 Single-Precision Shift Left - Time in microseconds = $22.4 + 6.4N$

9.3.2 Single-Precision Shift Right - For positive data, time in microseconds = $22.4 + 9.6N$.
For negative data, time in microseconds = $22.4 + 11.2N$.

9.3.3 Double-Precision Shift Left - Time in microseconds = $40.0 + 20.8N$

9.3.4 Double-Precision Shift Right - For positive data, time in microseconds = $40.0 + 24.0N$.
For negative data, time in microseconds = $40.0 + 25.6N$.

10. PROGRAM

10.4 Program Listing

A listing of all four subroutines with SPSL located at 0600 is as follows:

```

/SHIFT RIGHT SHIFT LEFT SUBROUTINES
/SINGLE AND DOUBLE PRECISION
/SHIFTS ARE ARITHMETIC RATHER THAN LOGICAL
/BITS SHIFTED OUT OF REGISTER ARE LOST
/DURING LEFT SHIFTS ZEROS ENTER LEAST SIG. BIT
/DURING POSITIVE RIGHT SHIFTS ZEROS ENTER MOST SIG. BIT
/DURING NEGATIVE RIGHT SHIFTS SIGN IS PROPAGATED
/ENTER WITH -N IN AC
/CALLING SEQUENCE : JMS SPSL OR SPSR OR DPSL OR DPSR
/
/ ADDRESS OF DATA
/ RETURN, RESULT IN AC FOR SINGLE
/ RESULT (MSB) IN AC FOR DOUBLE
/ RESULT (LSB) IN LSH FOR DOUBLE

```

*600

0600	0000	SPSL,	0	
0601	3302		DCA CNTR	/SINGLE PRECISION SHIFT LEFT
0602	1600		TAD I SPSL	
0603	3303		DCA ADDR	
0604	1703		TAD I ADDR	
0605	2200		ISZ SPSL	
0606	7104		CLL RAL	
0607	2302		ISZ CNTR	
0610	5206		JMP .-2	
0611	5600		JMP I SPSL	
0612	0000	SPSR,	0	
0613	3302		DCA CNTR	/SINGLE PRECISION SHIFT RIGHT
0614	1612		TAD I SPSR	
0615	3303		DCA ADDR	
0616	1703		TAD I ADDR	
0617	2212		ISZ SPSR	
0620	7100		CLL	
0621	7510		SPA	
0622	7020		CML	
0623	7010		RAR	
0624	2302		ISZ CNTR	
0625	5220		JMP .-5	
0626	5612		JMP I SPSR	

0627	0000	DPSL,	0	
0630	3302		DCA CNTR	/DOUBLE PRECISION SHIFT LEFT
0631	1627		TAD I DPSL	
0632	3303		DCA ADDR	
0633	1703		TAD I ADDR	
0634	3304		DCA MSH	/MOST SIGNIFICANT HALF
0635	2303		ISZ ADDR	
0636	1703		TAD I ADDR	
0637	3305		DCA LSH	/LEAST SIGNIFICANT HALF
0640	2227		ISZ DPSL	
0641	1305		TAD LSH	/SHIFT LEFT
0642	7104		CLL RAL	
0643	3305		DCA LSH	
0644	1304		TAD MSH	
0645	7004		RAL	
0646	3304		DCA MSH	
0647	2302		ISZ CNTR	
0650	5241		JMP .-7	
0651	1304		TAD MSH	
0652	5627		JMP I DPSL	
0653	0000	DPSR,	0	
0654	3302		DCA CNTR	/DOUBLE PRECISION SHIFT RIGHT
0655	1653		TAD I DPSR	
0656	3303		DCA ADDR	
0657	1703		TAD I ADDR	
0660	3304		DCA MSH	/MOST SIGNIFICANT HALF
0661	2303		ISZ ADDR	
0662	1703		TAD I ADDR	
0663	3305		DCA LSH	/LEAST SIGNIFICANT HALF
0664	2253		ISZ DPSR	
0665	1304		TAD MSH	/SHIFT RIGHT
0666	7100		CLL	
0667	7510		SPA	
0670	7020		CML	
0671	7010		RAR	
0672	3304		DCA MSH	
0673	1305		TAD LSH	
0674	7010		RAR	
0675	3305		DCA LSH	
0676	2302		ISZ CNTR	
0677	5265		JMP .-12	

0700	1304		TAD MSH
0701	5653		JMP I DPSR
0702	0000	CNTR,	0
0703	0000	ADDR,	0
0704	0000	MSH,	0
0705	0000	LSH,	0
ADDR	0703		
CNTR	0702		
DPSL	0627		
DPSR	0653		
LSH	0705		
MSH	0704		
SPSL	0600		
SPSR	0612		

1. Logical Shift Subroutines, DEC-08-FMKA-D.

2. ABSTRACT

Two basic subroutines, shift right at both single and double precision are presented as a package. The shifts are logical in nature.

3. REQUIREMENTS

3.1 Storage

Core storage required for these subroutines is 12 (decimal) locations for single precision and 24 (decimal) locations for double precision.

3.3 Equipment

Basic PDP-8

4. USAGE

4.1 Loading

These subroutines may be loaded using the Binary Loader. See Digital-8-2-U-Rim for a complete description of this loader.

4.2 Calling Sequence

Call with $-N$ (the 2's complement form of N) in the accumulator. N is a binary integer specifying the number of bit positions the data word is to be shifted

In the location following the calling JMS is the address of the data in the case of single precision. For double precision this location contains the address of the most significant portion of the data which must be stored in two consecutive words.

The subroutines return to the location following that containing the data address.

For single precision the result is in the accumulator upon return. For double precision the most significant part of the result is in the accumulator on return while the balance of the result is in location LESTSG.

4.5 Errors

It is quite possible by specifying too large an N effectively to shift data completely out of a computer word or words.

6. DESCRIPTION

6.1 Discussion

These subroutines are logical shift subroutines. It is important to note that there is no difference between arithmetic and logical shifts in the case of left shifts. Consequently only two new subroutines in addition to those described in Digital-8-8-U-Sym are required to supply all logical shifts.

Logical right shifts are defined as those in which bits shifted "out" of the least significant bit position are lost. Bits moving into the most significant bit position are always 0.

6.3 Examples

The following examples illustrate the nature of the single-precision logical right shift. In each example, a shift of four bits is shown.

<u>Data</u>	<u>Result</u>
000 010 111 000	000 000 001 011
111 010 000 000	000 011 101 000

9. EXECUTION TIMES

9.3 Timing Equations

Time needed for a given shift may be calculated from the following equations.

9.3.1 Single-Precision Logical Right Shift - Time in microseconds = $22.4 + 6.4N$.

9.3.2 Double-Precision Logical Right Shift - Time in microseconds = $36.8 + 24.0N$.

10. PROGRAM

10.4 Program Listing

A listing of both subroutines with LSRSP located in 0200 is as follows:

```

/SINGLE PRECISION
/LOGICAL SHIFT RIGHT SUBROUTINES
/SINGLE AND DOUBLE PRECISION
/ENTER WITH -N IN AC
/DCA ADDRESS FOLLOWS CALLING JMS
/RETURN WITH DATA IN AC
/MOST SIGNIFICANT PART FOR DOUBLE
/LEAST SIG. PART FOR DOUBLE IN LESTSG
0200 0000 LSRSP, 0 /SINGLE PRECISION
0201 3236 DCA TIMES
0202 1600 TAD I LSRSP
0203 3237 DCA COMMUN

```

0204	1637		TAD I COMMUN	
0205	7110		CLL RAR	/SHIFT LOOP
0206	2236		ISZ TIMES	
0207	5205		JMP .-2	
0210	2200		ISZ LSRSP	/EXIT
0211	5600		JMP I LSRSP	
0212	0000	LSRDP,	0	/DOUBLE PRECISION
0213	3236		DCA TIMES	
0214	1612		TAD I LSRDP	
0215	3237		DCA COMMUN	
0216	1637		TAD I COMMUN	
0217	3240		DCA MOSTSG	
0220	2237		ISZ COMMUN	
0221	1637		TAD I COMMUN	
0222	3241		DCA LESTSG	
0223	1240	SHIFT,	TAD MOSTSG	/SHIFT LOOP
0224	7110		CLL RAR	
0225	3240		DCA MOSTSG	
0226	1241		TAD LESTSG	
0227	7010		RAR	
0230	3241		DCA LESTSG	
0231	2236		ISZ TIMES	
0232	5223		JMP SHIFT	
0233	1240		TAD MOSTSG	/EXIT
0234	2212		ISZ LSRDP	
0235	5612		JMP I LSRDP	
0236		TIMES,	0	
0237		COMMUN,	0	
0240		MOSTSG,	0	
0241		LESTSG,	0	

digital

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS