

2650 MICROPROCESSOR APPLICATIONS MEMO

INTRODUCTION

Binary arithmetic routines, like addition, subtraction, multiplication, and division, are often used in microprocessor-based systems. This applications memo provides several suggested examples for processing binary arithmetic routines on the 2650 microprocessor. These examples include:

- **SIGNED BINARY ADDITION/SUBTRACTION**
Two-byte operands giving a two-byte result.
- **UNSIGNED BINARY MULTIPLICATION**
One-byte operands giving a two-byte result.
Two-byte operands giving a four-byte result.
- **SIGNED BINARY MULTIPLICATION**
One-byte operands giving a two-byte result.
Two-byte operands giving a four-byte result.
- **BINARY DIVISION – UNSIGNED AND SIGNED**
Two-byte dividend and quotient with one-byte divisor and remainder.

In these examples, emphasis is placed on minimizing program memory requirements rather than on processing speed. The different branch instructions and the indexing features of the Signetics 2650 proved useful in minimizing memory requirements.

REGISTERS	HARDWARE AFFECTED						
	R0	R1	R2	R3	R1'	R2'	R3'
PSU	F	II	SP				
PSL	CC	IDC	RS	WC	OVF	COM	C
	X	X		X	X		X

RAM REQUIRED (BYTES): 6

ROM REQUIRED (BYTES): 45

EXECUTION TIME: Variable

MAXIMUM SUBROUTINE
NESTING LEVELS: None

ASSEMBLER/COMPILER USED: PIPHASM

1. BINARY ADDITION/SUBTRACTION FOR TWO-BYTE SIGNED INTEGERS

FUNCTION:

Performs the addition or subtraction of two 2-byte signed integers giving a two-byte result.

$(OPR1, OPR1 + 1) +/- (OPR2, OPR2 + 1) \rightarrow RSLT, RSLT + 1$

PARAMETERS:

Input: OPR1, OPR1 + 1 contains augend/subtrahend
OPR2, OPR2 + 1 contains addend/minuend
COM-flag in PSL indicates addition/subtraction:
COM = 0 addition
COM = 1 subtraction

Output: RSLT, RSLT + 1 contains sum/difference.
The condition code CC is set to the proper value of the two byte result.
OPR1, OPR2 and RSLT are MS-bytes.

SPECIAL REQUIREMENTS

None

Refer to Figures 1.1 and 1.2 for flowchart and program listing.

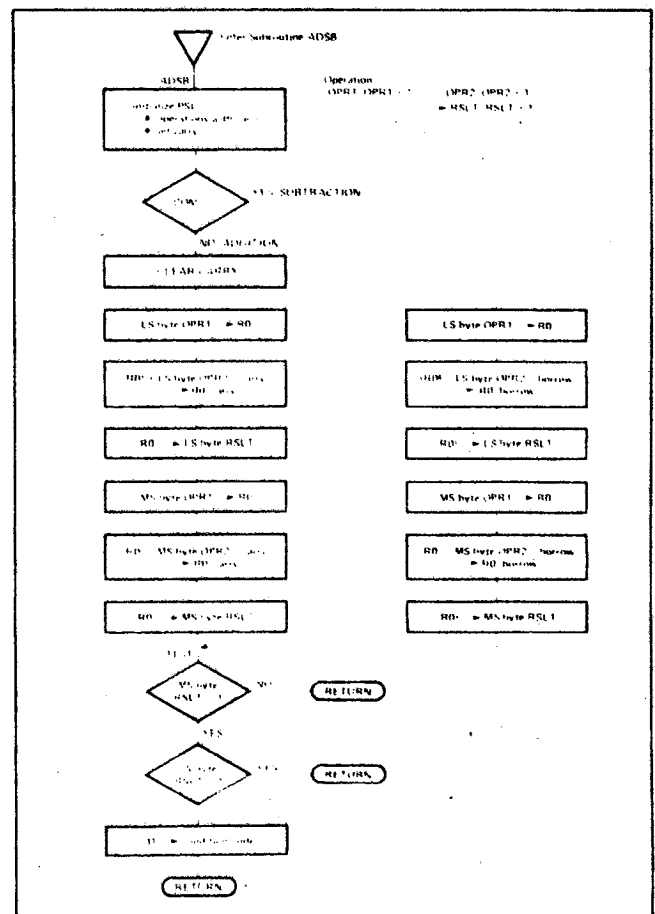


FIGURE 1-1 Flowchart for Double Precision Addition/Subtraction

```

1      + P0760010
2      +*****
3      + BINARY DOUBLE PRECISION ADDITION/SUBTRACTION
4      +*****
5      + OPERATION:
6      + (OPR1,OPR1+1)+/-(OPR2,OPR2+1)-->RSLT,RSLT+1
7      + OPR1,OPR2,RSLT ARE MOST SIG BYTES
8      + COM IN PSL IS USED AS ADD/SUB FLAG
9      + COM=0 IS ADD; COM=1 IS SUBTRACT
10     + AFTER ADD/SUB THE CC,OVF,AND C BITS IN PSL
11     + ARE VALID FOR THE RESULT
12     +
13     + DEFINITION OF SYMBOLS
14     +
15     0000      R0 EQU 0          PROCESSOR REGISTERS
16     0001      R1 EQU 1
17     0002      R2 EQU 2
18     0003      R3 EQU 3
19     0000      CC1 EQU H'00'     PSL: MSB OF CONDITION CODE
20     0040      CC0 EQU H'40'     LSB OF CONDITION CODE
21     0000      WC EQU H'00'     1=WITH,0=WITHOUT CARRY
22     0002      COM EQU H'02'    1=LOGICAL,0=ARITH COMP
23     0001      C EQU H'01'     CARRY/BORROW
24     0000      Z EQU 0          BRANCH COND: ZERO
25     0003      UN EQU 3         UNCONDITIONAL
26     0000      ON EQU 0         ALL BITS ARE 1
27     +
28     ORG H'500' START OF SUBROUTINE
29     +
30     0500 0500 77 09      ADSB PPSL WC+C     ARITH WITH CARRY;SET CARRY
31     0502      05 02      LODI,R1 2         LOAD INDEX REGISTER
32     0504      05 02      TPSEL COM
33     0506      18 0F      BCTR,ON LPSB      BRANCH IF SUBTRACTION
34     0508      75 01      CPSL C          ADDITION,CLEAR CARRY
35     050A 050A 0B 45 2D    LPAD LODA,R0 OPR1,R1,-  BYTE OF FIRST OPERAND TO R0
36     050D      0D 65 2F    ADDA,R0 OPR2,R1     ADD BYTE OF SECOND OPERAND
37     0510      CD 65 31    STRA,R0 RSLT,R1     STORE RESULT
38     0513      59 75      BRNR,R1 LPAD     BRANCH IF NOT DONE
39     0515      1B 0B      BCTR,UN TEST
40     0517 0517 0D 45 2D    LPSB LODA,R0 OPR1,R1,-  BYTE OF FIRST OPERAND TO R0
41     051A      AD 65 2F    SUBA,R0 OPR2,R1     SUB BYTE OF SECOND OPERAND
42     051D      CD 65 31    STRA,R0 RSLT,R1     STORE RESULT
43     0520      59 75      BRNR,R1 LPSB     BRANCH IF NOT DONE
44     0522 0522 98 00      TEST BCFR,Z RTRN     RETURN IF MS BYTE NOT ZERO
45     0524      0C 05 32    LODA,R0 RSLT+1
46     0527      14          RETC,Z          RETURN IF LS BYTE ALSO ZERO
47     0528      75 00      CPSL CC1        SET CC. TO 01 (POSITIVE)
48     052A      77 40      PPSL CC0
49     052C 052C 17          RTRN RETC,UN
50     +
51     052D      OPR1 RES 2         LOCATION OF: FIRST OPERAND
52     052F      OPR2 RES 2         SECOND OPERAND
53     0531      RSLT RES 2        RESULT
54     END

```

FIGURE 1-2

2. BINARY MULTIPLICATION FOR ONE-BYTE UNSIGNED INTEGERS

FUNCTION:

One byte by one byte multiplication for unsigned integers, giving a two-byte result.

$(OPR1) \times (OPR2) \rightarrow RSLT, RSLT + 1$

PARAMETERS:

Input: OPR1 contains multiplier
OPR2 contains multiplicand

Output: RSLT contains high-order product-byte.
RSLT + 1 contains low-order product-byte.

SPECIAL REQUIREMENTS:

None

Refer to Figures 2.1 and 2.2 for flowchart and program listing.

REGISTERS	HARDWARE AFFECTED						
	R0	R1	R2	R3	R1'	R2'	R3'
PSU	F	II	SP				
PSL	CC	IDC	RS	WC	OVF	COM	C
	X	X		X	X		X

RAM REQUIRED (BYTES): 4

ROM REQUIRED (BYTES): 29

EXECUTION TIME: Variable

MAXIMUM SUBROUTINE NESTING LEVELS: None

ASSEMBLER/COMPILER USED: PIPHASM

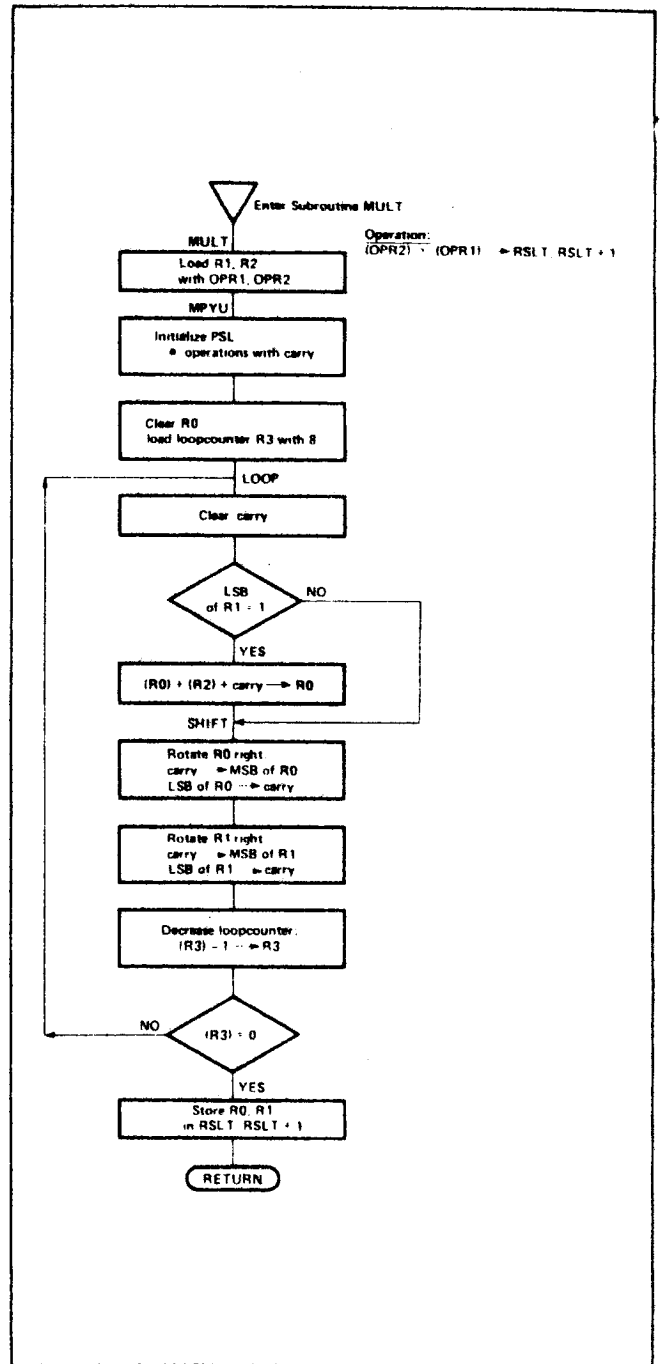


FIGURE 2-1 Flowchart for Unsigned Multiplication (One-Byte Operands; Two-Byte Result)

```

1          *          PD760030          *
2          *          *          *
3          *          *          *
4          *          *          *
5          *          *          *
6          *          *          *
7          *          *          *
8          *          *          *
9          *          *          *
10         *          *          *
11         *          *          *
12         *          *          *
13         *          *          *
14         *          *          *
15         *          *          *
16         *          *          *
17         *          *          *
18         *          *          *
19         *          *          *
20         *          *          *
21         *          *          *
22         *          *          *
23         *          *          *
24         *          *          *
25         *          *          *
26         *          *          *
27         *          *          *
28         *          *          *
29         *          *          *
30         *          *          *
31         *          *          *
32         *          *          *
33         *          *          *
34         *          *          *
35         *          *          *
36         *          *          *
37         *          *          *
38         *          *          *
39         *          *          *
40         *          *          *
41         *          *          *
42         *          *          *
43         *          *          *
44         *          *          *
45         *          *          *
46         *          *          *
47         *          *          *
48         *          *          *
49         *          *          *
50         *          *          *
51         *          *          *
52         *          *          *
53         *          *          *
54         *          *          *
55         *          *          *
56         *          *          *
57         *          *          *

```

14	0000	R0	EQU	0	
15	0001	R1	EQU	1	
16	0002	R2	EQU	2	
17	0003	R3	EQU	3	
18	0001	R4	EQU	1	
19	0002	R5	EQU	2	
20	0003	R6	EQU	3	
21	0003	UN	EQU	3	UNCONDITIONAL BRANCHING
22	0000	ON	EQU	0	
23	0002	LT	EQU	2	
24	0000	Z	EQU	0	
25	0001	P	EQU	1	
26	0002	N	EQU	2	
27	0008	MC	EQU	8	
28	0001	C	EQU	1	
29	0040	F	EQU	H'40'	
30	0004	OVF	EQU	4	
31	0002	COM	EQU	2	

```

32         *          *          *
33         *          *          *
34         *          *          *
35         *          *          *
36         *          *          *
37         *          *          *
38         *          *          *
39         *          *          *
40         *          *          *
41         *          *          *
42         *          *          *
43         *          *          *
44         *          *          *
45         *          *          *
46         *          *          *
47         *          *          *
48         *          *          *
49         *          *          *
50         *          *          *
51         *          *          *
52         *          *          *
53         *          *          *
54         *          *          *
55         *          *          *
56         *          *          *
57         *          *          *

```

36	0500	OPR1	RES	2	
37	0502	OPR2	RES	2	
38	0504	RSLT	RES	4	

```

42         *          *          *
43         *          *          *
44         *          *          *
45         *          *          *
46         *          *          *
47         *          *          *
48         *          *          *
49         *          *          *
50         *          *          *
51         *          *          *
52         *          *          *
53         *          *          *
54         *          *          *
55         *          *          *
56         *          *          *
57         *          *          *

```

43	0600	0600	0D 05 00	MULT	LODA,R1	OPR1	GET OPERAND IN R1
44	0603		0E 05 02		LODA,R2	OPR2	GET OPERAND IN R2
45	0606	0606	77 08	MPTU	PSSL	MC	ARITH
46	0608		20		EORZ	R0	CLEAR R0
47	0609		07 08		LODI,R3	8	LOAD LOOP COUNTER R3
48	060B	060B	75 01	LOOP	CPSL	C	CLEAR CARRI
49	060D		F5 01		TMI,R1	H'01'	
50	060F		98 01		BCFR,ON	SHFT	SKIP ADDITION IF LSB R1=0
51	0611		82		ADDZ	R2	ADD MULTIPLICAND TO PARTIAL PROD
52	0612	0612	50	SHFT	RRR,R0		ROTATE PARTIAL PROD AND MULTIPLIER
53	0613		51		RRR,R1		
54	0614		FB 75		BDRR,R3	LOOP	BRANCH TO LOOP IF NOT READY
55	0616		CC 05 04		STRA,R0	RSLT	SAVE RESULT IN RESULT AREA
56	0619		CD 05 04		STRA,R1	RSLT+1	SAVE RESULT IN RESULT AREA
57	061C		17		RETC,UN		RETURN TO MAIN PROGRAM

FIGURE 2-2

3. BINARY MULTIPLICATION FOR TWO-BYTE UNSIGNED INTEGERS

FUNCTION:

Two byte by two byte multiplication for unsigned integers, giving a four byte result.

$$(OPR2, OPR2 + 1) \times (OPR1, OPR1 + 1) \longrightarrow RSLT, RSLT + 1, RSLT + 2, RSLT + 3$$

PARAMETERS:

Input: (OPR1, OPR1 + 1) contains multiplier
(OPR2, OPR2 + 1) contains multiplicand

Output: RSLT, RSLT + 1, RSLT + 2, RSLT + 3 contains product.
OPR1, OPR2, and RSLT are most-significant bytes.

SPECIAL REQUIREMENTS:

None

Refer to Figures 3.1 and 3.2 for flowchart and program listing.

HARDWARE AFFECTED							
REGISTERS	R0	R1	R2	R3	R1'	R2'	R3'
	X	X		X			
PSU	F	II	SP				
PSL	CC	IDC	RS	WC	OVF	COM	C
	X	X		X	X		X

RAM REQUIRED (BYTES): 8

ROM REQUIRED (BYTES): 57

EXECUTION TIME: Variable

MAXIMUM SUBROUTINE NESTING LEVELS: None

ASSEMBLER/COMPILER USED: PIPHASM

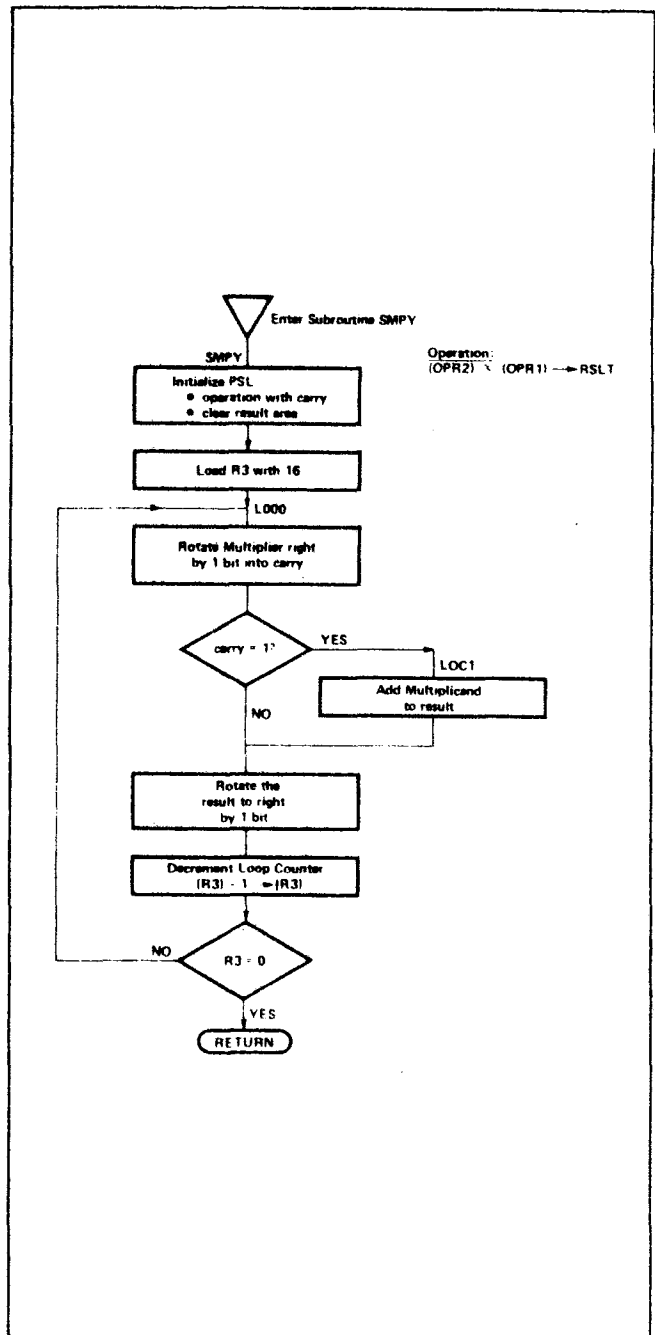


FIGURE 3-1 Flowchart for Unsigned Multiplication (Two-Byte Operands; Four-Byte Result)

```

58          + PD760031          +
59          +*****+
60          + BINARY MULTIPLICATION FOR 2 TWO-BYTE INTEGERS
61          +*****+
62          +
63          + MULTIPLIER IS IN OPR1 , OPR1+1
64          + MULTIPLICAND IS IN OPR2 ,OPR2+1
65          + RESULT WILL BE IN RSLT ,RSLT+1 ,RSLT+2 ,RSLT+3
66          +   ORG H'790'
67          +
68 0790 0790 77 08      SMPY   PPSL   WC           SET MODE
69 0792          20      EORZ   R0           CLEAR RESULT
70 0793          CC 05 04 STRA,R0 RSLT   CLEAR RESULT +1
71 0796          CC 05 05 STRA,R0 RSLT+1 CLEAR RESULT +1
72 0799          07 10   LODI,R3 16        LOAD COUNT
73 079B 079B 05 FE      LOOO   LODI,R1 -2    TO GET 254
74 079D          75 01   CPSL   C           CLEAR CARRY
75 079F 079F 0D 64 02   LOCD   LODA,R0 OPR1-256+2,R1 FOR INDEXING INTO OPR1
76 07A2          50      RRR,R0           ROTATE RIGHT WITH C
77 07A3          CD 64 02 STRA,R0 OPR1-256+2,R1
78 07A6          D9 77   BIRR,R1 LOCD    ROTATE 2ND TIME
79          +           THIS ROTATES MULTIPLIER BY 1 BIT TO GET THE LSB
80          +           INTO CARRY
81 07A8          20      EORZ   R0           CLEAR R0
82 07A9          D0      RRL,R0           GET CARRY INTO LSB
83 07AA          F8 02   BDRR,R0 LOC1
84 07AC          1B 0D   BCTR,UM LOC4
85          +
86 07AE 07AE 05 02      LOC1  LODI,R1 2        GET INDEX
87 07B0 07B0 0D 65 03   LOC2  LODA,R0 RSLT-1,R1  ADD MULTIPLICAND TO PRODUCT
88 07B3          0D 65 01   ADDA,R0 OPR2-1,R1
89 07B6          CD 65 03   STRA,R0 RSLT-1,R1
90 07B9          F9 75   BDRR,R1 LOC2    FINISH THE ADD
91          +
92          +
93 07BB 07BB 05 FC      LOC4  LODI,R1 -4    ROTATE THE PRODUCT TO RIGHT
94 07BD 07BD 0D 64 08   LOC5  LODA,R0 RSLT-256+4,R1
95 07C0          50      RRR,R0           ROTATE RESULT
96 07C1          CD 64 08   STRA,R0 RSLT-256+4,R1
97 07C4          D9 77   BIRR,R1 LOC5
98 07C6          FB 53   BDRR,R3 LOOO    FINISH THE LOOP
99 07C8          17      RETC,UM

```

FIGURE 3-2

4. BINARY MULTIPLICATION FOR ONE-BYTE SIGNED INTEGERS

FUNCTION:

One byte by one byte multiplication for signed integers giving a two-byte result.

$(OPR1) \times (OPR2) \rightarrow RSLT, RSLT + 1$

The Booth algorithm is used (see Figure 4.1).

PARAMETERS:

Input: OPR1 contains multiplier
OPR2 contains multiplicand

Output: RSLT contains high-order product byte.
RSLT + 1 contains low-order product byte.

SPECIAL REQUIREMENTS:

None

Refer to Figures 4.1 and 4.2 for flowcharts and to Figure 4.3 for program listing.

		HARDWARE AFFECTED							
REGISTERS		R0	R1	R2	R3	R1'	R2'	R3'	
		X	X	X	X				
PSU		F	II	SP					
PSL		CC	IDC	RS	WC	OVF	COM	C	
		X	X		X	X		X	

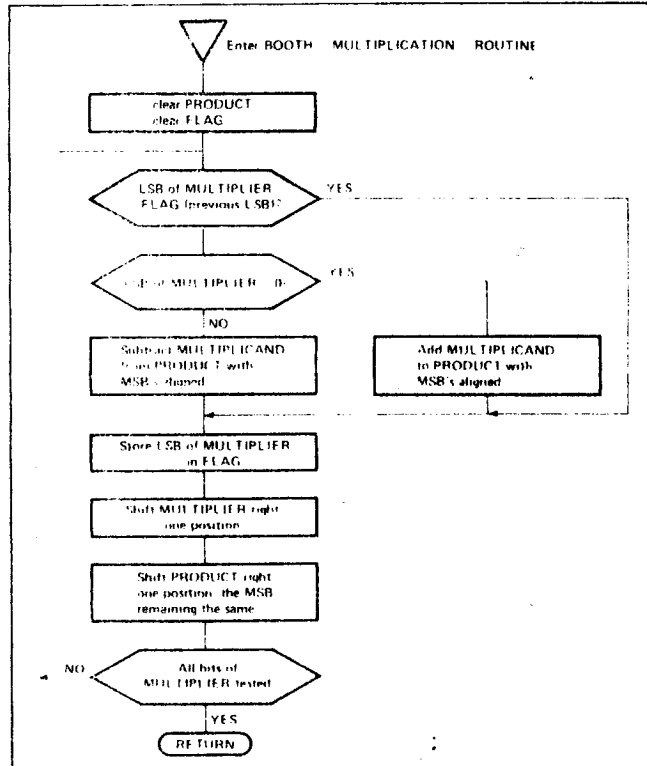


FIGURE 4-1 Flowchart of Booth Algorithm
Multiplicand X Multiplier → Product

RAM REQUIRED (BYTES):	4
ROM REQUIRED (BYTES):	51
EXECUTION TIME:	Variable
MAXIMUM SUBROUTINE NESTING LEVELS:	None
ASSEMBLER/COMPILER USED:	PIPHASM

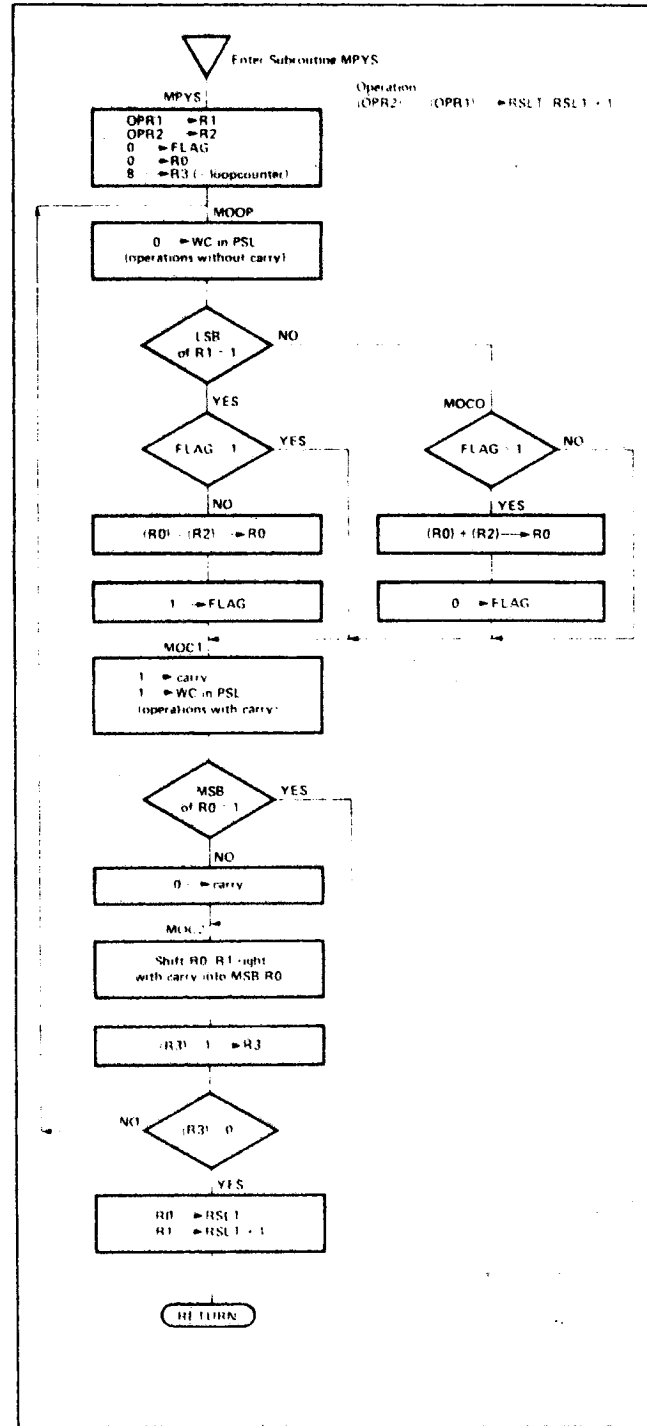


FIGURE 4-2 Flowchart for Signed Multiplication Using Booth Algorithm (One-Byte Operands; Two-Byte Result)

```

100          *      PD760032
101          *
102          * *****
103          * BINARY MULTIPLICATION USING BOOTH-ALGORITHM
104          * FOR 2 ONE-BYTE SIGNED INTEGERS.
105          * *****
106          * FIRST OPERAND IS IN OPR1
107          * SECOND OPERAND IS IN OPR2
108          * PRODUCT WILL BE IN RSLT,RSLT+1
109          *
110          *      ORG  H'000'
110 0000 0000 74 40      MPYS  CPSU  F          CLEAR FLAG IN PSU
111 0002          00 05 00      LODA,R1 OPR1      GET 1ST OPERAND
112 0005          0E 05 02      LODA,R2 OPR2      GET 2ND OPERAND
113 0008          07 08      LODI,R3  8          LOAD LOOP COUNTER R3
114 000A          20          EORZ   R0          CLEAR R0
115 000B 000B 75 00      MOOP  CPSL  WC          CLEAR WC IN PSL
116 000D          F5 01      TMI,R1  H'01'
117 000F          98 09      BCFR,0N MOC0      LSB OF R1 SET?
118 0011          B4 40      TPSU   F          YES
119 0013          18 0C      BCTR,0N MOC1      FLAG =1?
120 0015          A2          SUBZ   R2          NO,SUBTRACT WITHOUT BORROW
121 0016          76 40      PPSU   F          SET FLAG
122 0018          1B 07      BCTR,UN MOC1      BRANCH TO DOUBLE SHIFT
123 001A 001A B4 40      MOC0  TPSU   F          LSB OF R1 WAS 0
124 001C          98 03      BCFR,0N MOC1      FLAG =1?
125 001E          82          ADDZ   R2          YES,ADD WITHOUT CARRY
126 001F          74 40      CPSU   F          CLEAR FLAG
127 0021 0021 77 09      MOC1  PPSL  WC+C      SET C AND WC
128 0023          60          IORZ   R0
129 0024          1A 02      BCTR,N  MOC2      MSB OF R0 SET?
130 0026          75 01      CPSL   C          NO,CLEAR CARRY
131 0028 0028 50          MOC2  RRR,R0      SHIFT R0 R1 RIGHT
132 0029          51          RRR,R1      MSB OF R0 IS SAME
133 002A          FB 5F      BDRR,R3 MOOP      BRANCH TO LOOP IF NOT READY
134 002C          CC 05 04      STRA,R0 RSLT      STORE RESULT
135 002F          CD 05 05      STRA,R1 RSLT+1
136 0032          17          RETC,UN      EXIT SUBROUTINE MPYS
137

```

FIGURE 4-3

5. BINARY MULTIPLICATION FOR TWO-BYTE SIGNED INTEGERS

FUNCTION:

Two byte by two byte multiplication for signed integers giving a four byte result.

$(OPR1, OPR1 + 1) \times (OPR2, OPR2 + 1)$

→ RSLT, RSLT + 1, RSLT + 2, RSLT + 3.

The Booth algorithm (Figure 4.1) is used.

PARAMETERS:

Input: OPR1, OPR1 + 1 contains multiplicand
OPR2, OPR2 + 1 contains multiplier

Output: RSLT, RSLT + 1, RSLT + 2, RSLT + 3 contains product.

OPR1, OPR2, and RSLT are most-significant bytes.

SPECIAL REQUIREMENTS

None

Refer to Figure 5.1 for flowchart and to Figure 5.2 for program listing.

REGISTERS	HARDWARE AFFECTED							
	R0	R1	R2	R3	R1'	R2'	R3'	
PSU	F	II	SP					
PSL	CC	IDC	RS	WC	OVF	COM	C	
	X	X		X	X		X	

RAM REQUIRED (BYTES):	8
ROM REQUIRED (BYTES):	71
EXECUTION TIME:	Variable
MAXIMUM SUBROUTINE NESTING LEVELS:	None
ASSEMBLER/COMPILER USED:	PIPHASM

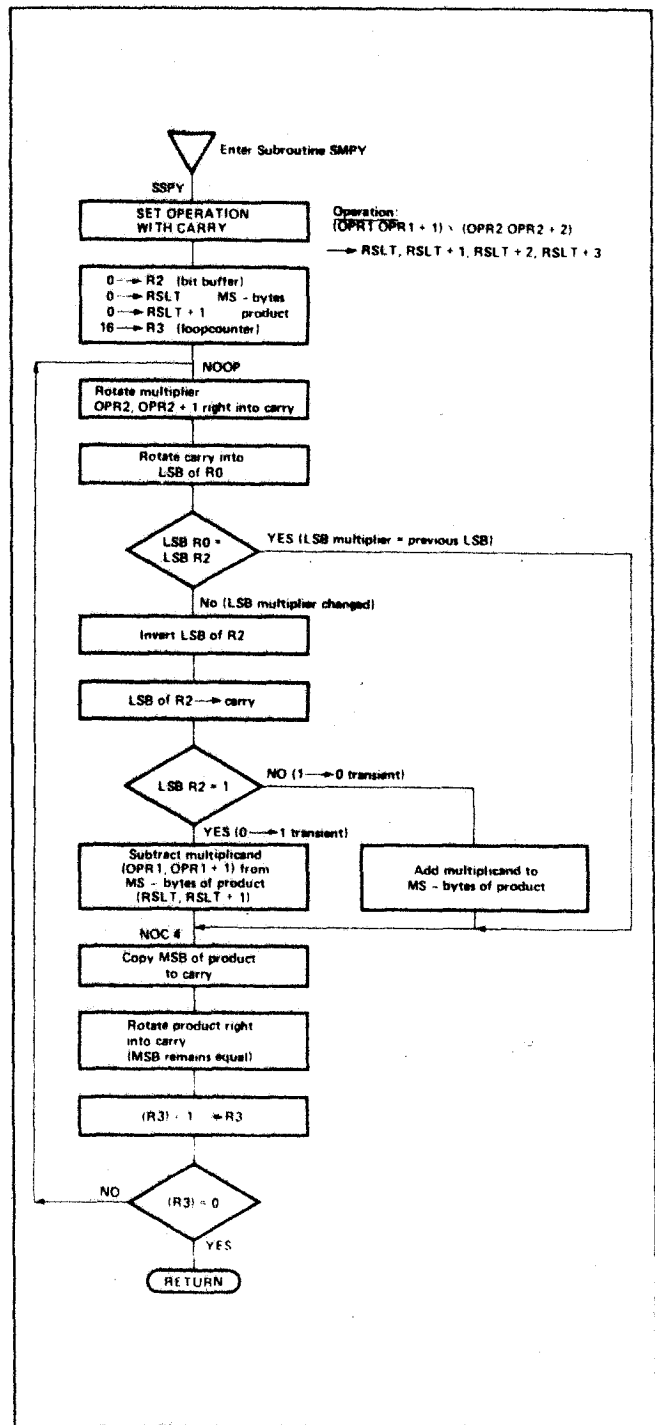


FIGURE 5-1 Flowchart for Signed Multiplication Using Booth Algorithm (Two-Byte Operands; Four-Byte Result)

```

138          * PD760033
139          *****
140          * BINARY MULTIPLICATION FOR TWO BYTE SIGNED INTEGERS
141          *****
142          * MULTIPLICAND IS IN LOCATIONS OPR1,OPR1+1
143          * MULTIPLIER IS IN LOCATIONS OPR2,OPR2+1
144          *
145          * RESULT WILL BE STORED IN RSLT,RSLT+1,RSLT+2,RSLT+3
146          *
147          * AFTER MULTIPLICATION THE MULTIPLICAND IS UNCHANGED
148          * THE MULTIPLIER IS DESTROYED
149          * THE MULTIPLICAND MUST BE UNEQUAL H'0000'
150          *****
151 0833 0833 77 08          SSPT      PPSL      WC          ARITH AND ROTATE WITH C
152 0835          20          EORZ      R0          CLEAR R0
153 0836          C2          STRZ      R2          CLEAR R2
154 0837          CC 05 04   STRA,R0   RSLT      CLEAR 2 MSBYTES OF PRODUCT
155 083A          CC 05 05   STRA,R0   RSLT+1
156 083D          07 10     LODI,R3   16          LOAD LOOP COUNTER R3
157 083F 083F 05 FE          NOOP      LODI,R1   -2          LOAD INDEX REG WITH 254
158 0841 0841 0B 64 04     NOC0     LODA,R0   OPR2-256+2,R1 ROTATE MULTIPLIER
159 0844          50          RRR,R0    INTO CARRY
160 0845          CD 64 04   STRA,R0   OPR2-256+2,R1
161 0848          D9 77     BIRR,R1   NOC0       BRANCH IF NOT DONE
162 084A          20          EORZ      R0          CLEAR R0
163 084B          D0          RRL,R0    ROTATE CARRY IN LSB OF R0
164 084C          22          EORZ      R2          LSB OF R0 BECOMES 1 FOR CHANGE
165 084D          18 19     BCTR,Z    NOC4       BRANCH IF NO CHANGE
166 084F          22          EORZ      R2          INVERT LSB OF R2
167 0850          C2          STRZ      R2          RESTORE NEW R2
168 0851          50          RRR,R0    LSB OF R2 INTO CARRY OR BORROW
169 0852          05 02     LODI,R1   2          LOAD INDEX
170 0854 0854 0D 45 04     NOC1     LODA,R0   RSLT,R1,-   LOAD BYTE OF RSLT IN R0
171 0857          F6 01     TMI,R2   1
172 0859          18 05     BCTR,ON  NOC2       BRANCH TO SUBTRACT IF LSB R2=1
173 085B          8D 65 00   ADDA,R0   OPR1,R1    ADD BYTE MPLCND TO RSLT
174 085E          18 03     BCTR,UN  NOC3
175 0860 0860 AD 65 00     NOC2     SUBA,R0   OPR1,R1    SUB BYTE MPLCND FROM RSLT
176 0863 0863 CD 65 04     NOC3     STRA,R0   RSLT,R1    RESTORE INTERMEDIATE RSLT
177 0866          59 6C     BRNR,R1  NOC1       BRANCH IF ADD SUBTRACT NOT READY
178          *
179 0868 0868 0C 05 04     NOC4     LODA,R0   RSLT
180 086B          D0          RRL,R0
181 086C          04 FC     LODI,R0   -4          LOAD INDEX
182 086E 086E 0D 64 08     NOC5     LODA,R0   RSLT-256+4,R1 FETCH MS BYTE PRODUCT
183 0871          50          RRR,R0    ROTATE RSLT,PROD+1 ETC TO RIGHT
184 0872          CD 64 08   STRA,R0   RSLT-256+4,R1 KEEPING MSB SAME
185 0875          D9 77     BIRR,R1   NOC5       BRANCH IF NOT DONE
186 0877          FB 46     BDRR,R3  NOOP
187 0879          17          RETC,UN
188          END

```

FIGURE 5-2

6. BINARY DIVISION

A. UNSIGNED INTEGERS
TWO-BYTE DIVIDEND; ONE-BYTE DIVISOR

FUNCTION:

Division of a two byte dividend by a one byte divisor, resulting in a two-byte quotient and a one-byte remainder.

$$\frac{(DVDN, DVDN + 1)}{(DVSR)} \rightarrow \left\{ \begin{array}{l} DVDN, DVDN + 1 \text{ (quotient)} \\ R1 \text{ (remainder)} \end{array} \right.$$

PARAMETERS:

Input: DVDN, DVDN + 1 contains dividend
DVSR contains divisor
DVDN is most-significant byte

Output: DVDN, DVDN + 1 contains quotient
R1 contains remainder
DVDN is most-significant byte.
Dividend is destroyed after execution of division.

SPECIAL REQUIREMENTS:

None

Refer to Figure 6.1 for flowchart and to Figure 6.2 for program listing.

HARDWARE AFFECTED							
REGISTERS	R0	R1	R2	R3	R1'	R2'	R3'
	X	X	X	X			
PSU	F	II	SP				
PSL	CC	IDC	RS	WC	OVF	COM	C
	X	X		X	X	X	X

RAM REQUIRED (BYTES):	3
ROM REQUIRED (BYTES):	45
EXECUTION TIME:	Variable
MAXIMUM SUBROUTINE NESTING LEVELS:	None
ASSEMBLER/COMPILER USED:	PIPHASM

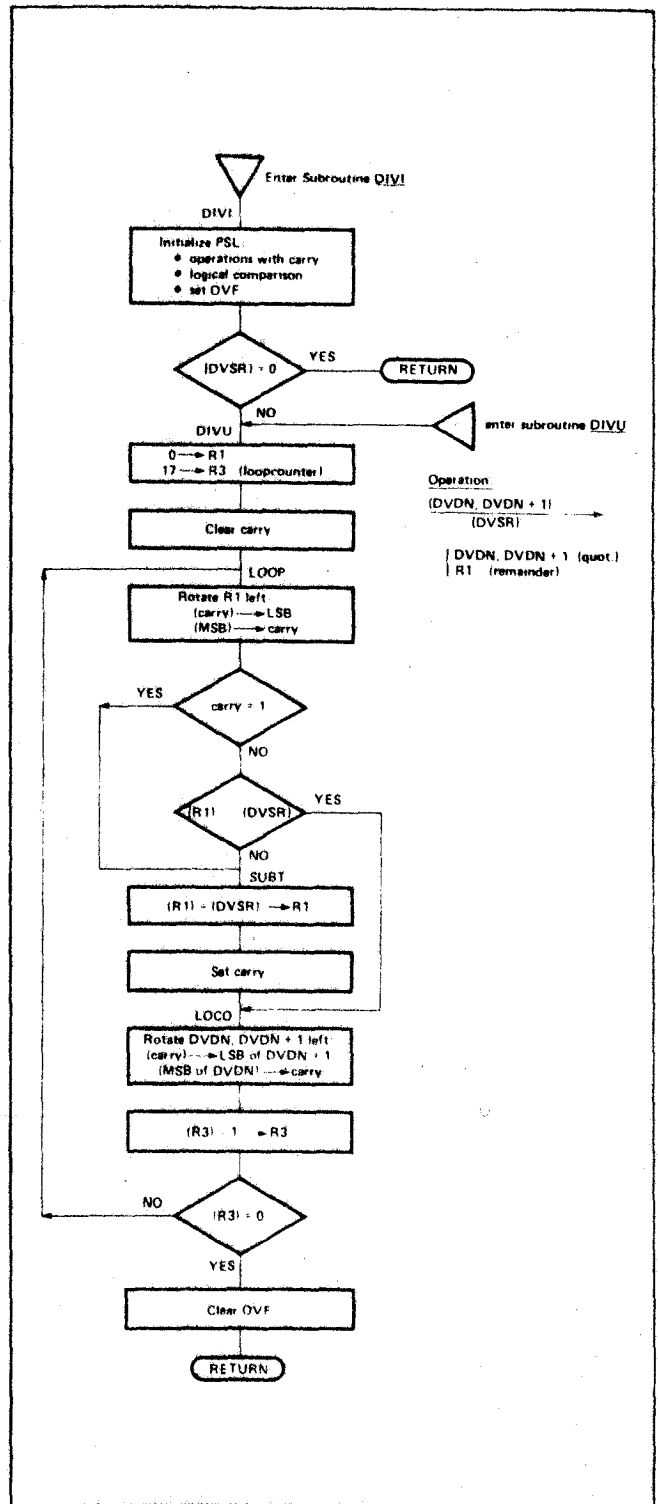


FIGURE 6-1 Flowchart for Unsigned Division (Dividend or Quotient: Two-Bytes; Divisor or Remainder: One-Byte)

```

1          *          PB760040          *
2          *          *          *
3          *          BINARY DIVISIONS FOR INTEGERS          *
4          *          *          *
5          *          DIVIDEND IS IN DVDN,DVDN+1 16 BITS          *
6          *          DIVISOR IS IN DVSR          8 BITS          *
7          *          QUOTIENT WILL BE IN DVDN,DVDN+1 16 BITS          *
8          *          AFTER DIVISION, DIVIDEND WILL BE DESTROYED          *
9          *          R1 WILL HOLD REMAINDER          *
10         *          OVF=1 IMPLIES OVERFLOW          *
11         *          *          *
12         *          *          *
13         *          SYMBOL DEFINITIONS          *
14         0000          R0          EQU          0
15         0001          R1          EQU          1
16         0002          R2          EQU          2
17         0003          R3          EQU          3
18         0001          R4          EQU          1
19         0002          R5          EQU          2
20         0003          R6          EQU          3
21         0003          UN          EQU          3          UNCONDITIONAL BRANCHING
22         0001          C          EQU          1
23         0000          ON          EQU          0
24         0002          LT          EQU          2
25         0000          Z          EQU          0
26         0000          EQ          EQU          0
27         0001          P          EQU          1
28         0002          N          EQU          2
29         0008          WC          EQU          8
30         0004          OVF          EQU          4
31         0002          COM          EQU          2
32         *          *          *
33         *          ORG          H'500'          UNSIGNED DIVISION SUBROUTINE
34         *          *          *
35         0500 0500 77 0E          DIV1          PPSL          WC+OVF+COM          ARITH ROTATE WITH CARRY
36         0502          0C 06 02          LODA,R0          DVSR          FETCH DIVISOR
37         0505          14          RETC,Z          RETURN WITH OVF =1 IF DVSR =0
38         *          *          *
39         0506 0506 05 00          DIVU          LODI,R1          0          CLR R1
40         0508          07 11          LODI,R3          17          LOAD LOOP COUNTER R3
41         050A          75 01          CPSL          C          CLEAR CARRY
42         050C 050C D1          LOOP          RRL,R1          ROTATE CARRY IN LSB OF R1
43         050D          05 01          TPSL          C
44         050F          18 05          BCTR,ON          SUBT          GO TO SUBTRACT IF CARRY =1
45         0511          EB 06 02          COMA,R1          DVSR
46         0514          1A 07          BCTR,LT          LOC0          IF R1<DVSR,NO SUBTRACTION
47         0516 0516 77 01          SUBT          PPSL          C          CLR BORROW
48         0518          AB 06 02          SUBA,R1          DVSR          SUBTR DVSR FROM REMAINDER
49         051B          77 01          PPSL          C          SET CARRY
50         051D 051D 06 02          LOC0          LODI,R2          2          LOAD INDEX REGISTER
51         051F 051F 0E 46 00          LOCI          LODA,R0          DVDN,R2,-          ROTATE QUOTIENT BIT
52         0522          D0          RRL,R0          DVDN,DVDN+1 AND MSB OF
53         0523          CE 66 00          STRA,R0          DVDN,R2          DVDN INTO CARRY
54         0526          5A 77          BRNR,R2          LOC1          BRANCH IF ROTATE NOT READY
55         0528          FB 62          BDRR,R3          LOOP          BRANCH IF DIVISION NOT READY
56         052A          75 04          CPSL          OVF          CLEAR OVF IN PSL
57         052C          17          RETC,UN          RETURN TO MAIN PROGRAM
58         *          *          *

```

FIGURE 6-2

B. SIGNED INTEGERS
TWO-BYTE DIVIDEND; ONE-BYTE DIVISOR

FUNCTION:

Division of a two-byte dividend by a one-byte divisor, resulting in a two-byte quotient and a one-byte remainder.

$$\frac{(DVDN, DVDN + 1)}{(DVSR)} \rightarrow \left\{ \begin{array}{l} DVDN, DVDN + 1 \text{ (quotient)} \\ R1 \text{ (remainder)} \end{array} \right.$$

PARAMETERS:

Input: DVDN, DVDN + 1 contains dividend
 DVSR contains divisor
 DVDN is most-significant byte.

Output: DVDN, DVDN + 1 contains quotient
 R1 contains remainder
 DVDN is most-significant byte.
 Dividend is destroyed after execution of division;
 negative divisor becomes positive

SPECIAL REQUIREMENTS:

Software: Unsigned division subroutine

Refer to Figure 6.3 for flowchart and to Figure 6.4 for program listing.

REGISTERS	HARDWARE AFFECTED							
	R0	R1	R2	R3	R1'	R2'	R3'	
PSU	F	II	SP					
PSL	CC	IDC	RS	WC	OVF	COM	C	
	X	X		X	X	X	X	

RAM REQUIRED (BYTES):	4
ROM REQUIRED (BYTES):	61
EXECUTION TIME:	Variable
MAXIMUM SUBROUTINE NESTING LEVELS:	1
ASSEMBLER/COMPILER USED:	PIPHASM

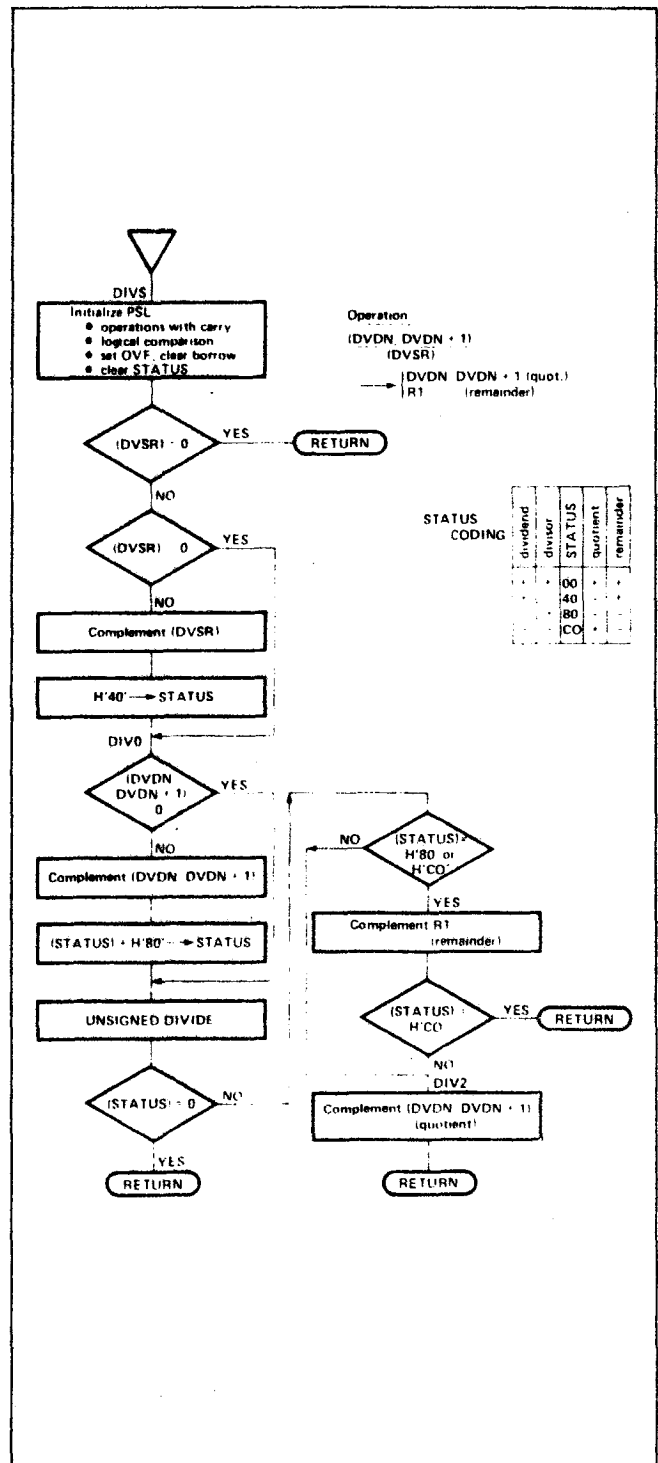


FIGURE 6-3 Flowchart for Signed Division (Dividend & Quotient: 2 Bytes; Divisor & Remainder: 1 Byte)

```

59      * PD760041
60      * *****
61      * SIGNED DIVISION
62      * *****
63      *
64      * NEGATIVE DIVIDEND AND OR DIVISOR ARE COMPLEMENTED
65      * PRIOR TO EXECUTION OF DIVISION
66      *
67      * SIGNS ARE CODED IN STATUS:
68      * STATUS CODING: DVDM DVSR STAT QUOT RMDR
69      *           + + 00 + +
70      *           + - 40 - +
71      *           - + 80 - -
72      *           - - C0 + -
73      * DIVIDEND MUST BE UNEQUAL H'0000' (NO CORRECT OVFL)
74      * NEGATIVE SIGN OF DIVISOR IS LOST AFTER EXECUTION.
75      052D 052D 77 00      DIVS      PPSL      WC+OVFL+C      ARITH ROTATE WITH CARRY ETC
76      052F      20          EORZ      R0
77      0530      C1          STRZ      R1          CLEAR R1
78      0531      0E 06 02    LODA,R2  DVSR      FETCH DIVISOR IN R2
79      0534      14          RETC,Z      RETURN WITH OVFL SET IF DVSR= 0
80      0535      19 06      BCTR,P    DIV0     BRANCH IF DIVISOR >0
81      0537      A2          SUBZ      R2      TAKE 2S COMPLEMENT OF DVSR
82      0538      CC 06 02    STRA,R0  DVSR      RESTORE DIVISOR
83      053B      05 40      LODI,R1  H'40'     LOAD STATUS IN R1
84      053D 053D 0E 06 00    DIV0     LODA,R2  DVDM      FETCH MS BYTE OF DIVIDEND
85      0540      9A 04      BCFR,N   DIV1     BRANCH IF DIVIDEND NOT<0
86      0542      3B 10      BSTR,UN  CMLP      TAKE 2S COMPLEMENT OF DIVIDEND
87      0544      05 00      ADDI,R1  H'00'     UPDATE STATUS
88      0546 0546 CD 06 03    DIV1     STRA,R1  STAT      SAVE STATUS
89      0549      3F 05 06    BSTA,UN  DIVU     CALL UNSIGNED DIVISION
90      054C      0F 06 03    LODA,R3  STAT      LOAD STATUS IN R3
91      054F      14          RETC,Z      RETURN IF BOTH DVDM AND DVSR NOT<0
92      0550      19 07      BCTR,P    DIV2     BRANCH IF DVDM WAS NOT <0 AND DVSR<0
93      0552      77 01      PPSL      C          CLEAR BORROW
94      0554      20          EORZ      R0          CLEAR R0
95      0555      A1          SUBZ      R1      TAKE 2 S COMPLEMENT OF REMAINDER
96      0556      C1          STRZ      R1      RESTORE REMAINDER IN R1
97      0557      D3          RRL,R3           SHIFT R3 LEFT
98      0558      16          RETC,N      RETURN IF BOTH DVDM,DVSR<0
99      0559 0559 3B 01      DIV2     BSTR,UN  CMLP      TAKES 2S COMPL. OF QUOTIENT
100     055B      17          RETC,UN     RETURN TO MAINPROGRAM
101
102
103      * SUBROUTINE TO TAKE 2S COMPL
104      * OF (DVDM,DVDM+1)
105      *
106     055C 055C 77 01      CMLP     PPSL      C          CLEAR BORROW
107     055E      07 02      LODI,R3  2          LOAD INDEX REG
108     0560 0560 20          CMLP     EORZ      R0          CLR R0
109     0561      AF 46 00    SUBA,R0  DVDM,R3,-  COMPLEMENT BYTE
110     0564      CF 66 00    STRA,R0  DVDM,R3    RESTORE RESULT
111     0567      5B 77      BRNR,R3  CMLP     BRANCH IF NOT DONE
112     0569      17          RETC,UN
113
114      0600      DVDM     RES      2          DIVIDEND AND QUOTIENT
115      0602      DVSR     RES      1          DIVISOR
116      0603      STAT     RES      1          STATUS REG
117
118      END

```

FIGURE 6-4