



DATA GENERAL  
CORPORATION

Southboro,  
Massachusetts 01772  
(617) 485-9100

PROGRAM

Double Precision Signed Multiply

TAPES

ASCII Source: 090-000016

ABSTRACT

This routine multiplies two double precision, fixed point, two's complement numbers to form a quadruple precision, two's complement product.

1. REQUIREMENTS

1.1 Memory

1K or larger alterable memory

1.2 Equipment

NOVA central processor

1.3 External Subroutines

Unsigned multiply (.MPYU)

1.4 Other

None

2. OPERATING PROCEDURE

2.1 Calling Sequence

JSR .DMPY  
address of first word of two word multiplier  
return

2.2 Input Format

One double precision operand must be provided in AC0, AC1 (high order, low order). The address of the first word of the second operand must be supplied in the word following the JSR .DMPY.

2.3 Output Format

The quadruple precision product is stored in four consecutive memory locations, highest order to lowest order. AC2 will contain the address of the first, and highest order, word of the result.

2.4 Error Returns

None

2.5 State of Active Registers upon Exit

All accumulators and Carry are destroyed.

2.6 Cautions to User

None

3. DISCUSSION

3.1 Algorithms

The sign of the result is initially determined according to the algebraic rules for multiplication of signed numbers. The absolute values of the operands are used to determine the absolute value of the result. If necessary, this result is negated.

Assume the absolute values of the operands are:

$$2^{*16} * A + B$$

and  $2^{*16} * C + D$

The product of these operands is:

$$2^{*32} * A * C + 2^{*16} * B * C + 2^{*16} * A * D + B * D$$

where the single precision products can be formed using the unsigned multiply routine. Thus four single precision multiplies and three additions are required to compute the result.

3.2 Limitations and Accuracy

The routine is exact.

3.3 Size and Timing

Double precision multiply and double precision divide are supplied as one routine. The size for both routines is 213 (octal) words.

The average execution time is 257  $\mu$  seconds in addition

to four unsigned multiplies at 340  $\mu$  seconds each.  
Total average time is, therefore, 1.62 milliseconds.

### 3.4 References

Write-up 093-000011 describes the double precision divide subroutine which is provided along with .DMPY.

Write-up 093-000015 describes the unsigned multiply routine.

### 3.5 Flow Diagrams

Not applicable

## 4. EXAMPLES AND APPLICATIONS

An ASCII source tape (090-000016) of .DMPY is provided with the NOVA software. This tape should be edited into a user routine that requires double precision multiplication.

## 5. PROGRAM LISTING

A listing of .DMPY follows. No origin has been given, enabling the user to edit this routine anywhere into his program.

```

; SIGNED DOUBLE MULTIPLY
; MULTIPLIES TWO SIGNED, DOUBLE PRECISION NUMBERS

; INPUT:          D1 IN AC0 (HIGH ORDER), AC1 (LOW ORDER)
;                D2 POINTED TO BY WORD AFTER JSK .DMPY

; OUTPUT:         D1*D2 IN STORAGE POINTED TO BY AC2
;                AC2 POINTS TO HIGHEST ORDER

; CALLING SEQUENCE:
;                JSK      .DMPY
;                ADDRESS OF D2
;                RETURN

; DESTROYED:     ALL ACS AND CARRY

; REQUIRES:      .MPYU (UNSIGNED MULTIPLY)

```

```

00000 054076 .DMPY: STA 3,.BC03      ; SAVE RETURN
00001 152400      SUB 2,2        ; CLEAR RESULT FLAG WORD
00002 004045      JSK .BB50      ; FORM ABSOLUTE VALUE OF D1
00003 040077      STA 0,.BC10      ; SAVE ABS(D1)
00004 044100      STA 1,.BC10+1
00005 034076      LDA 3,.BC03      ; RESTORE AC3
00006 010076      ISZ .BC03      ; BUMP PAST ADDRESS CONSTANT
00007 035400      LDA 3,0,3      ; GET ADDRESS OF D2
00010 021400      LDA 0,0,3      ; GET D1
00011 025401      LDA 1,1,3
00012 004045      JSK .BB50      ; FORM ABSOLUTE VALUE OF D2
00013 151200      MOVK 2,2
00014 152500      SUBL 2,2
00015 050110      STA 2,.BC13      ; 0 FOR RESULT MINUS, 1 FOR
; PLUS
00016 040101      STA 0,.BC10+2      ; SAVE ABS(D2)
00017 044102      STA 1,.BC10+3
00020 030100      LDA 2,.BC10+1      ; D1(LOW) * D2(LOW)
00021 006113      JSK 0,.BC30      ; UNSIGNED MULTIPLY
00022 044100      STA 1,.BC10+1      ; STORE LOWEST ORDER OF RESULT
00023 024101      LDA 1,.BC10+2      ; D1(LOW) * D2(HIGH)
00024 006114      JSK 0,.BC31      ; ADD HIGH ORDER TO FIRST CROSS
; PRODUCT
00025 030077      LDA 2,.BC10      ; D1(HIGH)
00026 040077      STA 0,.BC10      ; SAVE HIGH ORDER OF 1ST CROSS
00027 121000      MOV 1,0        ; LOW ORDER TO BE ADDED TO
; SECOND CROSS
00030 024102      LDA 1,.BC10+3      ; D2(LOW) * D1(HIGH)
00031 006114      JSK 0,.BC31      ; SECOND CROSS PRODUCT
00032 044102      STA 1,.BC10+3      ; 2ND LOWEST ORDER OF RESULT
00033 024101      LDA 1,.BC10+2      ; D1(HIGH) * D2(HIGH)
00034 006114      JSK 0,.BC31      ; ADD HIGH ORDER OF CROSS TO
; HIGH ORDER MULTIPLY
00035 030077      LDA 2,.BC10      ; ADD HIGH ORDER OF 1ST CROSS
00036 147022      ADDE 2,1,SEC
00037 101420      INCE 0,0
00040 030102      LDA 2,.BC10+3
00041 034100      LDA 3,.BC10+1      ; GET LOW ORDER RESULTS
00042 014110      DSE .BC13      ; TEST SIGN OF RESULT
00043 000056      JMP .BB53      ; NEGATIVE, - NEGATE RESULT
00044 000066      JMP .BBS4      ; POSITIVE - STORE RESULTS

```

```

; ENTRY AT .BB50 FORMS ABSOLUTE VALUE OF NUMBER
;     IN AC0, AC1 AND INCREMENTS AC2

; ENTRY AT .BB56 NEGATES AC0, AC1 AND INCREMENTS AC2

; ENTRY AT .BB51 NEGATES AC0, AC1

```

```

00045 101113 .BB50:  MOVL# 0,0,SNC
00046 001400          JMP 0,3
00047 151400 .BB56:  INC 2,2
00050 124404 .BB51:  NEG 1,1,SZK
00051 100001          COM 0,0,SKP
00052 100400          NEG 0,0
00053 001400          JMP 0,3

```

```

; ENTRY AT .BB52 FORMS ABSOLUTE VALUE OF NUMBER
;     IN AC0, AC1, AC2, AC3

; ENTRY AT .BB53 FORMS QUADRUPLE NEGATE

; ENTRY AT .BB54 STORES THE ACS IN A BLOCK STARTING
;     AT .BC10

; IF .DMPY CALLED THE ROUTINE, CARRY WILL BE ZERO

; IF .DDIV CALLED THE ROUTINE, CARRY WILL BE NON-ZERO

```

```

00054 101113 .BB52:  MOVL# 0,0,SNC
00055 000066          JMP .BB54
00056 174404 .BB53:  NEG 3,3,SZK
00057 150001          COM 2,2,SKP
00060 150464          NEGC 2,2,SZK
00061 124001          COM 1,1,SKP
00062 124464          NEGC 1,1,SZK
00063 100061          COMC 0,0,SKP
00064 100465          NEGC 0,0,SNK
00065 101060          MOVC 0,0

00066 040077 .BB54:  STA 0,.BC10
00067 044100          STA 1,.BC10+1
00070 050101          STA 2,.BC10+2
00071 054102          STA 3,.BC10+3
00072 101002          MOV 0,0,SEC
00073 000136          JMP .BC98          ; RETURN TO DIVIDE ROUTINE
00074 030115 .BB55:  LDA 2,.BC32          ; POINTER TO MULTIPLY RESULTS
                                ; OR DIVIDE REMAINDER
00075 002076          JMP 0,.BC03          ; RETURN

```

00076	000000	.BC03:	0	; SAVE RETURN
	000004	.BC10:	.BLK 4	; SAVE ABS(U),
				; RESULTS OF MULTIPLY,
				; OR REMAINDER OF DIVIDE
	000002	.BC11:	.BLK 2	; SAVE ABS(V)
	000003	.BC12:	.BLK 3	; TEMPORARY STORAGE
00110	000000	.BC13:	0	; SIGNS OF QUO./REMAINDER
				; FLAGS,
				; OR SIGN OF MULTIPLY RESULT
00111	000000	.BC14:	0	; ITERATION COUNT WORD
00112	000041	.BC20:	41	; ITERATION FOR DIVIDE (33)
00113	000214	.BC30:	.MPYU	; UNSIGNED MULTIPLY
00114	000215	.BC31:	.MPYA	; UNSIGNED MULTIPLY AND ADD
00115	000077	.BC32:	.BC10	; POINTER TO RESULTS

```

; SIGNED DOUBLE DIVIDE
; DIVIDES TWO SIGNED, MULTIPLE PRECISION NUMBERS

; INPUT:      DIVIDEND (U), A QUADRUPLE PRECISION
;             TWO'S COMPLEMENT INTEGER,
;             POINTED TO (HIGH ORDER WORD) BY AC2
;             AND OCCUPYING FOUR CONTIGUOUS MEMORY LOCATIONS

;             DIVISOR (V), A DOUBLE PRECISION TWO'S
;             COMPLEMENT NUMBER IN AC0, AC1 (HIGH, LOW)

; OUTPUT:     U0,U1,U2,U3/V0,V1
;             DOUBLE PRECISION, SIGNED QUOTIENT IN AC0, AC1
;             (HIGH, LOW)
;             DOUBLE PRECISION SIGNED (SAME AS DIVIDEND)
;             REMAINDER POINTED TO BY AC2 (HIGH, LOW)

; CALLING SEQUENCE:
;             JSR      .DDIV
;             RETURN

; ERROR CONDITIONS:      IF ABS(U0,U1) >= ABS(V0,V1),
;                         CARRY WILL BE
;                         SET AND RESULTS UNPREDICTABLE

;                         IF THE QUOTIENT HAS AN ABSOLUTE
;                         VALUE >= 2**31,
;                         CARRY WILL BE SET AND
;                         RESULTS UNPREDICTABLE.

; DESTROYED:      ALL ACS AND CARRY

```

```

00116 054076 .DDIV: STA 3,.BC03      ; SAVE RETURN
00117 050077      STA 2,.BC10      ; SAVE AC2 TEMPORARILY
00120 152620      SUBR 2,2      ; CLEAR FOR RESULT SIGN FLAGS
00121 004045      JSR .BB50      ; FORM ABSOLUTE VALUE OF
; DIVISOR
00122 040103 .BC99: STA 0,.BC11      ; SAVE ABS(V)
00123 044104      STA 1,.BC11+1
00124 155120      MOVZL 2,3      ; POSITION DIVISOR SIGN TO BIT
; 14, SET CARRY
00125 030077      LDA 2,.BC10      ; RESTORE AC2
00126 021000      LDA 0,0,2      ; GET DIVIDEND
00127 101112      MOVL# 0,0,SEC
00130 175400      INC 3,3
00131 054110      STA 3,.BC13      ; SIGN FLAGS ARE SET
00132 025001      LDA 1,1,2
00133 035003      LDA 3,3,2
00134 031002      LDA 2,2,2
00135 000054      JMP .BB52      ; FORM ABSOLUTE VALUE OF
; DIVIDEND (NOTE CARRY SET)

```



```

00136 030103 .BC98: LDA 2,.BC11      ; GET DIVISOR FOR ERROR TEST
00137 034104      LDA 3,.BC11+1
00140 142433      SUBZ# 2,0,SNC      ; ERROR IF U0,U1 >= V0,V1
00141 000145      JMP .+4
00142 142033      ADCZ# 2,0,SNC
00143 166432      SUBZ# 3,1,SZC
00144 002076      JMP @.BC03      ; ERROR, CARRY IS SET
00145 034112      LDA 3,.BC20      ; +33
00146 054111      STA 3,.BC14      ; STORE LOOP COUNT
00147 101020      MOVZ 0,0      ; CLEAR CARRY
00150 000166      JMP .BC90

00151 166423 .BC89: SUBZ 3,1,SNC      ; SUBTRACT DIVISOR FROM HIGH
                                ; DIVIDEND
00152 142021      ADCZ 2,0,SKP
00153 142420      SUBZ 2,0
00154 000166      JMP .BC90

00155 125100 .BC88: MOVL 1,1
00156 101100      MOVL 0,0
00157 030103      LDA 2,.BC11      ; GET DIVISOR
00160 034104      LDA 3,.BC11+1
00161 142413      SUB# 2,0,SNC      ; WILL DIVISOR GO INTO
                                ; DIVIDEND
00162 000166      JMP .BC90      ; NO
00163 142415      SUB# 2,0,SNR
00164 166412      SUB# 3,1,SZC
00165 000151      JMP .BC89      ; YES
00166 034102 .BC90: LDA 3,.BC10+3
00167 030101      LDA 2,.BC10+2
00170 175100      MOVL 3,3      ; SHIFT DIVIDEND AND QUOTIENT
00171 151100      MOVL 2,2
00172 050101      STA 2,.BC10+2
00173 054102      STA 3,.BC10+3
00174 014111      DSZ .BC14
00175 000155      JMP .BC88      ; NOT DONE YET

00176 030110      LDA 2,.BC13      ; GET FLAGS
00177 151202      MOVR 2,2,SZC
00200 004047      JSR .BB56      ; NEGATE REMAINDER, INC AC2
00201 040077      STA 0,.BC10      ; AND STORE IT
00202 044100      STA 1,.BC10+1
00203 020101      LDA 0,.BC10+2      ; GET QUOTIENT
00204 024102      LDA 1,.BC10+3
00205 115102      MOVL 0,3,SZC      ; TEST FOR >2**31
00206 002076      JMP @.BC03      ; YES, CARRY SET FOR ERROR
00207 151202      MOVR 2,2,SZC
00210 004050      JSR .BB51      ; NEGATE QUOTIENT
00211 101020      MOVZ 0,0      ; CLEAR CARRY
00212 000074      JMP .BB55      ; RETURN

```