DECUS
PROGRAM LIBRARY

DECUS NO.	8-531 A & B
TITLE	'TRIPLE' - 36 BIT PDP-8/E SIMULATOR and 'TRIPLE' 8BAL MACROS
AUTHOR	David M. Kristol
COMPANY	Wilmington, Delaware
DATE	December 9, 1971
SOURCE LANGUAGE	PAL-8

SUBJECT: 36-bit PDP-8 Simulator ('TRIPLE') 8-531A

FROM: David M. Kristol

DATE: 9 December 1971

1. Introduction

1.1 Multiple precision operations on a PDP-8 computer are usually a nuisance. The package that is currently available for 36-bit operations (DECUS 5/8-21) is not convenient to use. Taking the Floating Point Package as an example, a package was written ('TRIPLE') for the PDP-8 (8, 8/I, 8/L, 8/E) computer which simulates a 36-bit PDP-8/E computer without EAE. Extended memory and non-extended memory versions are available. (References to extended memory operations do not apply to the non-extended memory version).

1.2 Instructions are in normal PDP-8 format (12 bits). The AC, MQ, and operands for memory reference instructions are 36-bit. In general, TRIPLE is designed so that conventional PAL coding will effect 36-bit operations. Of course, 3 words must be reserved for each operand.

2. Operation

2.1 Entry

TRIPLE is entered with a JMS. The saved address becomes the simulator PC. The following other registers are set:

Real		Simulated
Link	⟶	Link
AC	⟶	low order AC. High 24 bits set to Ø.
DF	⟶	IF (i.e., IF set to calling field)
	⟶	DF

2.2 Instructions

2.2.1 The following are legal:

2.2.1.1 All memory reference, including indirect addressing

2.2.1.2 IOT's except special PDP-8/E: SKON, SRQ, GTF, RTF, CAF

2.2.1.3 OPR's: all Group I & II, MQL, MQA, SWP, CLA. Microprogramming is permissable.

2.2.2 Indirect addresses are strictly twelve bits. They may be considered to be the high order 12 bits of 36. The following coding is necessary to increment a pointer through a 36-bit operand array:

```
        .
        .
        .
TAD I POINT
DCA TEMP
STL IAC RAL             /3
TAD POINT-2             /ADD TO ACTUAL POINT
DCA POINT-2             /SAVE
ISZ COUNT               /DONE?
JMP ...                 /NO
        .
        .
        .
```

2.2.3 Accessing 12-bit literals is treacherous. For people with PS/8, there is a set of macros available for use with the 8BAL macro processor (both in DECUS). These permit generation of 36-bit numeric, character, and symbolic literals.

2.2.4 IOT's which normally would alter the AC (e.g., KRB) will only modify the low order 12 bits of TRIPLE's simulated AC. RDF and RIF perform a jam transfer, not an inclusive 'OR'.

2.2.5 OPR's may be micro-programmed using PDP-8/E sequencing to generate the following 36-bit constants in the AC:

CLA STL RAL or CLA IAC		1
CLA STL RTL or CLA IAC CLL RAL		2
CLA IAC STL RAL		3
CLA IAC CLL RTL		4
CLA IAC STL RTL		6
CLA CMA	7777 7777 7777	-1
CLA CMA CLL RAL	7777 7777 7776	-2
CLA CMA CLL RTL	7777 7777 7775	-3
CLA STL RTR	2000 0000 0000	
CLA STL RAR	4000 0000 0000	
CLA IAC STL RTR	6000 0000 0000	
CLA CMA CLL RAR	3777 7777 7777	
CLA CMA CLL RTR	5777 7777 7777	

2.3 Exit

Simulation ends with the detection of a micro-programmed 'HLT', control going to the next instruction (in normal PDP-8 mode). Registers are set as follows:

Simulated		Real
Link	⟶	Link
low order AC	→	AC
IF	⟶	IF
DF	⟶	DF

In addition, the simulated AC, MQ, and Link remain unchanged in their assigned locations.

3. Special Locations

3.1 Page Ø Storage

TRIPLE uses locations Ø-4, 2Ø-31 in its own field for temporary storage:

4	simulated MA
2Ø-22	simulated AC
23	simulated Link
24-26	simulated MQ
27-31	simulated MB (may be used outside TRIPLE, but will be destroyed by TRIPLE)

3.2 Since debugging a simulated program could be a problem, TRIPLE provides a way to interrupt the simulation by setting two locations:

2	12-bit breakpoint address
3	field of breakpoint (ØØNØ)

A debugging program (e.g., ODT) should be used to set these locations, and a breakpoint at location 1, all in the same field as TRIPLE. When the breakpoint occurs, the instruction whose address is in locations 3 and 2 WILL NOT YET HAVE BEEN EXECUTED. The AC and Link print-out will be meaningless. Variables and the simulated AC, MQ, and Link may be examined. The breakpoint may be proceeded from as usual, including multiple "proceeds". To change the breakpoint, change locations 2 and 3.

4. Space Requirements

TRIPLE occupies 64ØØ-7577 of any memory field, as well as the page Ø locations already mentioned. Its entry point is 64ØØ.

Non-extended memory TRIPLE occupies 66ØØ-7577 of any memory field, as well as the page Ø locations already mentioned. Its entry point is 66ØØ.

5. Notes on the Example

5.1 The following example illustrates the use of TRIPLE. The package will work in any memory field. The calling program must reside in the same field as the package when the non-extended memory version is used (usually field Ø). Otherwise, the Data Field must contain the calling field number (standard DEC extended memory linkage).

5.2 The first three instructions call TRIPLE in field 2. The rest of the program is executed by the simulator. Note that standard PAL coding is used without difficulty. Labels of the form 'ZDDD' where D is a digit, are generated by 8BAL literal processing macros (section 2.2.3).

5.3 The program accepts two 36-bit decimal numbers and types the 72-bit product, 36-bit quotient, and 36-bit fractional remainder in octal. Typed numbers are terminated by any non-digit. An example:

```
6*4=ØØØØØØØØØØØØØØØØØØØØØØØØØ3Ø
ØØØØØØØØØØØ1.4ØØØØØØØØØØØ
```

5.4 Note that the multiplication and division subroutines resemble their 12-bit equivalents very closely.

```
                / TEST OF TRIPLE PRECISION PACKAGE

                MQL=7421
                MQA=7501
                SWP=MQL!MQA
                TRIPLE=6400

                        *200
0200  6201               CDF 0
0201  6222               CIF 20
0202  4777'              JMS TRIPLE
0203  6046               TLS
0204  4257  LOOP,        JMS GETNUM          /GET 36-BIT DECIMAL #
0205  0000               0;0;"*              /TYPE '*' AFTER LAST DIGIT
0206  0000
0207  0252
0210  7421               MQL                 /SAVE IN MQ
0211  7501               MQA
0212  3250               DCA R1              /AND R1
0213  4257               JMS GETNUM
0214  0000               0;0;"=
0215  0000
0216  0275
0217  3235               DCA R2
0220  1235               TAD R2
0221  3223               DCA .+2
0222  4776'              JMS MULT            /MULTIPLE #'S
0223  0000               ZBLOCK 3
0226  4775'              JMS OUTPUT          /PRINT HI-ORDER
0227  7501               MQA
0230  4775'              JMS OUTPUT          /AND LOW-ORDER
0231  4774'              JMS CRLF
0232  1250               TAD R1
0233  7421               MQL
0234  4773'              JMS DVI             /DIVIDE FIRST BY SECOND
0235  0000  R2,          ZBLOCK 3
0240  7521               SWP
0241  4775'              JMS OUTPUT          /PRINT MQ
0242  1314  Z001,        TAD Z002
0243  4772'              JMS TYO             /PRINT '.'
0244  1235               TAD R2
0245  3250               DCA R1

0246  7521               SWP
0247  4773'              JMS DVI             /DIVIDE REMAINDER TO GET FRACTION
0250  0000  R1,          ZBLOCK 3
0253  7521               SWP                 /JUST PRINT MQ
0254  4775'              JMS OUTPUT
0255  4774'              JMS CRLF
0256  5204               JMP LOOP

0257  0000  GETNUM,      .-.
0260  7300               CLA CLL
0261  3771'              DCA TEMP
0262  4770' LOOP1,       JMS TYI
```

```
0263  1317  Z004,   TAD Z005
0264  7540          SMA SZA          />9?
0265  5304          JMP OUT          /YES
0266  1322  Z007,   TAD Z008
0267  7510          SPA              /NO.  <0?
0270  5304          JMP OUT          /YES
0271  3767'         DCA TEMP1        /NO.  SAVE DIGIT
0272  1771'         TAD TEMP
0273  7104          CLL RAL
0274  7104          CLL RAL
0275  1771'         TAD TEMP
0276  7104          CLL RAL          /TEMP*10
0277  1767'         TAD TEMP1        /ADD DIGIT
0300  3771'         DCA TEMP
0301  6036          KRB
0302  4772'         JMS TYO          /PRINT NEW DIGIT
0303  5262          JMP LOOP1

0304  7200  OUT,    CLA
0305  1657          TAD I GETNUM     /GET CHAR
0306  4772'         JMS TYO
0307  7125          STL RAL IAC
0310  1255          TAD GETNUM-2
0311  3255          DCA GETNUM-2     /SKIP ARG
0312  1771'         TAD TEMP         /RETURN #
0313  5657          JMP I GETNUM


0314  0000  Z002,   0;0;".
0315  0000
0316  0256
0317  7777  Z005,   -1;-1;-"9
0320  7777
0321  7507
0322  0000  Z008,   0;0;"9-"0
0323  0000
0324  0011
```

```
0367  0430
0370  0433
0371  0425
0372  0440
0373  0630
0374  0446
0375  0400
0376  0600
0377  6400

                      PAGE

0400  0000  OUTPUT,  .-.
0401  3225           DCA TEMP
0402  1254  Z014,    TAD Z015
0403  3222           DCA COUNT        /12 DIGITS
0404  1225           TAD TEMP
0405  7104           CLL RAL
0406  3225           DCA TEMP
0407  1225  OUT1,    TAD TEMP
0410  7006           RTL
0411  7004           RAL
0412  3225           DCA TEMP
0413  1225           TAD TEMP
0414  0257  Z020,    AND Z021
0415  1262  Z026,    TAD Z027
0416  4240           JMS TYO          /PRINT DIGIT
0417  2222           ISZ COUNT
0420  5207           JMP OUT1
0421  5600           JMP I OUTPUT

0422  0000  COUNT,   ZBLOCK 3
0425  0000  TEMP,    ZBLOCK 3
0430  0000  TEMP1,   ZBLOCK 3

0433  0000  TYI,     .-.
0434  6031           KSF
0435  5234           JMP .-1
0436  6036           KRB
0437  5633           JMP I TYI

0440  0000  TYO,     .-.
0441  6041           TSF
0442  5241           JMP .-1
0443  6046           TLS
0444  7200           CLA
0445  5640           JMP I TYO


0446  0000  CRLF,    .-.
0447  1265  Z029,    TAD Z030
0450  4240           JMS TYO
0451  1270  Z035,    TAD Z036
0452  4240           JMS TYO
0453  5646           JMP I CRLF
```

```
0454  7777  Z0015,  7777;7777;7764
0455  7777
0456  7764
0457  0000  Z0021,  0;0;7
0460  0000
0461  0007
0462  0000  Z0027,  0;0;"0
0463  0000
0464  0260
0465  0000  Z0030,  0;0;215
0466  0000
0467  0215
0470  0000  Z0036,  0;0;212
0471  0000
0472  0212
```

```
                        PAGE

0600  0000  MULT,   .-.             /MULT TWO 36 BIT NUMBERS
0601  7300          CLA CLL
0602  1276  Z047,   TAD M45
0603  3225          DCA EAESC       /37 STEPS
0604  5212          JMP MUY2

0605  7420  MUY1,   SNL             /BIT SET?
0606  5211          JMP .+3
0607  7100          CLL             /YES
0610  1600          TAD I MULT      /ADD MULTIPLICAND TO AC
0611  7010          RAR             /ROTATE AC...
0612  7521  MUY2,   SWP             /AND MQ (GETS NEXT BIT IN LINK)
0613  7010          RAR
0614  7521          SWP
0615  2225          ISZ EAESC       /DONE?
0616  5205          JMP MUY1        /NO
0617  3225          DCA EAESC       /YES.  SAVE HIGH ORDER RESULT
0620  7125          STL IAC RAL     /3
0621  1777'         TAD MULT-2
0622  3777'         DCA MULT-2      /SKIP ARGUMENT
0623  1225          TAD EAESC       /RETRIEVE HIGH ORDER
0624  5600          JMP I MULT

0625  0000  EAESC,  ZBLOCK 3


0630  0000  DVI,    .-.             /DIVIDE 72-BIT NUMBER BY 36
0631  3270          DCA DTEMP       /SAVE HIGH ORDER DIVIDEND
0632  1630          TAD I DVI
0633  3273          DCA DIVIS       /SAVE DIVISOR
0634  7125          STL IAC RAL     /3
0635  1226          TAD DVI-2
0636  3226          DCA DVI-2       /SAVE UPDATED RETURN
0637  1276          TAD M45         /37 STEP COUNT
0640  3225          DCA EAESC
0641  1273          TAD DIVIS
0642  7141          CIA CLL
0643  1270          TAD DTEMP
0644  7630          SZL CLA         /DIVIDE CHECK?
0645  5266          JMP DVI2        /YES.  RETURN L<>0
0646  1273  DVI1,   TAD DIVIS       /NO
0647  7041          CIA
0650  1270          TAD DTEMP
0651  7430          SZL             /AC>DIVISOR?
0652  3270          DCA DTEMP       /YES.  SAVE
0653  7701          CLA MQA         /GET MQ
0654  7004          RAL
0655  7421          MQL             /SHIFT MQ LEFT AND RESTORE
0656  1270          TAD DTEMP
0657  7004          RAL
0660  3270          DCA DTEMP       /SAVE SHIFTED AC
0661  2225          ISZ EAESC       /DONE?
0662  5246          JMP DVI1        /NO
```

```
0663  1270          TAD DTEMP       /YES
0664  7010          RAR             /CORRECT REMAINDER
0665  7410          SKP
0666  1270  DVI2,   TAD DTEMP       /RETURN WITH HI ORDER
0667  5630          JMP I DVI       /RETURN

0670  0000  DTEMP,  ZBLOCK 3
0673  0000  DIVIS,  ZBLOCK 3


0676  7777  M45,    7777;7777;7733
0677  7777
0700  7733

0777  0576
            $
```


COUNT	0422	CRLF	0446	DIVIS	0673	DTEMP	0670	DVI	0630
DVI1	0646	DVI2	0666	EAESC	0625	GETNUM	0257	LOOP	0204
LOOP1	0262	MQA	7501	MQL	7421	MULT	0600	MUY1	0605
MUY2	0612	M45	0676	OUT	0304	OUTPUT	0400	OUT1	0307
R1	0250	R2	0235	SWP	7521	TEMP	0425	TEMP1	0430
TRIPLE	6400	TYI	0433	TYO	0440	Z001	0242	Z002	0314
Z004	0263	Z005	0317	Z007	0266	Z008	0322	Z014	0402
Z015	0454	Z020	0414	Z021	0457	Z026	0415	Z027	0462
Z029	0447	Z030	0465	Z035	0451	Z036	0470	Z047	0602

SUBJECT: TRIPLE precision literal macros 8-5318

FROM: David M. Kristol

DATE: 2 March 1972

A set of four 8BAL macros provides literal capabilities for assemblies using the triple precision package 'TRIPLE'. Three have the basic calling form:

```
label   @xLIT:inst <lit>name
```

where
- x — is a one character literal type
- inst — is the instruction using the literal (often TAD)
- lit — is the literal
- name — may be omitted, or, if supplied, may be used later as an operand (see examples below)
- label — is the label for 'inst' (may be omitted)

@NLIT -- Numeric Literals

'lit' may be a single positive or negative 36-bit octal number
Examples:

```
        @NLIT:TAD<-77>      results in the following code:

    ZØ1Ø, TAD ZØ11
          and later in the program,
    ZØ11, 7777;7777;77Ø1
```

```
    PLACE, @NLIT:AND <17777> C1777    results in:

    PLACE, AND C17777
          and later in the program,
   C17777, Ø;1;7777
```

Other instructions referencing C17777 should <u>not</u> use the @NLIT macro

@CLIT -- Character Literals

The first character of 'lit' should be a '-' if the literal value will be negative.

Examples:

```
        @CLIT:TAD<-"9>     is legal
as is   @CLIT:TAD<"9-"Ø>
But,    @CLIT:TAD<-"Ø+"9>  will result in an incorrect
```

literal.

@SLIT -- Symbolic Literals

'lit' is a 12-bit symbolic expression. The expression value is saved in the high-order part of the 36-bit "word".

Example:

```
@SLIT:<TABLE>
DCA POINT
```

The fourth macro, @LITS, causes all literals in the present literal pool to be dumped, and the pool to be purged. It should be used before each new PAGE pseudo-op in PAL8 to force literals onto the same page on which they are referenced. The following 8BAL statement should be the first in the source program, to initialize the @LITS macro:

```
@SSET @LITSET=
```

```
/ MACROS FOR TRIPLE PRECISION PACKAGE
/  THIS LINE MUST APPEAR BEFORE FIRST USE
/  OF ANY OF THE MACRO'S BELOW:
        @SSET @LITSET=
/  TO INITIALIZE LITERAL POOL

LAB     @DEF @SLIT:INST,LIT,CRSYM       SYMBOLIC LITERAL
        @LIT1:INST,LAB,CRSYM
        @SSET @CRSYM=LIT;0;0
SLIT@

        @DEF @LIT1:INST,LAB,PLACE
LAB,    INST PLACE
        @SSET @LITSET=@LITSET,PLACE
LIT1@

LAB     @DEF @NLIT:INST,LIT,CRSYM       NUMERIC LITERAL
        @LIT1:INST,LAB,CRSYM
        @SET @SIGN=0
        @SSET @S=LIT
        @LCS @C=LIT
        @IFNE:@"@C-@"-,2↑
        @SET @SIGN=1
        @LDEL @S            REMOVE SIGN
        @OCT
        @FOUR:T3
        @FOUR:T2
        @FOUR:T1
        @IFEQ:@SIGN,1↑   CHECK FOR -
        @2SCOMP
        @SSET @CRSYM=@T1;@T2;@T3
NLIT@

        @DEF @2SCOMP
        @SET @CARRY=0
        @SET @T3=-@T3
        @IFNE:@T3,1↑
        @SET @CARRY=1
        @SET @T2=-@T2-1+@CARRY
        @SET @CARRY=0
        @IFNE:@T2,1↑
        @SET @CARRY=1
        @SET @T1=-@T1-1+@CARRY
2SCOMP@

        @DEF @FOUR:NAME
        @SET @C=4
        @SET @NAME=0
        @LOOP
        @SET @T=0
        @IFNL:@S,3↑
        @RCS @T=@S
        @RDEL @S
        @SET @T=@"@T-260
        @SET @NAME=@NAME/10+1000*@T
        @SET @C=@C-1
        @IFGT:@C,@
FOUR@
```

```
        @DEF @LITS      PURGING MACRO
        @LOOP
        @IFNL:@LITSET,2↑
        @GET
        @IFEQ:,@
        @SSET @LITSET=
LITS@

        @DEF @GET
        @SSET @T=
        @LDEL @LITSET
        @LOOP
        @LCS @TT=@LITSET
        @IFEQ:@"@TT-@",,4↑
        @SSET @T=@T@TT
        @LDEL @LITSET
        @IFNL:@LITSET,1↑
        @IFEQ:,@
@T,     @@T
        @XEC:   @SET @@T=0      RESET USED NAME
GET@

LAB     @DEF @CLIT:INST,LIT,CRSYM       FOR CHAR'S
        @LIT1:INST,LAB,CRSYM
        @LCS @T=LIT
        @IFEQ:@"@T-@"-,2↑
        @SSET @CRSYM=0;0;LIT
        @SKP:1
        @SSET @CRSYM=-1;-1;LIT
CLIT@
```
