
MMM MMM AAAAAAAAA CCCCCCCCCCC RRRRRRRRRRR 000000000
MMM MMM AAAAAAAAA CCCCCCCCCCC RRRRRRRRRRR 000000000
MMM MMM AAAAAAAAA CCCCCCCCCCC RRRRRRRRRRR 000000000
MMMMMM MMMMMM AAA AAA CCC RRR RRR OOO OOO
MMMMMM MMMMMM AAA AAA CCC RRR RRR OOO OOO
MMMMMM MMMMMM AAA AAA CCC RRR RRR OOO OOO
MMM MMM MMM AAA AAA CCC RRR RRR OOO OOO
MMM MMM MMM AAA AAA CCC RRR RRR OOO OOO
MMM MMM MMM AAA AAA CCC RRR RRR OOO OOO
MMM MMM AAA AAA CCC RRRRRRRRRRR OOO OOO
MMM MMM AAA AAA CCC RRRRRRRRRRR OOO OOO
MMM MMM AAA AAA CCC RRRRRRRRRRR OOO OOO
MMM MMM AAAAAAAAAAAAAAA CCC RRR RRR OOO OOO
MMM MMM AAAAAAAAAAAAAAA CCC RRR RRR OOO OOO
MMM MMM AAAAAAAAAAAAAAA CCC RRR RRR OOO OOO
MMM MMM AAA AAA CCC RRR RRR OOO OOO
MMM MMM AAA AAA CCC RRR RRR OOO OOO
MMM MMM AAA AAA CCC RRR RRR OOO OOO
MMM MMM AAA AAA CCCCCCCCCCC RRR RRR 000000000
MMM MMM AAA AAA CCCCCCCCCCC RRR RRR 000000000
MMM MMM AAA AAA CCCCCCCCCCC RRR RRR 000000000
MMM MMM AAA AAA CCCCCCCCCCC RRR RRR 000000000

FILEID**COMPUT

```
  CCCCCCCC    OOOOOO    MM      MM  PPPPPPPP    UU      UU  TTTTTTTTTT
  CCCCCCCC    OOOOOO    MM      MM  PPPPPPPP    UU      UU  TTTTTTTTTT
CC           OO      OO  MMMM  MMMM  PP      PP  UU      UU      TT
CC           OO      OO  MMMM  MMMM  PP      PP  UU      UU      TT
CC           OO      OO  MM  MM  MM  PP      PP  UU      UU      TT
CC           OO      OO  MM  MM  MM  PP      PP  UU      UU      TT
CC           OO      OO  MM      MM  PPPPPPPP    UU      UU      TT
CC           OO      OO  MM      MM  PPPPPPPP    UU      UU      TT
CC           OO      OO  MM      MM  PP          UU      UU      TT
CC           OO      OO  MM      MM  PP          UU      UU      TT
CC           OO      OO  MM      MM  PP          UU      UU      TT          ....
  CCCCCCCC    OOOOOO    MM      MM  PP          UUUUUUUUUU      TT          ....
  CCCCCCCC    OOOOOO    MM      MM  PP          UUUUUUUUUU      TT          ....


LL             IIIIII      SSSSSSSS
LL             IIIIII      SSSSSSSS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II        SSSSSS
LL               II        SSSSSS
LL               II              SS
LL               II              SS
LL               II              SS
LL               II              SS
LLLLLLLLLL     IIIIII    SSSSSSSS
LLLLLLLLLL     IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  MAC$COMPUT      ARITHMEITIC ROUTINES
0000     2          .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ; FACILITY:     VAX MACRO ASSEMBLER OBJECT LIBRARY
0000    31 ;
0000    32 ; ABSTRACT:
0000    33 ;
0000    34 ; The VAX-11 MACRO assembler translates MACRO-32 source code into object
0000    35 ; modules for input to the VAX-11 LINKER.
0000    36 ;
0000    37 ; ENVIRONMENT:  USER MODE
0000    38 ;
0000    39 ; AUTHOR: Benn Schreiber, CREATION DATE: 20-AUG-78
0000    40 ;
0000    41 ; MODIFIED BY:
0000    42 ;
0000    43 ;       V03-001 MTR0031         Mike Rhodes     10-Apr-1983
0000    44 ;               Change obsolete $MAC_TIRCMDDEF macro reference
0000    45 ;               to $MAC_OBJCODDEF.
0000    46 ;
0000    47 ;       V01.02  RN0005          R. Newland      27-Aug-1979
0000    48 ;               Remove .ALIGN LONG statements and unecessary
0000    49 ;               branch statements.
0000    50 ;
0000    51 ;--
```

```
0000     53          .SBTTL  DECLARATIONS
0000     54 ;
0000     55 ; INCLUDE FILES:
0000     56 ;
0000     57
0000     58 ;
0000     59 ; MACROS:
0000     60 ;
0000     61
0000     62          $MAC_OBJCODDEF                  ;DEFINE OBJECT CODE COMMANDS ETC.
0000     63
0000     64
0000     65 ;
0000     66 ; EQUATED SYMBOLS:
0000     67 ;
0000     68
0000     69 ;
0000     70 ; OWN STORAGE:
0000     71 ;
0000     72
```

```
                     0000     74          .SBTTL  COMPUTATIONAL ROUTINES
                     0000     75
                00000000     76          .PSECT  MAC$RO_CODE_P15,NOWRT,GBL,LONG
                     0000     77
                     0000     78 ;++
                     0000     79 ; FUNCTIONAL DESCRIPTION:
                     0000     80 ;
                     0000     81 ;        THESE ROUTINES PERFORM ARITHMETIC OPERATIONS ON THE VALUES
                     0000     82 ;        STORED ON MAC$AL_VALSTACK.  THE PASS 2 VERSIONS EMIT CODE
                     0000     83 ;        TO THE OBJECT FILE, WHILE THE PASS 1 VERSIONS DO NOT.  THE
                     0000     84 ;        RESULT IS RETURNED ON THE TOP OF THE STACK.
                     0000     85 ;
                     0000     86 ;--
                     0000     87
                     0000     88 P2$ADD::
                     0000     89          $OBJ_CHKBYT #TIR$C_OPR_ADD      ;GENERATE OBJECT COMMAND
                     0006     90
                     0006     91 P1$ARITH_ADD::
          00A0   30  0006     92          BSBW    LD_XQ_AND_STORE         ;LOAD R0 AND R1
       51   50   C0  0009     93          ADDL2   R0,R1                   ;FORM SUM
                     000C     94
                     000C     95          .IF DF CHECK_OVERFLOW
                     000C     96
                     000C     97          BVC     P1$ARITH_SAME           ;BRANCH IF NO OVERFLOW
                     000C     98          BRW     SET_OVERFLOW            ;OVERFLOW--GO FLAG IT
                     000C     99
                     000C    100          .IFF    ;DEFINED CHECK_OVERFLOW
                     000C    101
                  05 000C    102          RSB
                     000D    103
                     000D    104          .IFTF   ;DEFINED CHECK_OVERFLOW
                     000D    105
                     000D    106
                     000D    107 P2$SAME::
                     000D    108          $OBJ_CHKBYT #TIR$C_OPR_NOP      ;GENERATE COMMAND
                     0013    109
                     0013    110 P1$ARITH_SAME::
                  05 0013    111          RSB
                     0014    112
                     0014    113 P2$AND::
                     0014    114          $OBJ_CHKBYT #TIR$C_OPR_AND      ;GENERATE COMMAND
                     001A    115
                     001A    116 P1$ARITH_AND::
              008C   30  001A    117          BSBW    LD_XQ_AND_STORE         ;LOAD R0 AND R1
          50   50   D2  001D    118          MCOML   R0,R0                   ;GET COMPLEMENT
          51   50   CA  0020    119          BICL2   R0,R1                   ;DO THE AND
                  05 0023    120          RSB
                     0024    121
                     0024    122 P2$ASH::
                     0024    123          $OBJ_CHKBYT #TIR$C_OPR_ASH      ;GENERATE COMMAND
                     002A    124
                     002A    125 P1$ARITH_ASH::
              007C   30  002A    126          BSBW    LD_XQ_AND_STORE
                50   D5  002D    127          TSTL    R0                      ;NEG SHIFT COUNT?
                0B   14  002F    128          BGTR    10$                     ;IF GTR NO
                14   13  0031    129          BEQL    30$                     ;IF EQL DO NOT TOUCH SHIFT COUNT
   50   FFFFFFE0 8F   C8  0033    130          BISL2   #^XFFFFFFE0,R0          ;YES--MAKE IT NEG MODULO 32
```

```
                07   11  003A   131          BRB     20$
          50  FFFFFFE0 8F   CA  003C   132 10$:     BICL2   #^XFFFFFFE0,R0          ;MAKE MODULO 32
              51   51   50   78  0043   133 20$:     ASHL    R0,R1,R1                ;DO THE SHIFT
                        05   05  0047   134 30$:     RSB                             ;RETURN TO STORE RESULT
                             0048   135
                             0048   136 P2$NEG::
                             0048   137          $OBJ_CHKBYT #TIR$C_OPR_NEG      ;GENERATE COMMAND
                             004E   138
                             004E   139 P1$ARITH_NEG::
                   0060      30  004E   140          BSBW    LD_XEQ_STORE_0          ;GET THE OPERAND
                51   51      CE  0051   141          MNEGL   R1,R1                   ;NEGATE
                             05  0054   142          RSB                             ;RETURN TO STORE
                             0055   143
                             0055   144 P2$NOT::
                             0055   145          $OBJ_CHKBYT #TIR$C_OPR_COM      ;GENERATE COMMAND
                             005B   146
                             005B   147 P1$ARITH_NOT::
                      54     10  005B   148          BSBB    LD_XEQ_STORE_0          ;GET THE OPERAND
                51   51      D2  005D   149          MCOML   R1,R1                   ;COMPLEMENT IT
                             05  0060   150          RSB                             ;RETURN TO STORE RESULT
                             0061   151
                             0061   152 P2$OR::
                             0061   153          $OBJ_CHKBYT #TIR$C_OPR_IOR      ;GENERATE COMMAND
                             0067   154
                             0067   155 P1$ARITH_OR::
                      40     10  0067   156          BSBB    LD_XQ_AND_STORE         ;LOAD OPERANDS
                51   50      C8  0069   157          BISL2   R0,R1                   ;FORM RESULT
                             05  006C   158          RSB
                             006D   159
                             006D   160 P2$SUB::
                             006D   161          $OBJ_CHKBYT #TIR$C_OPR_SUB      ;GENERATE COMMAND
                             0073   162
                             0073   163 P1$ARITH_SUB::
                      34     10  0073   164          BSBB    LD_XQ_AND_STORE
          51          50     C2  0075   165          SUBL2   R0,R1                   ;FORM RESULT
                             0078   166
                             0078   167          .IFT    ;DEFINED CHECK_OVERFLOW
                             0078   168
                             0078   169 GO_SET_OVF:
                             0078   170          BVS     SET_OVERFLOW            ;BRANCH IF OVERFLOW
                             0078   171
                             0078   172          .IFTF   ;DEFINED CHECK_OVERFLOW
                             0078   173
                             05  0078   174          RSB
                             0079   175
                             0079   176 P2$XOR::
                             0079   177          $OBJ_CHKBYT #TIR$C_OPR_EOR      ;GENERATE COMMAND
                             007F   178
                             007F   179 P1$ARITH_XOR::
                      28     10  007F   180          BSBB    LD_XQ_AND_STORE
          51          50     CC  0081   181          XORL2   R0,R1                   ;FORM RESULT
                             05  0084   182          RSB
                             0085   183
                             0085   184 P2$MUL::
                             0085   185          $OBJ_CHKBYT #TIR$C_OPR_MUL      ;GENERATE COMMAND
                             008B   186
                             008B   187 P1$ARITH_MUL::
```

```
                 1C  10  008B   188          BSBB    LD_XQ_AND_STORE
              51 50  C4  008D   189          MULL2   R0,R1                      ;FORM RESULT
                     0090   190
                     0090   191          .IFT    ;DEFINED CHECK_OVERFLOW
                     0090   192
                     0090   193          BVS     SET_OVERFLOW                 ;BRANCH IF OVERFLOW
                     0090   194
                     0090   195          .IFTF   ;DEFINED CHECK_OVERFLOW
                     0090   196
                  05 0090   197          RSB
                     0091   198
                     0091   199 P2$DIV::
                     0091   200          $OBJ_CHKBYT #TIR$C_OPR_DIV       ;GENERATE COMMAND
                     0097   201
                     0097   202 P1$ARITH_DIV::
                  10 10  0097   203          BSBB    LD_XQ_AND_STORE
                  50 D5  0099   204          TSTL    R0                       ;SEE IF DIVIDING BY ZERO
                  07 12  009B   205          BNEQ    10$                      ;IF NEQ NO--GO DIVIDE
0000'CF    FF 8F  98  009D   206          CVTBL   #-1,W^MAC$GL_VAL3        ;YES--FLAG DIVIDE BY ZERO
                  05 00A3   207          RSB
   51 51 50  C7  00A4   208 10$:    DIVL3   R0,R1,R1                 ;PERFORM OPERATION
                     00A8   209
                     00A8   210          .IFT    ;DEFINED CHECK_OVERFLOW
                     00A8   211
                     00A8   212          BVS     SET_OVERFLOW                 ;BRANCH IF OVERFLOW
                     00A8   213          RSB                                  ;NO OVERFLOW
                     00A8   214 SET_OVERFLOW:
                     00A8   215          INCL    W^MAC$GL_VAL3            ;INDICATE OVERFLOW
                     00A8   216
                     00A8   217          .ENDC   ;DEFINED CHECK_OVERFLOW
                     00A8   218
                  05 00A8   219          RSB
```

```
                     00A9    221 ;++
                     00A9    222 ; FUNCTIONAL DESCRIPTION:
                     00A9    223 ;
                     00A9    224 ;       LD_XQ_AND_STORE IS CALLED TO POP THE FIRST OPERAND INTO
                     00A9    225 ;       R0, AND THE SECOND OPERAND INTO R1.  THE CALLER IS THEN
                     00A9    226 ;       CALLED TO PERFORM THE OPERATION, RETURNING THE RESULT IN
                     00A9    227 ;       R1.  THIS RESULT IS PUSHED ON THE VALUE STACK.
                     00A9    228 ;
                     00A9    229 ;       LD_XEQ_STORE_0 POPS ONE OPERAND FROM THE STACK INTO R1
                     00A9    230 ;       AND THEN CALLS THE CALLER BACK TO PERFORM A UNARY OPERATION
                     00A9    231 ;       SUCH AS NEGATION OR COMPLEMENTATION.
                     00A9    232 ;
                     00A9    233 ;--
                     00A9    234
                     00A9    235 LD_XQ_AND_STORE:
                     00A9    236         $VPOP   R0                      ;GET THE FIRST OPERAND
                     00B1    237
                     00B1    238 LD_XEQ_STORE_0:
                     00B1    239         $VPOP   R1                      ;GET THE SECOND (OR ONLY) OPERAND
                 9E  16  00B9    240         JSB     @(SP)+              ;CALL CO-ROUTINE TO XEQ FUNCTION
                     00BB    241         $VPUSH  R1                      ;STACK RESULT
                     05  00C3    242         RSB
                     00C4    243
                     00C4    244         .END
```

```
EOM$C_ABORT     =  00000003
EOM$C_ERROR     =  00000002
EOM$C_SUCCESS   =  00000000
EOM$C_WARNING   =  00000001
LD_XEQ_STORE_0     000000B1  R       02
LD_XQ_AND_STORE    000000A9  R       02
MAC$AC_VACSTACK    ********    X     02
MAC$CHRBYT         ********    X     02
MAC$GL_VAL3        ********    X     02
OBJ$C_EOM_ABORT =  00000003
OBJ$C_EOM_ERR   =  00000002
OBJ$C_EOM_OK    =  00000000
OBJ$C_EOM_WRN   =  00000001
P1$ARITH_ADD       00000006  RG      02
P1$ARITH_AND       0000001A  RG      02
P1$ARITH_ASH       0000002A  RG      02
P1$ARITH_DIV       00000097  RG      02
P1$ARITH_MUL       0000008B  RG      02
P1$ARITH_NEG       0000004E  RG      02
P1$ARITH_NOT       0000005B  RG      02
P1$ARITH_OR        00000067  RG      02
P1$ARITH_SAME      00000013  RG      02
P1$ARITH_SUB       00000073  RG      02
P1$ARITH_XOR       0000007F  RG      02
P2$ADD             00000000  RG      02
P2$AND             00000014  RG      02
P2$ASH             00000024  RG      02
P2$DIV             00000091  RG      02
P2$MUL             00000085  RG      02
P2$NEG             00000048  RG      02
P2$NOT             00000055  RG      02
P2$OR              00000061  RG      02
P2$SAME            0000000D  RG      02
P2$SUB             0000006D  RG      02
P2$XOR             00000079  RG      02
SYM$F_DEF       =  00000002
SYM$F_REL       =  00000008
SYM$F_UNI       =  00000004
SYM$F_VALIDATE  =  00000010
SYM$F_WEAK      =  00000001
TIR$C_OPR_ADD   =  00000033
TIR$C_OPR_AND   =  00000037
TIR$C_OPR_ASH   =  0000003D
TIR$C_OPR_COM   =  0000003B
TIR$C_OPR_DIV   =  00000036
TIR$C_OPR_EOR   =  00000039
TIR$C_OPR_IOR   =  00000038
TIR$C_OPR_MUL   =  00000035
TIR$C_OPR_NEG   =  0000003A
TIR$C_OPR_NOP   =  00000032
TIR$C_OPR_SUB   =  00000034
TIR$C_STO_L     =  00000016
TIR$C_STO_LW    =  00000016
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
MAC$RO_CODE_P15	000000C4 (196.)	02 (2.)	NOPIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.03	00:00:02.28
Command processing	143	00:00:00.38	00:00:02.10
Pass 1	271	00:00:05.04	00:00:18.48
Symbol table sort	0	00:00:00.43	00:00:01.17
Pass 2	56	00:00:00.97	00:00:03.29
Symbol table output	5	00:00:00.05	00:00:00.59
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	513	00:00:06.91	00:00:27.93

The working set limit was 1200 pages.
34673 bytes (68 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 503 non-local and 4 local symbols.
244 source lines were read in Pass 1, producing 13 object records in Pass 2.
37 pages of virtual memory were used to define 36 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[MACRO.OBJ]MACRO.MLB;1	4
_255DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	8

689 GETS were required to define 8 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:COMPUT/OBJ=OBJ$:COMPUT MSRC$:COMPUT/UPDATE=(ENH$:COMPUT)+LIB$:MACRO/LIB
```

ACTPRI LIS

ARGSCN LIS

BDYSCN LIS

CRFSUB LIS

ACTOPC LIS

ACTSTA LIS

APSECT LIS

CRFDAT LIS

ACTREF LIS

COMPUT LIS