```
LLL               BBBBBBBBBBBB      RRRRRRRRRRRR
LLL               BBBBBBBBBBBBB     RRRRRRRRRRRRR
LLL               BBBBBBBBBBBBB     RRRRRRRRRRRRR
LLL               BBB          BBB  RR           RRR
LLL               BBB          BBB  RRR          RRR
LLL               BBB          BBB  RRR          RRR
LLL               BBB          BBB  RRR          RRR
LLL               BBB          BBB  RRR          RRR
LLL               BBB          BBB  RRR          RRR
LLL               BBBBBBBBBBBBB     RRRRRRRRRRRRR
LLL               BBBBBBBBBBBBB     RRRRRRRRRRRRR
LLL               BBBBBBBBBBBBB     RRRRRRRRRRRRR
LLL               BBB          BBB  RRR   RRR
LLL               BBB          BBB  RRR   RRR
LLL               BBB          BBB  RRR   RRR
LLL               BBB          BBB  RRR      RRR
LLL               BBB          BBB  RRR      RRR
LLL               BBB          BBB  RRR      RRR
LLLLLLLLLLLLLLLL  BBBBBBBBBBBB      RRR         RRR
LLLLLLLLLLLLLLLL  BBBBBBBBBBBBB     RRR         RRR
LLLLLLLLLLLLLLLL  BBBBBBBBBBBBB     RRR         RRR
LLLLLLLLLLLLLLLL  BBBBBBBBBBBBB     RRR         RRR
```

```
_$2

Val
---
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
00F
00F
00F
00F
00F
00F
00F
00F
00F
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
```

```
  GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  MM      MM  EEEEEEEEEE  MM      MM
  GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  MM      MM  EEEEEEEEEE  MM      MM
GG          EE              TT      MMMM  MMMM  EE          MMMM  MMMM
GG          EE              TT      MMMM  MMMM  EE          MMMM  MMMM
GG          EE              TT      MM  MM  MM  EE          MM  MM  MM
GG          EE              TT      MM  MM  MM  EE          MM  MM  MM
GG          EEEEEEEE        TT      MM      MM  EEEEEEEE    MM      MM
GG          EEEEEEEE        TT      MM      MM  EEEEEEEE    MM      MM
GG  GGGGGG  EE              TT      MM      MM  EE          MM      MM
GG  GGGGGG  EE              TT      MM      MM  EE          MM      MM
GG      GG  EE              TT      MM      MM  EE          MM      MM      ....
GG      GG  EE              TT      MM      MM  EE          MM      MM      ....
  GGGGGG    EEEEEEEEEE      TT      MM      MM  EEEEEEEEEE  MM      MM      ....
  GGGGGG    EEEEEEEEEE      TT      MM      MM  EEEEEEEEEE  MM      MM      ....

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  GETMEM  Allocate/deallocate virtual memory
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:  Memory allocation routiens
0000    31 ;
0000    32 ; ABSTRACT:  ALLOCATE AND DEALLOCATE VIRTUAL BLOCK
0000    33 ;
0000    34 ;
0000    35 ; ENVIRONMENT:  VAX Native mode
0000    36 ;
0000    37 ; AUTHOR:  K.D. MORSE, CREATION DATE:  25-APR-77
0000    38 ;
0000    39 ; MODIFIED BY:
0000    40 ;
0000    41 ;       X03.01  JWT0056         Jim Teague              20-Sep-1982
0000    42 ;               Remove restrictions on amount of memory to allocate
0000    43 ;               or deallocate.
0000    44 ;       X01.01          001     B.L. SCHREIBER  9-FEB-1979
0000    45 ;               Correct error in allocation routine.
0000    46 ;       V01.02          008     B.L. SCHREIBER  26-OCT-1979
0000    47 ;               Declare $CRFMSG
0000    48 ;       V01.03          010     B.L. SCHREIBER  13-NOV-1979
0000    49 ;               Word-relative references
0000    50 ;       V01.04                  B.L. SCHREIBER  15-NOV-1979
0000    51 ;               Generalize into GETMEM.
0000    52 ;--
```

```
                     0000     54          .SBTTL  declarations
                     0000     55 ;
                     0000     56 ; Declar macros
                     0000     57 ;
                     0000     58          $libdef                                 ; Declare run-time library error cod
                     0000     59 ;
                     0000     60 ; EQUATED SYMBOLS:
                     0000     61 ;
                     0000     62
           00000000  0000     63 blk$l_addr = 0                                  ; Offset to addr of next block
           00000004  0000     64 blk$l_size = 4                                  ; Offset to size of this block
           00000008  0000     65 mem$k_roundup = 8                               ; Round allocation up to 8 bytes
           00000080  0000     66 mem$k_memexp = 128                              ; Number of pages to expand by
           00010000  0000     67 mem$k_maxblk = mem$k_memexp * 512               ; Largest block that can be allocate
           00158264  0000     68 mem$_blkwthinbl = lib$_badbloadr                ; Block deallocated within deallocat
           0015826C  0000     69 mem$_illblksize = lib$_badblosiz                ; Illegal block size
           00000001  0000     70 mem$_success = 1                                ; Success
                     0000     71
                     0000     72 ;
                     0000     73 ; Own storage:
                     0000     74 ;
                     0000     75
                00000000     76          .PSECT  $OWN$, NOPIC, USR, CON, REL, LCL, NOSHR, NOEXE, RD, WRT, NOVEC
                     0000     77
                     0000     78 mem$l_maxblk::                                  ; Size of expand region request
           00010000  0000     79          .long   mem$k_maxblk                    ; (set up by lbr$open)
                     0004     80 mem$l_memexp::                                  ; Number of pages in expand region r
           00000080  0004     81          .long   mem$k_memexp                    ; (set up by lbr$open)
00000000   00000000  0008     82 new$l_blks::    .LONG   0,0                     ; Addresses of expanded pages
00000000   00000000  0010     83 mem$l_dynmem::  .LONG   0,0                     ; Listhead of dynamic memory
                     0018     84
                     0018     85          .DEFAULT DISPLACEMENT, WORD            ; Use word displacement
```

```
                     0018    87          .SBTTL  LBR$GET_VM      Allocate virtual memory
                     0018    88 ;++
                     0018    89 ; Functional description:
                     0018    90 ;
                     0018    91 ; This routine allocates a block of dynamic memory.  The requested block
                     0018    92 ; size is rounded up to the nearest four bytes.  An error condition is
                     0018    93 ; returned if the block cannot be allocated.
                     0018    94 ; First fit algorithm is used.
                     0018    95 ;
                     0018    96 ; Calling sequence:
                     0018    97 ;
                     0018    98 ;       BSBW    LBR$GET_VM
                     0018    99 ;
                     0018   100 ; Input parameters:
                     0018   101 ;
                     0018   102 ;       R0              Address of longword containing number of bytes to allocate
                     0018   103 ;       R1              Address of longword to store allocated memory address
                     0018   104 ;
                     0018   105 ; Completion codes:
                     0018   106 ;
                     0018   107 ;       success:
                     0018   108 ;               r0 - contains a one
                     0018   109 ;       failure:
                     0018   110 ;               r0 - contains a zero
                     0018   111 ;
                     0018   112 ;
                     0018   113 ; Side effects:
                     0018   114 ;
                     0018   115 ; The dynamic memory list is updated.  More dynamic memory is acquired
                     0018   116 ; if necessary
                     0018   117 ;
                     0018   118 ;--
                     0018   119
                     0018   120
                00000000    121          .PSECT  $CODE$, NOPIC, USR, CON, REL, LCL, NOSHR, EXE, RD, NOWRT, NOVEC
                     0000   122
                     0000   123 ;crf$alblk::
                     0000   124 ;       moval   -(sp),r1                        ; set location to return result
                     0000   125 ;       bsbb    lbr$get_vm                      ; get the memory
                     0000   126 ;       popl    r1                              ; return result in r1
                     0000   127 ;       rsb
                     0000   128
                     0000   129 ;lnk$alloblk::
                     0000   130 lbr$get_vm::
                 1C  BB  0000   131          pushr   #^M<r2, r3, r4>                 ; Save registers
             54  51  D0  0002   132          movl    r1,r4                           ; Save return result address
          53 50  07  C1  0005   133          addl3   #<mem$k_roundup-1>,r0,r3        ; Round up to the nearest
             53  07  CA  0009   134          bicl2   #<mem$k_roundup-1>,r3           ; Multiple of four bytes
                 2E  14  000C   135          bgtr    40$
                     000E   136 ;       bleq    10$                             ; Check for size <= 0
                     000E   137 ;       cmpl    r3,mem$l_maxblk                 ; Check size > maximum
                     000E   138 ;       bleq    40$                             ; Branch on legal size
50   0015826C  8F  D0  000E   139 10$:    movl    #mem$_illblksize,r0             ; Report illegal block size
                 55  11  0015   140          brb     al_blk_exit                     ; Return
                     0017   141 ;
                     0017   142 ; Expand the program region
                     0017   143 ;
```

```
                     0017   144 20$:    $EXPREG_S       mem$l_memexp,new$l_blks; Expand dynamic memory
        3F 50   E9   002A   145          blbc    r0,al_blk_exit                  ; Branch if it failed
                     002D   146 ;
                     002D   147 ; Now insert memory into list.
                     002D   148 ;
51  0008'CF   D0   002D   149 30$:    movl    new$l_blks,r1                   ; Get address of new block
50  0000'CF   D0   0032   150          movl    mem$l_maxblk,r0                 ; Get size of new block
          36   10   0037   151          bsbb    lbr$free_vm                     ; Go insert new blk in list
        30 50   E9   0039   152          blbc    r0,al_blk_exit                  ; Branch if failed to insert new blo
                     003C   153 ;
                     003C   154 ; Search down list for first block >= size requested.
                     003C   155 ;
51  0010'CF   9E   003C   156 40$:    movab   mem$l_dynmem,r1                 ; Get listhead of dynamic memory
      52   51   D0   0041   157 50$:    movl    r1,r2                           ; Set new previous block
      51   62   D0   0044   158          movl    (r2),r1                         ; Get address of free block
           CE   13   0047   159          beql    20$                             ; End of list, go expand memory
  04 A1    53   D1   0049   160          cmpl    r3,blk$l_size(r1)               ; Requested size > block size?
           F2   14   004D   161          bgtr    50$                             ; Yes, keep looking
           10   13   004F   162          beql    60$                             ; Branch on same size
                     0051   163 ;
                     0051   164 ; Take part of this block and link the rest back into the list.
                     0051   165 ;
  04 A1    53   C2   0051   166          subl2   r3,blk$l_size(r1)               ; Subtract off requested size
  08   04 A1   D1   0055   167          cmpl    blk$l_size(r1),#8               ; Did we allocate the bookkeeping wo
           06   15   0059   168          bleq    60$                             ; If so, go pretend it's a good fit.
                     005B   169                                                  ; This can float at most 4 bytes.
  51   04 A1   C0   005B   170          addl2   blk$l_size(r1),r1               ; Get address of requested block
           05   11   005F   171          brb     70$                             ; Return
                     0061   172 ;
                     0061   173 ; Block was perfect fit.  delete it from the list.
                     0061   174 ;
62    61   D0   0061   175 60$:    movl    blk$l_addr(r1),blk$l_addr(r2)   ; Set pointer to next block
      61   7C   0064   176          clrq    blk$l_addr(r1)                  ; Clean up the block
50    01   D0   0066   177 70$:    movl    #mem$_success,r0                ; Set success status code
64    51   D0   0069   178          movl    r1,(r4)                         ; Return address to caller
                     006C   179 al_blk_exit:
      1C   BA   006C   180          popr    #^m<r2, r3, r4>
           05   006E   181          rsb
```

```
                     006F   183          .SBTTL  LBR$FREE_VM     Free virtual memory
                     006F   184 ;++
                     006F   185 ; Functional description:
                     006F   186 ;
                     006F   187 ; This routine deallocates a block of memory and inserts it into a dynamic
                     006F   188 ; Memory list.  The block is zeroed and its size is rounded up to the
                     006F   189 ; Nearest four bytes.  If it is adjacent to another block, the two blocks
                     006F   190 ; are compacted into one.
                     006F   191 ;
                     006F   192 ; Calling sequence:
                     006F   193 ;
                     006F   194 ;       BSBW    LBR$FREE_VM
                     006F   195 ;
                     006F   196 ; Input parameters:
                     006F   197 ;
                     006F   198 ;       R0              Address of longword containing size of block to deallocate
                     006F   199 ;       R1              Address of longword containing address of block to deallocat
                     006F   200 ;
                     006F   201 ; Completion codes:
                     006F   202 ;
                     006F   203 ;       success:
                     006F   204 ;               r0 - contains a one
                     006F   205 ;       failure:
                     006F   206 ;               r0 - contains a zero
                     006F   207 ;
                     006F   208 ; Side effects:
                     006F   209 ;
                     006F   210 ;       NONE
                     006F   211 ;
                     006F   212 ;--
                     006F   213
                     006F   214
                     006F   215 ;crf$dealblk::
                     006F   216 ;lnk$dealblk::
                     006F   217 lbr$free_vm::
                1C   BB 006F   218          pushr   #^m<r2, r3, r4>
     53    50   07   C1 0071   219          addl3   #<mem$k_roundup-1>,r0,r3        ; Round up to the nearest
           53   07   CA 0075   220 10$:     bicl2   #<mem$k_roundup-1>,r3           ; Multiple of four bytes
                0A   14 0078   221          bgtr    30$
                     007A   222 ;          bleq    20$                             ; Check for size <= 0
                     007A   223 ;          cmpl    r3,mem$l_maxblk                 ; Check size > maximum
                     007A   224 ;          bleq    30$                             ; Branch on legal size
50   0015826C 8F   D0 007A   225 20$:     movl    #mem$_illblksize,r0             ; Report illegal block size
              0063   31 0081   226          brw     deal_blk_exit                   ; Return
     50   0010'CF   9E 0084   227 30$:     movab   mem$l_dynmem,r0                 ; Get listhead of dynamic memory
                61   7C 0089   228          clrq    (r1)                            ; Clear block info long words
     54   53   51   C1 008B   229          addl3   r1,r3,r4                        ; Get address following new block
                     008F   230 ;
                     008F   231 ; Search down list for insertion point.
                     008F   232 ;
           52   50   D0 008F   233 40$:     movl    r0,r2                           ; R2 contains prev block
           50   60   D0 0092   234          movl    (r0),r0                         ; R0 contains next block addr
                25   13 0095   235          beqlu   60$                             ; Branch on end of list
           50   51   D1 0097   236          cmpl    r1,r0                           ; Is new addr > next addr
                F3   1A 009A   237          bgtru   40$                             ; Yes, keep looking
                     009C   238 ;
                     009C   239 ; Found insertion point.
```

```
                                009C   240 ;       r2 = addr of previous block
                                009C   241 ;       r1 = addr of new block
                                009C   242 ;       r0 = addr of next block
                                009C   243 ;
                     62   51    D0  009C   244         movl    r1,blk$l_addr(r2)               ; Point prev to new
                     50   54    D1  009F   245         cmpl    r4,r0                           ; Is new adjacent to next?
                          0B    13  00A2   246         beqlu   50$                             ; Yes, branch to compact
                          35    1A  00A4   247         bgtru   80$                             ; Error, within block
                     61   50    D0  00A6   248         movl    r0,blk$l_addr(r1)               ; No, point new to next
                04   A1   53    D0  00A9   249         movl    r3,blk$l_size(r1)               ; Set size of new block
                          14    11  00AD   250         brb     70$                             ; Go check adjacent to prev
                                00AF   251 ;
                                00AF   252 ; Compact with next block.
                                00AF   253 ;
           04   A1   04   A0    53    C1  00AF   254 50$:    addl3   r3,blk$l_size(r0),blk$l_size(r1) ; Set size = new+next
                     61   60    D0  00B5   255         movl    blk$l_addr(r0),blk$l_addr(r1)   ; Set next pointer
                          60    7C  00B8   256         clrq    (r0)                            ; Clear old next block
                          07    11  00BA   257         brb     70$                             ; Go check for compaction
                                00BC   258 ;
                                00BC   259 ; Set up new block on end of list.
                                00BC   260 ;
                     62   51    D0  00BC   261 60$:    movl    r1,blk$l_addr(r2)               ; Point prev to new
                04   A1   53    D0  00BF   262         movl    r3,blk$l_size(r1)               ; Set new block size
                                00C3   263 ;
                                00C3   264 ; Check for compact with previous block.
                                00C3   265 ;
                54   04   A2    52    C1  00C3   266 70$:    addl3   r2,blk$l_size(r2),r4            ; Get end of prev block
                     51   54    D1  00C8   267         cmpl    r4,r1                           ; Is new adjacent to prev?
                          0E    1A  00CB   268         bgtru   80$                             ; Error, block within prev
                          15    12  00CD   269         bnequ   90$                             ; No, not adjacent
                                00CF   270 ;
                                00CF   271 ; Compact with previous block.
                                00CF   272 ;
              04   A2    04   A1    C0  00CF   273         addl2   blk$l_size(r1),blk$l_size(r2)   ; Prev size = new+prev
                     62   61    D0  00D4   274         movl    blk$l_addr(r1),blk$l_addr(r2)   ; Set up next pntr
                          61    7C  00D7   275         clrq    (r1)                            ; Clear new block pntr & size
                          09    11  00D9   276         brb     90$                             ; Branch to exit
                                00DB   277 ;
                                00DB   278 ; Error, block within block.
                                00DB   279 ;
           50   00158264   8F    D0  00DB   280 80$:    movl    #mem$_blkwthinbl,r0             ; Report block within block
                          03    11  00E2   281         brb     deal_blk_exit                   ; Return
                     50   01    D0  00E4   282 90$:    movl    #mem$_success,r0                ; Set success status code
                                00E7   283 deal_blk_exit:
                          1C    BA  00E7   284         popr    #^m<r2, r3, r4>
                          05    00  00E9   285         rsb
                                00EA   286
                                00EA   287         .end
```

```
$$T1            = 00000000
AL_BLK_EXIT       0000006C R      03
BLK$L_ADDR      = 00000000
BLK$L_SIZE      = 00000004
DEAL_BLK_EXIT     000000E7 R      03
LBR$FREE_VM       0000006F RG     03
LBR$GET_VM        00000000 RG     03
LIB$_BADBLOADR  = 00158264
LIB$_BADBLOSIZ  = 0015826C
MEM$K_MAXBLK    = 00010000
MEM$K_MEMEXP    = 00000080
MEM$K_ROUNDUP   = 00000008
MEM$L_DYNMEM      00000010 RG     02
MEM$L_MAXBLK      00000000 RG     02
MEM$L_MEMEXP      00000004 RG     02
MEM$_BLKWTHINBL = 00158264
MEM$_ILLBLKSIZE = 0015826C
MEM$_SUCCESS    = 00000001
NEW$C_BLKS        00000008 RG     02
SYS$EXPREG        ******** GX     03
```

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
OWN	00000018 (24.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
$CODE$	000000EA (234.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.08	00:00:00.68
Command processing	100	00:00:00.45	00:00:01.98
Pass 1	139	00:00:01.95	00:00:05.79
Symbol table sort	0	00:00:00.10	00:00:00.15
Pass 2	64	00:00:00.68	00:00:01.38
Symbol table output	4	00:00:00.02	00:00:00.02
Psect synopsis output	1	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	342	00:00:03.33	00:00:10.05

The working set limit was 900 pages.
9952 bytes (20 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 105 non-local and 16 local symbols.
287 source lines were read in Pass 1, producing 15 object records in Pass 2.
11 pages of virtual memory were used to define 10 macros.

```
                        +---------------------------+
                        ! Macro library statistics !
                        +---------------------------+

Macro library name                      Macros defined
------------------                      --------------
_$255$DUA28:[SYSLIB]STARLET.MLB;2               7

263 GETS were required to define 7 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:GETMEM/OBJ=OBJ$:GETMEM MSRC$:GETMEM/UPDATE=(ENH$:GETMEM)
```

GETHELP
LIS

INDEX
LIS

GETPUT
LIS

GETMEM
LIS