```
EXIT - Exit from program

Exits from a program and returns to the AEMS Loader. This is the
recommended way to end an AEMS program rather than using BLWP @0
or some other method.

FREE - Free AEMS pages

Frees a previously allocated block of pages for reuse.
Parameters:
     pageno  -  number of the first page of the block
     npages  -  number of pages to free

MOVE - Move data in RAM

Moves data from one page to another. The pages need not be
mapped into the 4A's address space.
Parameters:
     Fpageno -  from extended page number
     Foffset -  offset in the from page
     Tpageno -  to extended page number
     Toffset -  offset in the to page
     length  -  length in bytes of data

VREAD - Read from VDP

Reads data from the VDP into a page of extended memory.  The
page need not be mapped into the 4A's address space.
Parameters:

     Vaddr  -  VDP address
     pageno -  extended page number
     offset -  offset within the page
     length -  length in bytes of data

VSET - Setup VDP

Loads VDP registers and tables for "standard" modes.
Parameters:
     Mode   -  0 - graphics mode
               1 - text mode
               2 - XB graphics mode
     Chars  -  0 - standard character set
               1 - true lower case letters

VWRITE - Write to VDP

Writes data to the VDP from a page of extended memory.  The page
need not be mapped into the 4A's address space.
Parameters:
     Vaddr  -  VDP address
     pageno -  extended page number
     offset -  offset within the page
     length -  length in bytes of data
```

```
.....................
        >05 Load into GRAM 4 at >8000.
        >06 Load into GRAM 5 at >A000.
        >07 Load into GRAM 6 at >C000.
        >08 Load into GRAM 7 at >E000.
        >09 Load into ROM BANK 1 at >6000.
        >0A Load into ROM BANK 2 at >6000.
        >A0 AEMS non-overlay program or overlay root.
        >A1 AEMS overlay segment.
```

Standard program files consist of the exact contents of memory written to a disk. A memory image program may be split into two or more segments depending upon the length of the program. Each of the segments is written to a separate disk file. Option 3 of TI WRITER will load a segment with a maximum size of >2000 while Option 5 of Editor/Assembler will load a segment with maximum size of >2400 bytes. LINKER produces segments with a maximum length of >1FFA. The name of the first segment is specified and the names for all following segments is derived by adding 1 to the last byte of the name of the previous segment. Each memory image file has a 3 word (or 6 bytes) header that indicates to a loader where in memory to load the program. The header is:

```
WORD 1 Flag word as described above.
WORD 2 Length of code in this segment in bytes.  Note that the
       length of the segment on disk is 6 bytes more to include
       the 6 bytes of header information.
WORD 3 Address at which this segment is to be loaded.
```

NOTE that the length of the memory image program may not be the same as the length of the tagged object program. This can be caused by a work area that has no code or data loaded into it occurring at the end of the program. LINKER will not waste disk space writing out an area that you have not filled with code or data. In fact, if your program contains areas inside the program that are unused and which exceed 518 bytes in size LINKER will break the program at that point and create two separate segments. The 518 bytes is the overhead for creating another file -- 1 sector for the disk catalog, 1 sector minimum for a file and 6 bytes of header information per file.

The AEMS Program File format is a bit more complicated than the standard in order to accomadate the much larger memory size and features of the AEMS system.

The header for an AEMS program or root segment of an overlay program is as follows.

```
WORD 1  Flag >FFA0 or >00A0.
WORD 2  Length of code in this segment in bytes.
WORD 3  Address at which this segment is to be loaded.
WORD 4  Total number of pages of memory required for the overlay
        program. This number does not include the two pages at
```

>2000 and >3000. If this number is zero then the program is not an overlay program.

WORD 5 Number of bytes in the "page relocation table" which begins at WORD 6. This may be zero.

WORD 6 If there are page relocation entries (WORD 5 is non-zero) then each word in the table is the address of a word containing a "relative page number" which is to be relocated.

Note that there may be more than one root segment in a program and that the root segment may be loaded in the high memory area >A000 to >FFFF. If there is more than one root segment WORD 4 (number of pages required) will be identical in all segments. It is redundant in all but the first segment.

The header for an AEMS overlay segment is as follows.

WORD 1 Flag >FFA1 or >00A1.

WORD 2 Length of code in this segment in bytes.

WORD 3 Address of this segment when it is mapped in for execution. The last 12 bits of this address is the offset within the relative page at which this segment is loaded.

WORD 4 Relative page number of this segment.

WORD 5 Number of bytes in the "page relocation table" which begins at WORD 6. This may be zero.

WORD 6 If there are page relocation entries (WORD 5 is non-zero) then each word in the table is the address of a word containing a "relative page number" which is to be relocated.

Note that an overlay segment may be more than one page in length.

Overlay Code

The "Overlay Manager" code shown below is added to the root segment of a program by the LINKER.

```
      * Map in N sequential pages and
      * simulate BLWP call to subroutine.
      *
      OVMGR  SBO   0           enable mapper regs
             MOV   *R11+,R10   Get N, # pages
             MOV   *R11+,R9    Get 1st mapper reg
             MOV   *R11+,R7    Get 1st page #
      OVMGR2 MOV   R7,*R9+     Set mapper reg
             INC   R7          Incr page #
             DEC   R10         Loop for N pages
             JGT   OVMGR2
             SBZ   0           Disable mapper regs
             MOV   *R11,*R11   Get real BLWP vector
             MOV   *R11+,R7    Get WSP
             MOV   *R11,R9     Get branch addr
```

```
        MOV   R13,@26(R7) Simulate BLWP
        MOV   R14,@28(R7)
        MOV   R15,@30(R7)
OVMGRW  EQU   $-12        OVMGR workspace
* CALL user subroutine.
        LWPI  0           R6,R7 Load user WSP
        B     @0          R8,R9 Go to user sub
        BSS   2           R10
        BSS   2           R11
        DATA  >1E00       R12 Mapper CRU addr
        BSS   6           R13-R15
```

Shown below is the "stub" inserted by the LINKER for each subroutine call that causes an overlay.

```
OSUB    DATA  OVMGRW      Overlay Manager
        DATA  $+2.
        BL    @OVMGR      Call manager
        DATA  N           # pages in overlay
        DATA  >400X       1st mapper reg addr
        DATA  n           1st page number
        DATA  sub         real BLWP vector
```

AEMS Mapper

AEMS uses the Texas Instruments SN74LS612 memory mapper chip along with additional logic to map the high memory addresses from a 16 bit address to a 23 bit address. A 23 bit address accommodates a 4 Megabyte memory.

This mapping is done by splitting the TI 99/4A 16 bit address into two parts: a 4 bit page number and a 12 bit page offset. The 12 bit page offset gives a 4K page. The 4 bit page number is used to select a "mapper register" containing an 11 bit page number. The 11 bit expanded page number is combined with the original 12 bit page offset to give a 23 bit expanded memory address.

The mapper can be inactive or active. When the mapper is inactive the TI99/4A will operate as though it had an ordinary 32K memory card. At power on, the mapper is inactive.

The mapper is activated by setting a CRU bit (>1E02) to one. When active only addresses >A000 to >FFFF are mapped. Mapping can be turned off by setting the CRU bit to zero.

The mapper has 6 active "registers" that specify which pages are mapped into the TI99/4A's address space. In order to access these registers (for write or read) the access must be enabled by setting CRU bit >1E00 to one. Access is disabled by setting the CRU bit to zero. The mapper registers are accessed by normal 9900 instructions at the following addresses:

Register Number	Access at Address	Maps Page in at
A	>40A0	>A000
B	>40B0	>B000
C	>40C0	>C000
D	>40D0	>D000
E	>40E0	>E000
F	>40F0	>F000

The access addresses will respond to instructions just like a normal word of RAM.

Typical code for operation of the mapper is shown below.

```
* Load Mapper Registers
       LI    R12,>1E00     CRU base address
       SBO   0             Enable register access
       LI    R0,20         Page # 20
       MOV   R0,@>40A0     Map page in at >A000
       LI    R0,50         Page # 50
       MOV   R0,@>40B0     Map page in at >B000
       SBZ   0             Disable register access
       SBO   1             Turn mapper on
       MOV   @>A000,R0     Get 1st word page 20
       MOV   @>B002,R1     Get 2nd word page 50
       SBZ   1             Turn mapper off
```

Only in special cases should the programmer have to manipulate the mapper or its registers directly. Assembler macros and library routines are provided for most routine tasks.

AEMS LIBRARY MANAGER

The AEMS Library Manager is a program to assist you to build and maintain libraries of routines for use by the LINKER.