

**NAME**

a.out – assembler and link editor output

**DESCRIPTION**

*A.out* is the output file of the assembler *as* and the link editor *ld*. Both programs make *a.out* executable if there were no errors and no unresolved external references.

This file has four sections: a header, the program and data text, a symbol table, and relocation bits (in that order). The last two may be empty if the program was loaded with the “-s” option of *ld* or if the symbols and relocation have been removed by *strip*.

The header always contains 8 words:

- 1 A magic number (407, 410, or 411(8))
- 2 The size of the program text segment
- 3 The size of the initialized portion of the data segment
- 4 The size of the uninitialized (bss) portion of the data segment
- 5 The size of the symbol table
- 6 The entry location (always 0 at present)
- 7 Unused
- 8 A flag indicating relocation bits have been suppressed

The sizes of each segment are in bytes but are even. The size of the header is not included in any of the other sizes.

When a file produced by the assembler or loader is loaded into core for execution, three logical segments are set up: the text segment, the data segment (with uninitialized data, which starts off as all 0, following initialized), and a stack. The text segment begins at 0 in the core image; the header is not loaded. If the magic number (word 0) is 407, it indicates that the text segment is not to be write-protected and shared, so the data segment is immediately contiguous with the text segment. If the magic number is 410, the data segment begins at the first 0 mod 8K byte boundary following the text segment, and the text segment is not writable by the program; if other processes are executing the same file, they will share the text segment. If the magic number is 411, the text segment is again pure, write-protected, and shared, and moreover instruction and data space are separated; the text and data segment both begin at location 0. See the 11/45 handbook for restrictions which apply to this situation.

The stack will occupy the highest possible locations in the core image: from 177776(8) and growing downwards. The stack is automatically extended as required. The data segment is only extended as requested by the *break* system call.

The start of the text segment in the file is 20(8); the start of the data segment is  $20+S_t$  (the size of the text) the start of the relocation information is  $20+S_t+S_d$ ; the start of the symbol table is  $20+2(S_t+S_d)$  if the relocation information is present,  $20+S_t+S_d$  if not.

The symbol table consists of 6-word entries. The first four words contain the ASCII name of the symbol, null-padded. The next word is a flag indicating the type of symbol. The following values are possible:

- 00 undefined symbol
- 01 absolute symbol
- 02 text segment symbol
- 03 data segment symbol
- 37 file name symbol (produced by *ld*)
- 04 bss segment symbol
- 40 undefined external (.globl) symbol
- 41 absolute external symbol
- 42 text segment external symbol
- 43 data segment external symbol

#### 44 bss segment external symbol

Values other than those given above may occur if the user has defined some of his own instructions.

The last word of a symbol table entry contains the value of the symbol.

If the symbol's type is undefined external, and the value field is non-zero, the symbol is interpreted by the loader *ld* as the name of a common region whose size is indicated by the value of the symbol.

The value of a word in the text or data portions which is not a reference to an undefined external symbol is exactly that value which will appear in core when the file is executed. If a word in the text or data portion involves a reference to an undefined external symbol, as indicated by the relocation bits for that word, then the value of the word as stored in the file is an offset from the associated external symbol. When the file is processed by the link editor and the external symbol becomes defined, the value of the symbol will be added into the word in the file.

If relocation information is present, it amounts to one word per word of program text or initialized data. There is no relocation information if the "suppress relocation" flag in the header is on.

Bits 3-1 of a relocation word indicate the segment referred to by the text or data word associated with the relocation word:

- 00 indicates the reference is absolute
- 02 indicates the reference is to the text segment
- 04 indicates the reference is to initialized data
- 06 indicates the reference is to bss (uninitialized data)
- 10 indicates the reference is to an undefined external symbol.

Bit 0 of the relocation word indicates if *on* that the reference is relative to the pc (e.g. "clr x"); if *off*, that the reference is to the actual symbol (e.g., "clr \*\$x").

The remainder of the relocation word (bits 15-4) contains a symbol number in the case of external references, and is unused otherwise. The first symbol is numbered 0, the second 1, etc.

#### SEE ALSO

as(I), ld(I), strip(I), nm(I)