Assembler directives and options

as options

It's particularly evident that *as* seldom sees the light of day when you look at the options, which differ greatly from one system to the next. GNU *as* doesn't even maintain compatibility between versions 1 and 2, as you can see in the following table:

`-a` *(GNU 2.x)*

List high-level language, assembly output, and symbols. This is the generic form of the `-a` option; the following variants modify this in some manner. Combinations are possible: for example, `-alh` lists the high-level input and assembly output, but not the symbol table. In order to get the high-level language input, you also need to specify the `-g option.`

`-ad` *(GNU 2.x)*

List high-level language, assembly output, and symbols, but omit debugging pseudo-ops from listing.

`-ah` *(GNU 2.x)*

List high-level language source.

`-al` *(GNU 2.x)*

List assembly output.

`-an` *(GNU 2.x)*

Disable forms processing of the listing. This only works in combination with other `-a` options.

`-as` *(GNU 2.x)*

List symbols.

`-D` *(GNU 1.x)*

Turn on assembler debugging (if available).

`-D` *(GNU 2.x)*

No effect—just for compatibility.

`-dl` *(SVR3)*

Don’t put line number information in object file.

`-f` *(GNU 2.x)*

skip preprocessing (for compiler output)

`-g` *(GNU 1.x)*

Generate debugging symbols for source language debugging of assembly programs.

`-I` *path* *(GNU 2.x)*

Add *path* to the search list for `.include` directives

`-K` *(GNU 2.x)*

Issue warnings when difference tables altered for long displacements.

`-k` *(GNU 1.x)*

Warn about problems with calculating symbol differences.

`-L` *(GNU)*

Keep local symbols starting with `L` in the symbol table output to object file.

`-m` *(System V)*

preprocess with *m4*

`-n` *(System V)*

Turn off long/short address optimization.

`-o` *(GNU 2.x, System V)*

Specify output file name.

`-Qy` *(System V)*

Put assembler version number in object file.

`-R` *(GNU 1.x)*

Merge the data segment into the text segment, making it read-only.

`-R` *(System V)*

Remove the input file after assembly.

`-W` *(GNU 1.x)*

Suppress warnings.

`-f` *(GNU 1.x)*

Suppress the preprocessor pass which removes comments and redundant white space from the input. This can also be done with the #NO_APP directive.

`-T` *(System V)*

Accept (and ignore) obsolete directives without complaining.

`-V` *(System V)*

Print the current version number.

`-v` *(GNU)*

Print the current version number.

`-W` *(GNU 2.x)*

Suppress warning messages

`-Y` *(System V)*

Specify directory for *m4* processor and predefined macros (`Y,`*dir*).

`-Yd` *(System V)*

Specify directory for predefined macros (`Yd,`*dir*).

`-Ym` *(System V)*

Specify directory for *m4* processor (`Ym,`*dir*).

as directives

Assembler directives are mainly provided for the convenience of the compiler, and are seldom documented. Here is a list of the directives provided by GNU *as*, one of the few which is documented. Many of these directives are provided only on certain platforms—read *Using as*, by Dean Elsner and Jay Fenlason, for specific information.

`.abort`

Abort the assembly. This is obsolescent. It was intended to be used by a compiler piping its output into the assembler when it discovered a fatal error.

`.ABORT`

A synonym for `.abort.`

`.align` *boundary* `[,` *content*]

Increment the assembler location counter, (the pointer to the location where the next byte will be emitted), to a boundary which has zeros in the last *boundary* binary positions. If *content* is specified, any bytes skipped will be filled with this value.

`.app-file` *string*

Specify the start of a new logical file *string*. This is obsolescent.

`.ascii` *string* `...`

Emit each *string* into consecutive addresses. Do not append a trailing `\0` character.

`.asciz` *string*

Emit each *string* into consecutive addresses. Append a trailing `\0` character.

`.byte` *expressions*

Emit zero or more *expression*s into the next output byte.

`.comm` *symbol* , *length*

Declare *symbol* a named common area in the bss section. *length* is the minimum length—the actual length will be determined by the linker as the maximum of the *length* fields of all object

files which define the symbol.

`.data` *subsection*

Switch to data section *subsection* (default zero). All assembled data will go to this section.

`.def` *name*

Begin defining COFF debugging information for a symbol *name*. The definition is completed by a `.endef` directive.

`.desc` *symbol*, *abs-expression*

Set the symbol descriptor to the low 16 bits of *abs-expression*. This is ignored if the assembler is outputting in COFF format.

`.double` *flonums*

Emit `double` floating point number *flonums*.

`.eject`

Force a page break in the assembly listing at this point.

`.else`

else in conditional assembly—see the `.if` directive.

`.endef`

End a symbol definition begun with `.def`.

`.endif`

End a conditional assembly block. See the `.if` directive.

`.equ` *symbol*, *expression*

Set the value of *symbol* to *expression*. This is the same thing as `.set`.

`.extern`

In some assemblers, define a symbol external to the program. This is ignored by GNU *as*, which treats all undefined symbols as external.

`.file` *string*

Specify the start of a new file. This directive is obsolescent, and may not be available.

`.fill` *repeat* , *size* , *value*

Create *repeat* repeated data blocks consisting of the low-order *size* bytes of *value*.

`.float` *flonums*

Emit floating point numbers *flonums*.

`.global` *symbol*

Defi ne *symbol* as an external symbol.

`.globl`*symbol*

A synonym for `.global`.

`.hword` *expressions*

Emit the values of each *expression*, truncated to 16 bits if necessary.

`.ident`

This directive is used by some assemblers to place tags in object fi les. GNU *as* ignores it.

`.if` *expression*

If *expression* evaluates to non zero, assemble the following code down to the corresponding `.else` or `.endif` directive. If the next directive is `.else`, do not assemble the code between the `.else` and the `.endif`. If *expression* evaluates to 0, do not assemble the code down to the corresponding `.else` or `.endif` directive.

`.ifdef` *symbol*

Like `.if`, but the condition is fulfi lled if *symbol* is defi ned.

`.ifndef` *symbol*

Like `.if`, but the condition is fulfi lled if *symbol* is not defi ned.

`.ifnotdef` *symbol*

Like `.if`, but the condition is fulfi lled if *symbol* is not defi ned.

`.include "`*file*"

Process the source fi le *file* before continuing this fi le.

`.int` *expression*s

Emit 32 bit values of each *expression*.

`.lcomm` *symbol* , *length*

Reserve *length* bytes of local common in bss, and give it the name *symbol*.

`.ln` *line-number*

Change the logical line number of the next line to *line-number*. This corresponds to the C pre-processor `line` directive.

`.ln` *line-number*

A synonym for `.line`.

`.list`

Increment the listing counter (initially 0). If the listing counter is > 0, the following lines will be listed in the assembly listing, otherwise they will not. `.nolist` decrements the counter.

`.long` *expressions*

A synonym for `.int`.

`.nolist`

Decrement the listing counter—see `.list`.

`.octa` *bignum*s

Evaluate each *bignum* as a 16 byte integer and emit its value.

`.org` *new-lc*, *fill*

Set the location counter of the current section to *new-lc*. *new-lc* must be either absolute or an expression in the current subsection: you can't use `.org` to cross sections. `.org` may not decrement the location counter. The intervening bytes are filled with the value *fill* (default 0).

`.psize` *lines*, *columns*

Set the page size for assembly listings to *lines* lines (default 60) and *columns* columns (default 200). If *lines* is set to 0, no automatic pagination will occur.

`.quad` *bignums*

Evaluate each *bignum* as an 8 byte integer and emit its value.

`.sbttl` *subheading*

Set the subtitle of assembly listings to *subheading*.

`.section` *name*, *subsection*

Switch to section called `name` (default `.text`), *subsection* (default zero). All emitted data goes to this section.

`.set` *symbol*, *expression*

Define the value of *symbol* to be *expression*. This may be used more than once to change the value of *symbol* after it is defined. The value of an external symbol will be the value of the last `.set` directive.

`.short` *expression*s

Emit the values of each *expression*, truncated to 16 bits if necessary.

`.single` *flonums*

Emit floating point numbers *flonums*. This is the same as `.float`.

`.space` *size*, *fill*

Emit *size* bytes of value *fill*. *fill* defaults to 0.

`.space`

Usually a synonym for *.block*, but on some hardware platforms GNU *as* uses it differently.

`.stabd`

Emit debug information. See page for more information.

`.stabn`

Emit debug information. See page for more information.

`.stabs`

Emit debug information. See page for more information.

`.text` *subsection*

Switch to text section *subsection* (default zero). All assembled data will go to this section.

`.title` *heading*

Set the title of the assembly listing to *heading*.

`.word` *expressions*

Emit 32 bit values of each *expression*.

Debug information

Debug information is very dependent on the kind of object file format in use: In *a.out* format, it is defined by the directives `.stabd`, `.stabn` and `.stabs`. They can take up to five parameters:

- *desc* is the symbol descriptor, and is 16 bits wide.
- *other* is the symbol's "other" attribute. This is normally not used.
- *string* is the name of the symbol.
- *type* is the symbol type, and is 8 bits wide.
- *value* is the value of the symbol, and must be absolute.

These symbols are used as follows:

.*stabd type*, *other*, *desc*

Define a debugging entry without a name. The value of the symbol is set to the current value of the location counter. This is commonly used for line number information, which is type 68 for line number references in the text segment. For example `.stabd 68, 0, 27` specifies that the current location is the beginning of line 27.

.*stabn type*, *other*, *desc*, *value*

Define a debugging entry without a name. The value of the symbol is set to *value*.

.*stabs string*, *type*, *other*, *desc*, *value*

Define a debugging entry with the name *string*. The value of the symbol is set to *value*.

For further information about *stabs* formats and types, see the header file *stab.h* and the man page *stab(5)*.

In *COFF* format, it is defined by the directives `.dim`, `.scl`, `.size`, `.tag`, `.type` and `.val`. They are enclosed in a `.def`/`.endef` pair. For example, to define a symbol `foo`, you would write

```
.def    foo
.value bar
.size 4
.endef
```

`.dim`

Set dimension information.

`.scl` *class*

Set the storage class value of the symbol to *class*.

`.size` *size*

Set the size of the symbol to *size*.

`.tag` *structname*

Specify the struct definition of the current symbol.

`.type` *int*

Set the type of the symbol to *type*.

`.val` *addr*

Set the value of the symbol to *addr*.

In *ELF* format, debug information is output to a special section called `.debug`, so no specific directives are needed.