GAS Tutorial - 7. Directives (3)

Young W. Lim

2016-07-13 Thr

Outline

Based on

"Using as", Dean Elsner, Jay Fenlason & friends

7.30 .comm symbol , length (1)

- declares a common symbol named symbol.
- linker merges a common sybol across object files
- if undefined, length bytes of unintialized memory will be allocated
- length byte must be absolute
- if sized conflicted, the largest length byte will be allocated

7.30 .comm symbol , length (2)

When using ELF or (as a GNU extension) PE

- an optional third argument : alignment
- 16 specified for ELF as a byte boundary
  - the least significant 4 bits of the address should be zero
- 5 for PE as a power of two
  - aligned to a 32-byte boundary
- must be an absolute expression
- when allocating uninitialized memory, alignment is used

when no alignment is specified,

- ELF: the largest power of two <= the size of the symbol
- up to a maximum of 16
- PE: the default section alignment of 4

7.30 .comm symbol , length (3)

- This is not the same as the executable image file alignment controlled by ld's '–section-alignment' option;
- image file sections in PE are aligned to multiples of 4096, which is far too large an alignment for ordinary variables.
- It is rather the default alignment for (non-debug) sections within object ('*.o') files, which are less strictly aligned.

7.31 .data subsection

- tells as to assemble the following statements onto the end of the data subsection numbered subsection
- the subsecton number is an absolute expression
- If subsection is omitted, it defaults to zero

7.33 .desc symbol , abs-expression

- sets the descriptor to the low 16 bits of an absolute expression
- is not available when as is configured for COFF output
- is only for a.out or b.out object format

7.49 .extern

- is accepted in the source program - for compatibility with other assemblers but it is ignored
- all undefined symbols are treated as external

7.55 .global symbol , .globl symbol

- makes the symbol visible to ld
- if it is defined, its value will be made know to other linked object files
- otherwise, it takes its attributes from other linked object files
- '.globl' and '.global' are accepted

7.56 .gnu_attribute tag ,value

- record a gnu object attribute for this file

7.59 .ident

- used by some assemblers to place tags in object files
- the behavior of this directive varies depending on the target

When using the a.out object file format

- simply accepts the directive for source-file compatibility
- does not emit anything for it.

When using COFF

- comments are emitted to the .comment or .rdata section
- depending on the target.

When using ELF

- comments are emitted to the .comment section

7.67 .lcomm symbol , length

- reserve length (an absolute expression) bytes for a local common denoted by symbol
- the section and value of symbol are those of the new local common
- the addresses are allocated in the bss section
- at run-time the bytes start off zeroed
- symbol is not declared global
- is normally not visible to ld.
- some targets permit a third argument (alignment)
- to be used with .lcomm.
- the desired alignment of the symbol in the bss section

7.68 .lflags

- as accepts this directive, for compatibility with other assemblers
- but ignores it.

7.73 .loc fileno lineno [column] [options] (1)

When emitting DWARF2 line number information,

- add a row to the .debug_line line number matrix corresponding to the immediately following assembly instruction.
- the fileno, lineno, and optional column arguments will be applied to the .debug_line state machine before the row is added.

7.73 .loc fileno lineno [column] [options] (2)

The options are a sequence of the following tokens in any order:

basic_block

- This option will set the basic_block register in the .debug_line state machine to true.

prologue_end

- This option will set the prologue_end register in the .debug_line state machine to true.

epilogue_begin

- This option will set the epilogue_begin register in the .debug_linestate machine to true.

7.73 .loc fileno lineno [column] [options] (3)

is_stmt value

- This option will set the is_stmt register in the .debug_line state machine to value, which must be either 0 or 1.

isa value

- This directive will set the isa register in the .debug_line state machine to value, which must be an unsigned integer.

discriminator value

- This directive will set the discriminator register in the .debug_line state machine to value, which must be an unsigned integer.

7.74 .loc_mark_labels enable

When emitting DWARF2 line number information

- makes the assembler emit an entry to the .debug_line line number matrix with the basic_block register in the state machine set whenever a code label is seen
- The enable argument should be either 1 or 0, to enable or disable this function respectively.

7.75 .local names

- available for ELF targets, marks each symbol in the comma separated list of names as a local symbol
- not be externally visible
- if the symbols do not already exist, they will be created.

For targets where the .lcomm directive does not accept an alignment argument (most ELF targets)

- be used in combination with .comm to define aligned local common data.

7.88 .psize lines , columns

- use to declare the number of lines and optionally, the number of columns
- when generating listings in each page
- default line value of 60 lines
- default column value of 200 columns
- generates formfeeds
  - whenever the specified number of lines is exceeded
  - whenever explicit .eject is used
- If you specify lines as 0, no formfeeds are generated

7.92 .reloc offset , reloc_name [, expression] (1)

- generate a relocation at offset of type reloc name with value expression.
- If offset is a number, the relocation is generated in the current section.
- If offset is an expression that resolves to a symbol plus offset, the relocation is generated in the given symbol's section.
- expression, if present, must resolve to a symbol plus addend or to an absolute value, but note that not all targets support an addend.

7.92 .reloc offset , reloc_name [, expression] (2)

- e.g. ELF REL targets such as i386 store an addend in the section contents rather than in the relocation.
- This low level interface does not support addends stored in the section.

7.96 .section name

- assemble the following code into a section named name
- used when arbitrarily named sections are supported
- on a.out targets, it is not accepted, even with a standard a.out section name.

7.96 .section name - COFF (1)

- .section name [, "flags "]
- .section name [, subsection]
- the optional flag argument is quoted
- If not quoted, it is taken as a subsection number

7.96 .section name - COFF (2)

- b : bss section (uninitialized data)
- n : section is not loaded
- w : writable section
- d : data section
- r : read-only section
- x : executable section
- s : shared section (meaningful for PE targets)
- a : ignored. (For compatibility with the ELF version)
- y : section is not readable (meaningful for PE targets)

7.96 .section name - COFF (3)

- the default flags depend upon the section name
- If the section name is not recognized, the default will be for the section to be loaded and writable.
- the n and w flags remove attributes from the section

7.96 .section name - ELF (1)

- .section : one of the ELF section stack manipulation directives
- related directives
  - .subsection
  - .pushsection
  - .popsection
  - .previous
- Usage: .section name [, "flags "[, @type [,flag_specific_arguments]]]
  - The optional flags argument :
  - a : section is allocatable
  - w : section is writable
  - x : section is executable
  - M : section is mergeable
  - S : section contains zero terminated strings
  - G : section is a member of a section group
  - T : section is used for thread-local-storage

7.96 .section name - ELF (2)

- Usage: .section name [, "flags "[, @type [,flag_specific_arguments]]]
  - The optional type argument :
  - @progbits : section contains data
  - @nobits : section does not contain data (only occupies space)
  - @note : contains data which is not used as the program
  - @init_array : contains an array of pointers to init functions
  - @fini_array : contains an array of pointers to finish functions
  - @preinit_array : contains an array of pointers to pre-init functions
- Many targets only support the first three section types.
- on ARM target, the @ character is the start of a comment. Use the % character.

7.96 .section name - ELF (3)

- Usage: .section name [, "flags "[, @type [,flag_specific_arguments]]]
  - If M (mergeable) flag is used, use like this
    - .section name , "flags "M, @type , entsize
  - Sections with the M flag but not S (String) flag
    - must contain fixed size constants (each entsize octets long)
  - Sections with both M and S flags
    - must contain zero terminated strings (each character entsize octects long)
  - The linker may remove duplicates within sections with the same name, same entity size and same flags. entsize must be an absolute expression.
  - For sections with both M and S, a string which is a suffix of a larger string is considered a duplicate.
    - Thus "def" will be merged with "abcdef"
    - A reference to the first "def" will be changed to a reference to "abcdef"+3.

7.96 .section name - ELF (4)

- Usage: .section name [, "flags "[, @type [,flag_specific_arguments]]]
- If flags contains the G (group) symbol, then use like this
  - .section name , "flags "G, @type , GroupName [, linkage]
  - the GroupName specifies the name of the section group
  - this particular section belongs to the GroupName
  - the optional linkage field
    - comdat : only one copy of this section should be retained
    - .gnu.linkonce : an alias for comdat
- If both the M and G flags are present, then use like this:
  - .section name , "flags "MG, @type , entsize , GroupName [, linkage]
  - If no flags are specified, the default flags depend upon the section name.
  - If the section name is not recognized, the default will be for the section to have none of the above flags: it will not be allocated in memory, nor writable, nor executable. The section will contain data.

7.96 .section name - ELF (5)

- compatibility with the Solaris assembler:
  - .section "name "[, flags . . .]
  - the section name is quoted.
  - there may be a sequence of comma separated flags:
    - #alloc : section is allocatable
    - #write : section is writable
    - #execinstr : section is executable
    - #tls : section is used for thread local storage
  - This directive replaces the current section and subsection.
  - See the contents of the gas testsuite directory gas/testsuite/gas/elf for some examples of how this directive and the other section stack directives work.

7.100 .size

- This directive is used to set the size associated with a symbol.

COFF Version

- only permitted inside .def/.endef pairs.
- .size expression

ELF Version

- .size name , expression

- sets the size associated with a symbol name
- The size in bytes is computed from expression
- which can make use of label arithmetic
- typically used to set the size of function symbols

7.104 .stabd, .stabn, .stabs (1)

- three directives that begin '.stab' + (d/n/s)
- all emit symbols that can be used by symbolic debuggers
- these symbols are not entered in the as hash table
- these cannot be referenced elsewhere in the source file
- up to five fields are required:
  - string
  - type
  - other
  - desc
  - value

7.104 .stabd, .stabn, .stabs (2)

string

- the symbol's name
- may contain any character except '000'
- more general than ordinary symbol names
- Some debuggers used to code arbitrarily complex structures into symbol names using this field.

type

- an absolute expression
- set to the low 8 bits of this expression
- any bit pattern is permitted
- ld and debuggers choke on silly bit pattern

other

- an absolute expression. T
- set to the low 8 bits of this expression

7.104 .stabd, .stabn, .stabs (3)

desc

- an absolute expression
- set to the low 16 bits of this expression

value

- an absolute expression which becomes the symbol's value.

- If a warning is detected while reading statement, the symbol has probably already been created
- a half-formed symbol in your object file
- this is compatible with earlier assemblers

7.104 .stabd, .stabn, .stabs (4)

.stabd type , other , desc

- the "name" of the symbol generated is not even an empty string but it is a null pointer, for compatibility
- Older assemblers used a null pointer so they didn't waste space in object files with empty strings
- the symbol's value is set to the location counter, relocatably
- When your program is linked, the value of this symbol is the address of the location counter when the .stabd was assembled.

.stabn type , other , desc , value

- The name of the symbol is set to the empty string ""

.stabs string , type , other , desc , value

- all five fields are specified.

7.107 .subsection name

- one of the ELF section stack manipulation directives
  - .section
  - .pushsection
  - .popsection
  - .previous
- replaces the current subsection with name
- the current section is not changed
- pushed onto the section stack
- in place of the then current top of stack subsection.

7.108 .symver (1)

- used to bind symbols to specific version nodes within a source file
- only supported on ELF platforms
- typically used when assembling files to be linked into a shared library
- often to use this in objects to be bound into an application itself so as to override a versioned symbol from a shared library.

7.108 .symver (2)

- usage : for ELF targets
- .symver name , name2@nodename

- if the symbol 'name' is defined within the file being assembled
- .symver creates a symbol alias with name2@nodename
  - symbol alias allow '@'
  - regular alias does not allow '@'
- the symbol is externally referenced by 'name2' in name2@nodename
- 'name' itself is merely a name of convenience
- possible to have definitions for multiple versions of a function within a single source file
- the compiler can unambiguously know which version of a function is being mentioned.

7.108 .symver (3)

- usage : for ELF targets
- .symver name , name2@nodename

- 'nodename' should be the name of a node specified in the version script supplied to the linker when building a shared library
- to override a versioned symbol from a shared library, then 'nodename' should correspond to the nodename of the symbol you are trying to override.
- If the symbol name is not defined within the file being assembled, all references to name will be changed to name2@nodename.
- If no reference to name is made, name2@nodename will be removed from the symbol table.

7.108 .symver (4)

- Another usage of the .symver directive
- .symver name , name2@@nodename

- 'name' must exist and be defined within the file being assembled
- similar to name2@nodename
- difference: name2@@nodename will also be used to resolve references to name2 by the linker.

7.108 .symver (5)

- the third usage of the .symver directive is:
- .symver name , name2@@@nodename

- if 'name' is not defined within the file being assembled, it is treated as name2@nodename
- if 'name' is defined within the file being assembled, the symbol name 'name' will be changed to name2@@nodename.

7.109 .tag structname

- generated by compilers to include auxiliary debugging information in the symbol table
- only permitted inside .def/.endef pairs
- used to link structure definitions in the symbol table with instances of those structures.

7.110 .text subsection

- tells as to assemble the following statements onto the end of the text subsection numbered 'subsection'
- 'subsection' is an absolute expression
- subsection is omitted, subsection number zero is used

7.116 .vtable_entry table , offset

- finds or creates a symbol table and creates a VTABLE_ENTRY relocation for it with an addend of offset.

7.117 .vtable_inherit child , parent

- finds the symbol child
- finds or creates the symbol parent
- creates a VTABLE_INHERIT relocation for the parent whose addend is the value of the child symbol
- As a special case the parent name of 0 is treated as referring to the ABS section.

7.119 .weak names

- sets the weak attribute on the comma separated list of symbol names
- if not already exist, it will be created
- on COFF targets other than PE, weak symbols are a GNU extension.
- on the PE target, weak symbols are supported natively as weak aliases.
- when a weak symbol is created that is not an alias, GAS creates an alternate symbol to hold the default value.

7.120 .weakref alias , target

- creates an alias to the target symbol that enables the symbol to be referenced with weak-symbol semantics, but without actually making it weak.
- If direct references or definitions of the symbol are present, then the symbol will not be weak, but if all references to it are through weak references, the symbol will be remarked as weak in the symbol table.
- The effect is equivalent to moving all references to the alias to a separate assembly source file, renaming the alias to the symbol in it, declaring the symbol as weak there, and running a reloadable link to merge the object files resulting from the assembly of the new source file and the old source file that had the references to the alias removed.
- The alias itself never makes to the symbol table, and is entirely handled within the assembler.