SPIM (and QtSPIM) assembler directives.

Assembler Syntax

Comments in assembler files begin with a sharp sign (#). Everything from the sharp sign to the end of the line is ignored.

Identifiers are a sequence of alphanumeric characters, underbars (_), and dots (.) that do not begin with a number. Instruction opcodes are reserved words that *cannot* be used as identifiers. Labels are declared by putting them at the beginning of a line followed by a colon, for example:

```
      .data
item: .word 1
      .text
      .globl main      # Must be global
main: lw         $t0, item
```

Numbers are base 10 by default. If they are preceded by *0x,* they are interpreted as hexadecimal. Hence, 256 and 0x100 denote the same value.

Strings are enclosed in doublequotes ("). Special characters in strings follow the C convention:

- newline `\n`
- tab `\t`
- quote `\"`

SPIM supports a subset of the MIPS assembler directives:

`.align n`	Align the next datum on a 2^n byte boundary. For example, `.align 2` aligns the next value on a word boundary. `.align 0` turns off automatic alignment of `.half`, `.word`, `.float`, and `.double` directives until the next `.data` or `.kdata` directive.
`.ascii str`	Store the string *str* in memory, but do not null-terminate it.

`.asciiz str`	Store the string *str* in memory and null-terminate it.
`.byte b1,..., bn`	Store the *n* values in successive bytes of memory.
`.data <addr>`	Subsequent items are stored in the data segment. If the optional argument *addr* is present, subsequent items are stored starting at address *addr*.
`.double d1, ..., dn`	Store the *n* floating-point double precision numbers in successive memory locations.
`.extern sym size`	Declare that the datum stored at *sym* is *size* bytes large and is a global label. This directive enables the assembler to store the datum in a portion of the data segment that is efficiently accessed via register `$gp`.
`.float f1,..., fn`	Store the *n* floating-point single precision numbers in successive memory locations.
`.globl sym`	Declare that label *sym* is global and can be referenced from other files.
`.half h1, ..., hn`	Store the *n* 16-bit quantities in successive memory halfwords.
`.kdata <addr>`	Subsequent data items are stored in the kernel data segment. If the optional argument *addr* is present, subsequent items are stored starting at address *addr*.
`.ktext <addr>`	Subsequent items are put in the kernel text segment. In SPIM, these items may only be instructions or words (see the `.word` directive below). If the optional argument *addr* is present, subsequent items are stored starting at address *addr*.
`.set noat` and `.set at`	The first directive prevents SPIM from complaining about subsequent instructions that use register `$at`. The second directive reenables the warning. Since pseudoinstructions expand into code that uses register `$at`, programmers must be very careful about leaving values in this register.
`.space n`	Allocate *n* bytes of space in the current segment (which must be the data segment in SPIM).

`.text <addr>`	Subsequent items are put in the user text segment. In SPIM, these items may only be instructions or words (see the `.word` directive below). If the optional argument *addr* is present, subsequent items are stored starting at address *addr*.
`.word w1,..., wn`	Store the *n* 32-bit quantities in successive memory words.

SPIM does not distinguish various parts of the data segment (`.data`, `.rdata`, and `.sdata`).