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 string*	Store the string in memory, but do not null-terminate it.
.asciiz string*	Store the string in memory and null-terminate it.
.byte b1,..., bn	Store the *n* 8-bit 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*. For example: `.data 0x00008000`
.double d1, ..., dn	Store the n floating-point double-precision numbers in successive memory locations.
.extern Symb size	Declare that the datum stored at *Symb* is of 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 Symb	Declare that label *Symb* is global so it can be referenced from other files.
.half h1,... hn	Store the *n* 16-bit quantities in successive memory half words.
.kdata <add>	Subsequent 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 kernal text segment. In SPIM, these items may only be instructions or words. If the optional argument *addr* is present, subsequent items are stored starting at address *addr* (e.g., `.ktext 0x80000080`).
.space n	Allocate *n* bytes of space in the current segment (which must be the data segment in PCSpim).
.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* (e.g., `.data 0x00400000`).
.word w1,..., wn	Store the *n* 32-bit quantities in successive memory words.
.word w : n	Stores the 32-bit value *w* into *n* successive memory words.

*Strings are enclosed in double quotes (`"`). Special characters in strings follow the C convention: `newline: \n, tab: \t, quote: \"`. Instruction op-codes are reserved words and may not be used as labels. Labels must appear at the beginning of a line followed by a colon. The ASCII code "back space" is not supported by the SPIM simulator. 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.