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*.
`.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.

```
char A[4] = {1, 2, 3, 4};
int result;

result = A[0] + A[1] + A[2] + A[3];
```

```
if (v0 < 0)
    v0 --;

else
    v0 ++;          E:
v1 = v0;            L:
```

```
if (x < 10) {
        ...
}
```

```
slti $t0, $t4, 10           # immediate version of slt
____ $t0, $zero, skip_if_body   # beq(a) or bne(b)?
```