NAME

assembler−format − format of input assembly files

SYNOPSIS

```
; is a single line comment, you can write everything here
.data
; You can use labels here as well as in memory locations
LABEL:
   .byte   1, 2, 3, LABEL      ; 8 bit numbers
   .half   1, 2, 3, LABEL      ; 16 bit numbers
   .word   1, 2, 3, LABEL      ; 32 bit numbers
   .dword  1, 2, 3, LABEL      ; 64 bit numbers
   .eqv    CONST, 20           ; CONST is set to 20

.text
; You can also use labels here in the .text section
; Write instructions and macros here
START:
   nop
   addi    zero, zero, 0
   addi    t1, zero, 2
   mul     t0, sp, t1
LOD:
   j       LOD                 ; You can use labels in macros
```

DESCRIPTION

The assembler(1) command converts assembly files to output in binary or mif format. The syntax is based on MIPS assembly syntax and is modified to lessen programmer burden and increase compatibility to already established tools and conventions.

SYNTAX

In general, every line contains nothing (blanks), a section, a label or a label and operation. Comments can be added to the end of any line or inserted into a line that otherwise would contain nothing (blanks). The assembler is leniant with the placement of these components to eachother. A operation is an instruction, macro or directive.

A single line:

```
LABEL: <OPERATION> ; COMMENT
```

Multiple lines that fall into a single line:

```
LABEL:
            ; COMMENT
; COMMENT

                    <OPERATION>
   ; COMMENT
```

Similiar to above, but in a different format:

```
LABEL: ; COMMENT
   <OPERATION>                  ; COMMENT
   ; COMMENT
```

All three examples are equal, if the operations are equal.

TEXT AND DATA SECTIONS

Sections are ordered and used to outline section of particular kind. Sections of any kind can only be defined once. Currently only data and text sections are supported with no plans for supporting other sections.

The use of sections are optional and a assembly file containing no explicit sections, implicitly defines a text section spanning the whole file.

Assembly files must contain a text section and optionally can contain a non−empty data section.

Data sections must come before text sections.

Valid assembly file that includes a data and text section:

```
.data                   ; Outlines a data section spanning to the next text section
  <DIRECTIVE>
  <DIRECTIVE>

.text                   ; Outlines a text section spanning to the end of the file
  <INSTRUCTION>
  <INSTRUCTION>
```

Valid assembly file that only includes a text section:

```
.text
  <INSTRUCTION>
  <INSTRUCTION>
```

Valid assembly file that implicitly defines a text section:

```
START:
  <INSTRUCTION>
  <INSTRUCTION>
```

Invalid assembly file since the data section is empty (this may be changed in the future):

```
.data
.text
  <INSTRUCTION>
```

Invalid assembly file since it only contains a data section:

```
.data
  <DIRECTIVE>
  <DIRECTIVE>
```

See *DIRECTIVES*, *MACROS* and *INSTRUCTIONS* for details.

LABEL DEFINITIONS

TEST

OPERATIONS

TEST

TEST

TEST

REGISTERS

IMMEDIATES

DIRECTIVES

MACROS

INSTRUCTIONS

SEE ALSO

assembler(1)