```
The following assembler directives are supported

.data                - start of data segment
.text                - start of code segment
.code                - start of code segment (same as .text)
.org    <n>          - start address
.space  <n>          - leave n empty bytes
.asciiz <s>          - enters zero terminated ascii string
.ascii  <s>          - enter ascii string
.align  <n>          - align to n-byte boundary
.word   <n1>,<n2>..  - enters word(s) of data (64-bits)
.byte   <n1>,<n2>..  - enter bytes
.word32 <n1>,<n2>..  - enters 32 bit number(s)
.word16 <n1>,<n2>..  - enters 16 bit number(s)
.double <n1>,<n2>..  - enters floating-point number(s)

where <n> denotes a number like 24, <s> denotes a string like "fred", and
<n1>,<n2>.. denotes numbers seperated by commas. The integer registers can
be referred to as r0-r31, or R0-R31, or $0-$31 or using standard MIPS
pseudo-names, like $zero for r0, $t0 for r8 etc. Note that the size of an
immediate is limited to 16-bits. The maximum size of an immediate register
shift is 5 bits (so a shift by greater than 31 is illegal).

Floating point registers can be referred to as f0-f31, or F0-F31

The following instructions are supported

lb      - load byte
lbu     - load byte unsigned
sb      - store byte
lh      - load 16-bit half-word
lhu     - load 16-bit half word unsigned
sh      - store 16-bit half-word
lw      - load 32-bit word
lwu     - load 32-bit word unsigned
sw      - store 32-bit word
ld      - load 64-bit double-word
sd      - store 64-bit double-word
l.d     - load 64-bit floating-point
s.d     - store 64-bit floating-point
halt    - stops the program

daddi   - add immediate
daddui  - add immediate unsigned
andi    - logical and immediate
ori     - logical or immediate
xori    - exclusive or immediate
lui     - load upper half of register immediate

slti    - set if less than or equal immediate
sltiu   - set if less than or equal immediate unsigned

beq     - branch if pair of registers are equal
bne     - branch if pair of registers are not equal
beqz    - branch if register is equal to zero
bnez    - branch if register is not equal to zero

j       - jump to address
jr      - jump to address in register
jal     - jump and link to address (call subroutine)
jalr    - jump and link to address in register (call subroutine)

dsll    - shift left logical
dsrl    - shift right logical
```



```
dsra    - shift right arithmetic
dsllv   - shift left logical by variable amount
dsrlv   - shift right logical by variable amount
dsrav   - shift right arithmetic by variable amount
movz    - move if register equals zero
movn    - move if register not equal to zero
nop     - no operation
and     - logical and
or      - logical or
xor     - logical xor
slt     - set if less than
sltu    - set if less than unsigned
dadd    - add integers
daddu   - add integers unsigned
dsub    - subtract integers
dsubu   - subtract integers unsigned
dmul    - signed integer multiplication
dmulu   - unsigned integer multiplication
ddiv    - signed integer division
ddivu   - unsigned integer division

add.d   - add floating-point
sub.d   - subtract floating-point
mul.d   - multiply floating-point
div.d   - divide floating-point
mov.d   - move floating-point
cvt.d.l - convert 64-bit integer to a double floating-point format
cvt.l.d - convert double floating-point to a 64-bit integer format
c.lt.d  - set FP flag if less than
c.le.d  - set FP flag if less than or equal to
c.eq.d  - set FP flag if equal to
bc1f    - branch to address if FP flag is FALSE
bc1t    - branch to address if FP flag is TRUE
mtc1    - move data from integer register to floating-point register
mfc1    - move data from floating-point register to integer register

Memory Mapped I/O area

Addresses of CONTROL and DATA registers

CONTROL: .word32 0x10000
DATA:    .word32 0x10008

Set CONTROL = 1, Set DATA to Unsigned Integer to be output
Set CONTROL = 2, Set DATA to Signed Integer to be output
Set CONTROL = 3, Set DATA to Floating Point to be output
Set CONTROL = 4, Set DATA to address of string to be output
Set CONTROL = 5, Set DATA+5 to x coordinate, DATA+4 to y coordinate, and DATA to RGB
colour to be output
Set CONTROL = 6, Clears the terminal screen
Set CONTROL = 7, Clears the graphics screen
Set CONTROL = 8, read the DATA (either an integer or a floating-point) from the keyboard
Set CONTROL = 9, read one byte from DATA, no character echo.
```