winMIPS64 Instruction Set

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 separated 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 bits is illegal).

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

The following *instructions* are supported. Note *reg* is an integer register, *freg* is a floating-point (FP) register, and *imm* is an immediate value.

```
lb reg,imm(reg)       - load byte
lbu reg,imm(reg)      - load byte unsigned
sb reg,imm(reg)       - store byte
lh reg,imm(reg)       - load 16-bit half-word
lhu reg,imm(reg)      - load 16-bit half word unsigned
sh reg,imm(reg)       - store 16-bit half-word
lw reg,imm(reg)       - load 32-bit word
lwu reg,imm(reg)      - load 32-bit word unsigned
sw reg,imm(reg)       - store 32-bit word
ld reg,imm(reg)       - load 64-bit double-word
sd reg,imm(reg)       - store 64-bit double-word
l.d freg,imm(reg)     - load 64-bit floating-point
s.d freg,imm(reg)     - store 64-bit floating-point
halt                  - stops the program
daddi reg,reg,imm     - add immediate
daddui reg,reg,imm    - add immediate unsigned
andi reg,reg,imm      - logical and immediate
ori reg,reg,imm       - logical or immediate
xori reg,reg,imm      - exclusive or immediate
lui  reg,imm          - load upper half of register immediate
slti reg,reg,imm      - set if less than immediate
sltiu reg,reg,imm     - set if less than immediate unsigned
beq reg,reg,imm       - branch if pair of registers are equal
bne reg,reg,imm       - branch if pair of registers are not equal
beqz reg,imm          - branch if register is equal to zero
bnez reg,imm          - branch if register is not equal to zero
j imm                 - jump to address
```

```
jr reg               - jump to address in register
jal imm              - jump and link to address (call subroutine)
jalr reg             - jump and link to address in register
dsll reg,reg,imm     - shift left logical
dsrl reg,reg,imm     - shift right logical
dsra reg,reg,imm     - shift right arithmetic
dsllv reg,reg,reg    - shift left logical by variable amount
dsrlv reg,reg,reg    - shift right logical by variable amount
dsrav reg,reg,reg    - shift right arithmetic by variable amount
movz reg,reg,reg     - move if register equals zero
movn reg,reg,reg     - move if register not equal to zero
nop                  - no operation
and reg,reg,reg      - logical and
or reg,reg,reg       - logical or
xor reg,reg,reg      - logical xor
slt reg,reg,reg      - set if less than
sltu reg,reg,reg     - set if less than unsigned
dadd reg,reg,reg     - add integers
daddu reg,reg,reg    - add integers unsigned
dsub reg,reg,reg     - subtract integers
dsubu reg,reg,reg    - subtract integers unsigned
dmul reg,reg,reg     - signed integer multiplication
dmulu reg,reg,reg    - unsigned integer multiplication
ddiv reg,reg,reg     - signed integer division
ddivu reg,reg,reg    - unsigned integer division
add.d freg,freg,freg - add floating-point
sub.d freg,freg,freg - subtract floating-point
mul.d freg,freg,freg - multiply floating-point
div.d freg,freg,freg - divide floating-point
mov.d freg,freg      - move floating-point
cvt.d.l freg,freg    - convert 64-bit integer to a double FP format
cvt.l.d freg,freg    - convert double FP to a 64-bit integer format
c.lt.d freg,freg     - set FP flag if less than
c.le.d freg,freg     - set FP flag if less than or equal to
c.eq.d freg,freg     - set FP flag if equal to
bc1f imm             - branch to address if FP flag is FALSE
bc1t imm             - branch to address if FP flag is TRUE
mtc1 reg,freg        - move data from integer register to FP register
mfc1 reg,freg        - move data from FP register to integer register
```