Real – Pseudo

Instruction Suffixes

```
 i : immediate
 o : trap overflow
 u : unsigned (no sxt, no trap)
.s : single-precision       (FP)
.d : double-precision       (FP)
.w : word                (integer)
```

Integer

Arithmetic

Addition
```
add $t1, $t2, $t3
add $t1, $t2, -100
add $t1, $t2, 100000
addi $t1, $t2, -100
addi $t1, $t2, 100000
addiu $t1, $t2, -100
addiu $t1, $t2, 100000
addu $t1, $t2, $t3
addu $t1, $t2, 100000
```

Subtraction
```
sub $t1, $t2, $t3
sub $t1, $t2, -100
sub $t1, $t2, 100000
subi $t1, $t2, -100
subi $t1, $t2, 100000
subiu $t1, $t2, 100000
subu $t1, $t2, $t3
subu $t1, $t2, 100000
```

Multiplication
```
mult $t1, $t2
multu $t1, $t2
mul $t1, $t2, $t3
mul $t1, $t2, -100
mul $t1, $t2, 100000
mulo $t1, $t2, $t3
mulo $t1, $t2, -100
mulo $t1, $t2, 100000
mulou $t1, $t2, $t3
mulou $t1, $t2, -100
mulou $t1, $t2, 100000
mulu $t1, $t2, $t3
mulu $t1, $t2, -100
mulu $t1, $t2, 100000
```

Multiply-Add
```
madd $t1, $t2
maddu $t1, $t2
```

Multiply-Subtract
```
msub $t1, $t2
msubu $t1, $t2
```

Division
```
div $t1, $t2
div $t1, $t2, $t3
div $t1, $t2, -100
div $t1, $t2, 100000
divu $t1, $t2
divu $t1, $t2, $t3
divu $t1, $t2, -100
divu $t1, $t2, 100000
```

Remainder
```
rem $t1, $t2, $t3
rem $t1, $t2, -100
rem $t1, $t2, 100000
remu $t1, $t2, $t3
remu $t1, $t2, -100
remu $t1, $t2, 100000
```

Absolute Value (positive)
```
abs $t1, $t2
```

Negate (invert sign)
```
neg $t1, $t2
negu $t1, $t2
```

Comparison
```
seq $t1, $t2, $t3
seq $t1, $t2, -100
seq $t1, $t2, 100000
sne $t1, $t2, $t3
sne $t1, $t2, -100
sne $t1, $t2, 100000
sge $t1, $t2, $t3
sge $t1, $t2, -100
sge $t1, $t2, 100000
sgeu $t1, $t2, $t3
sgeu $t1, $t2, -100
sgeu $t1, $t2, 100000
sgt $t1, $t2, $t3
sgt $t1, $t2, -100
sgt $t1, $t2, 100000
sgtu $t1, $t2, $t3
sgtu $t1, $t2, -100
sgtu $t1, $t2, 100000
sle $t1, $t2, $t3
sle $t1, $t2, -100
sle $t1, $t2, 100000
sleu $t1, $t2, $t3
sleu $t1, $t2, -100
sleu $t1, $t2, 100000
slt $t1, $t2, $t3
slti $t1, $t2, -100
sltiu $t1, $t2, -100
sltu $t1, $t2, $t3
```

Floating Point

Arithmetic

Addition
```
add.d $f2, $f4, $f6
add.s $f0, $f1, $f3
```

Subtraction
```
sub.d $f2, $f4, $f6
sub.s $f0, $f1, $f3
```

Multiplication
```
mul.d $f2, $f4, $f6
mul.s $f0, $f1, $f3
```

Division
```
div.d $f2, $f4, $f6
div.s $f0, $f1, $f3
```

Absolute Value (positive)
```
abs.d $f2, $f4
abs.s $f0, $f1
```

Negate (invert sign)
```
neg.d $f2, $f4
neg.s $f0, $f1
```

Square root
```
sqrt.d $f2, $f4
sqrt.s $f0, $f1
```

Comparison
```
c.eq.d $f2, $f4
c.eq.d 1, $f2, $f4
c.eq.s $f0, $f1
c.eq.s 1, $f0, $f1
c.le.d $f2, $f4
c.le.d 1, $f2, $f4
c.le.s $f0, $f1
c.le.s 1, $f0, $f1
c.lt.d $f2, $f4
c.lt.d 1, $f2, $f4
c.lt.s $f0, $f1
c.lt.s 1, $f0, $f1
```

Conversion

Floor (Round down)
```
floor.w.d $f1, $f2
floor.w.s $f0, $f1
```

Ceiling (Round up)
```
ceil.w.d $f1, $f2
ceil.w.s $f0, $f1
```

Round (Nearest integer)
```
round.w.d $f1, $f2
round.w.s $f0, $f1
```

Truncate (Drop decimal)
```
trunc.w.d $f1, $f2
trunc.w.s $f0, $f1
```

Convert (cvt.<dest>.<src>) (Set mode in FCSR)
```
cvt.d.s $f2, $f1
cvt.d.w $f2, $f1
cvt.s.d $f1, $f2
cvt.s.w $f0, $f1
cvt.w.d $f1, $f2
cvt.w.s $f0, $f1
```

Bitwise

Bitwise AND
```
and $t1, $t2, $t3
and $t1, $t2, 100
and $t1, 100
andi $t1, $t2, 100
andi $t1, $t2, 100000
andi $t1, 100000
andi $t1, 100
```

Bitwise OR
```
or $t1, $t2, $t3
or $t1, $t2, 100
or $t1, 100
ori $t1, $t2, 100
ori $t1, $t2, 100000
ori $t1, 100000
ori $t1, 100
```

Bitwise XOR
```
xor $t1, $t2, $t3
xor $t1, $t2, 100
xor $t1, 100
xori $t1, $t2, 100
xori $t1, $t2, 100000
xori $t1, 100000
xori $t1, 100
```

Bitwise NOR
```
nor $t1, $t2, $t3
```

Bitwise NOT
```
not $t1, $t2
```

Shift LEFT
```
sll $t1, $t2, 10
sllv $t1, $t2, $t3
```

Shift RIGHT
```
srl $t1, $t2, 10
srlv $t1, $t2, $t3
sra $t1, $t2, 10
srav $t1, $t2, $t3
```

Rotate LEFT
```
rol $t1, $t2, $t3
rol $t1, $t2, 10
```

Rotate RIGHT
```
ror $t1, $t2, $t3
ror $t1, $t2, 10
```

Count leading ones/zeros
```
clo $t1, $t2
clz $t1, $t2
```

Control Transfer

Jump
```
j target
jal target
jalr $t1, $t2
jalr $t1
jr $t1
b label
```

Branch Integer (conditional jump)
```
beq $t1, $t2, label
beq $t1, -100, label
beq $t1, 100000, label
beqz $t1, label
bge $t1, $t2, label
bge $t1, -100, label
bge $t1, 100000, label
bgeu $t1, $t2, label
bgeu $t1, -100, label
bgeu $t1, 100000, label
bgez $t1, label
bgezal $t1, label
bgt $t1, $t2, label
bgt $t1, -100, label
bgt $t1, 100000, label
bgtu $t1, $t2, label
bgtu $t1, -100, label
bgtu $t1, 100000, label
bgtz $t1, label
ble $t1, $t2, label
ble $t1, -100, label
ble $t1, 100000, label
bleu $t1, $t2, label
bleu $t1, -100, label
bleu $t1, 100000, label
blez $t1, label
blt $t1, $t2, label
blt $t1, -100, label
blt $t1, 100000, label
bltu $t1, $t2, label
bltu $t1, -100, label
bltu $t1, 100000, label
bltz $t1, label
bltzal $t1, label
bne $t1, $t2, label
bne $t1, -100, label
bne $t1, 100000, label
bnez $t1, label
```

Branch floating-point (conditional jump)
```
bc1f 1, label
bc1f label
bc1t 1, label
bc1t label
```

Load (-> register)

Load Address
```
la $t1, ($t2)
la $t1, -100
la $t1, 100($t2)
la $t1, 100000($t2)
la $t1, 100000
la $t1, 100
la $t1, label($t2)
la $t1, label+100000($t2)
la $t1, label+100000
la $t1, label
```

Load Immediate
```
li $t1, -100
li $t1, 100000
li $t1, 100
lui $t1, 100
```

Load Memory

Load byte (8-bit)
```
lb $t1, ($t2)
lb $t1, -100($t2)
lb $t1, -100
lb $t1, 100($t2)
lb $t1, 100000($t2)
lb $t1, 100000
lb $t1, 100
lb $t1, label($t2)
lb $t1, label+100000($t2)
lb $t1, label+100000
lb $t1, label
lbu $t1, ($t2)
lbu $t1, -100($t2)
lbu $t1, -100
lbu $t1, 100($t2)
lbu $t1, 100000($t2)
lbu $t1, 100000
lbu $t1, 100
lbu $t1, label($t2)
lbu $t1, label+100000($t2)
lbu $t1, label+100000
lbu $t1, label
```

Load half-word (16-bit)
```
lh $t1, ($t2)
lh $t1, -100($t2)
lh $t1, -100
lh $t1, 100($t2)
lh $t1, 100000($t2)
lh $t1, 100000
lh $t1, 100
lh $t1, label($t2)
lh $t1, label+100000($t2)
lh $t1, label+100000
lh $t1, label
lhu $t1, ($t2)
lhu $t1, -100($t2)
lhu $t1, -100
lhu $t1, 100($t2)
lhu $t1, 100000($t2)
lhu $t1, 100000
lhu $t1, 100
lhu $t1, label($t2)
lhu $t1, label+100000($t2)
lhu $t1, label+100000
lhu $t1, label
```

Load word (32-bit)
```
lw $t1, ($t2)
lw $t1, -100($t2)
lw $t1, -100
lw $t1, 100($t2)
lw $t1, 100000($t2)
lw $t1, 100000
lw $t1, 100
lw $t1, label($t2)
lw $t1, label+100000($t2)
lw $t1, label+100000
lw $t1, label
```

Load double-word (64-bit)
```
ld $t1, ($t2)
ld $t1, -100($t2)
ld $t1, 100000($t2)
ld $t1, 100000
ld $t1, label($t2)
ld $t1, label+100000($t2)
ld $t1, label+100000
ld $t1, label
```

Load floating-point (64/32 bit)
```
l.d $f2, ($t2)
l.d $f2, -100
l.d $f2, 100000($t2)
l.d $f2, 100000
l.d $f2, label($t2)
l.d $f2, label+100000($t2)
l.d $f2, label+100000
l.d $f2, label
l.s $f1, ($t2)
l.s $f1, -100
l.s $f1, 100000($t2)
l.s $f1, 100000
l.s $f1, label($t2)
l.s $f1, label+100000($t2)
l.s $f1, label+100000
l.s $f1, label
```

Load to Coprocessor 1 (FPU)
```
lwc1 $f1, ($t2)
lwc1 $f1, -100($t2)
lwc1 $f1, -100
lwc1 $f1, 100000($t2)
lwc1 $f1, 100000
lwc1 $f1, label($t2)
lwc1 $f1, label+100000($t2)
lwc1 $f1, label+100000
lwc1 $f1, label
ldc1 $f2, ($t2)
ldc1 $f2, -100($t2)
ldc1 $f2, -100
ldc1 $f2, 100000($t2)
ldc1 $f2, 100000
ldc1 $f2, label($t2)
ldc1 $f2, label+100000($t2)
ldc1 $f2, label+100000
ldc1 $f2, label
```

Load Linked
```
ll $t1, ($t2)
ll $t1, -100($t2)
ll $t1, -100
ll $t1, 100($t2)
ll $t1, 100000($t2)
ll $t1, 100000
ll $t1, 100
ll $t1, label($t2)
ll $t1, label+100000($t2)
ll $t1, label+100000
ll $t1, label
```

Unaligned load half-word (16-bit)
```
ulh $t1, ($t2)
ulh $t1, -100($t2)
ulh $t1, 100000($t2)
ulh $t1, 100000
ulh $t1, label($t2)
ulh $t1, label+100000($t2)
ulh $t1, label+100000
ulh $t1, label
ulhu $t1, ($t2)
ulhu $t1, -100($t2)
ulhu $t1, 100000($t2)
ulhu $t1, 100000
ulhu $t1, label($t2)
ulhu $t1, label+100000($t2)
ulhu $t1, label+100000
ulhu $t1, label
```

Unaligned load word (32-bit)
```
ulw $t1, ($t2)
ulw $t1, -100($t2)
ulw $t1, 100000($t2)
ulw $t1, 100000
ulw $t1, label($t2)
ulw $t1, label+100000($t2)
ulw $t1, label+100000
ulw $t1, label
```

Load word LEFT (32-bit)
```
lwl $t1, ($t2)
lwl $t1, -100($t2)
lwl $t1, -100
lwl $t1, 100($t2)
lwl $t1, 100000($t2)
lwl $t1, 100000
lwl $t1, 100
lwl $t1, label($t2)
lwl $t1, label+100000($t2)
lwl $t1, label+100000
lwl $t1, label
```

Load word RIGHT (32-bit)
```
lwr $t1, ($t2)
lwr $t1, -100($t2)
lwr $t1, -100
lwr $t1, 100($t2)
lwr $t1, 100000($t2)
lwr $t1, 100000
lwr $t1, 100
lwr $t1, label($t2)
lwr $t1, label+100000($t2)
lwr $t1, label+100000
lwr $t1, label
```

Store (register -> memory)

Store byte (8-bit)
```
sb $t1, ($t2)
sb $t1, -100($t2)
sb $t1, -100
sb $t1, 100($t2)
sb $t1, 100000($t2)
sb $t1, 100000
sb $t1, 100
sb $t1, label($t2)
sb $t1, label+100000($t2)
sb $t1, label+100000
sb $t1, label
```

Store half-word (16-bit)
```
sh $t1, ($t2)
sh $t1, -100($t2)
sh $t1, -100
sh $t1, 100($t2)
sh $t1, 100000($t2)
sh $t1, 100000
sh $t1, 100
sh $t1, label($t2)
sh $t1, label+100000($t2)
sh $t1, label+100000
sh $t1, label
```

Store word (32-bit)
```
sw $t1, ($t2)
sw $t1, -100($t2)
sw $t1, -100
sw $t1, 100($t2)
sw $t1, 100000($t2)
sw $t1, 100000
sw $t1, 100
sw $t1, label($t2)
sw $t1, label+100000($t2)
sw $t1, label+100000
sw $t1, label
```

Store double-word (64-bit)
```
sd $t1, ($t2)
sd $t1, -100($t2)
sd $t1, 100000($t2)
sd $t1, 100000
sd $t1, label($t2)
sd $t1, label+100000($t2)
sd $t1, label+100000
sd $t1, label
```

Store floating-point (64/32 bit)
```
s.d $f2, ($t2)
s.d $f2, -100
s.d $f2, 100000($t2)
s.d $f2, 100000
s.d $f2, label($t2)
s.d $f2, label+100000($t2)
s.d $f2, label+100000
s.d $f2, label
s.s $f1, ($t2)
s.s $f1, -100
s.s $f1, 100000($t2)
s.s $f1, 100000
s.s $f1, label($t2)
s.s $f1, label+100000($t2)
s.s $f1, label+100000
s.s $f1, label
```

Store from Coprocessor 1 (FPU)
```
swc1 $f1, ($t2)
swc1 $f1, -100($t2)
swc1 $f1, -100
swc1 $f1, 100000($t2)
swc1 $f1, 100000
swc1 $f1, label($t2)
swc1 $f1, label+100000($t2)
swc1 $f1, label+100000
swc1 $f1, label
sdc1 $f2, ($t2)
sdc1 $f2, -100($t2)
sdc1 $f2, -100
sdc1 $f2, 100000($t2)
sdc1 $f2, 100000
sdc1 $f2, label($t2)
sdc1 $f2, label+100000($t2)
sdc1 $f2, label+100000
sdc1 $f2, label
```

Store conditional
```
sc $t1, ($t2)
sc $t1, -100($t2)
sc $t1, -100
sc $t1, 100($t2)
sc $t1, 100000($t2)
sc $t1, 100000
sc $t1, 100
sc $t1, label($t2)
sc $t1, label+100000($t2)
sc $t1, label+100000
sc $t1, label
```

Unaligned store half-word (16-bit)
```
ush $t1, ($t2)
ush $t1, -100($t2)
ush $t1, 100000($t2)
ush $t1, 100000
ush $t1, label($t2)
ush $t1, label+100000($t2)
ush $t1, label+100000
ush $t1, label
```

Unaligned store word (32-bit)
```
usw $t1, ($t2)
usw $t1, -100($t2)
usw $t1, 100000($t2)
usw $t1, 100000
usw $t1, label($t2)
usw $t1, label+100000($t2)
usw $t1, label+100000
usw $t1, label
```

Store word LEFT (32-bit)
```
swl $t1, -100($t2)
swl $t1, -100
swl $t1, 100($t2)
swl $t1, 100000($t2)
swl $t1, 100000
swl $t1, 100
swl $t1, label($t2)
swl $t1, label+100000($t2)
swl $t1, label+100000
swl $t1, label
swl $t1, ($t2)
```

Store word RIGHT (32-bit)
```
swr $t1, ($t2)
swr $t1, -100($t2)
swr $t1, -100
swr $t1, 100($t2)
swr $t1, 100000($t2)
swr $t1, 100000
swr $t1, 100
swr $t1, label($t2)
swr $t1, label+100000($t2)
swr $t1, label+100000
swr $t1, label
```

Move (register -> register)

Move
```
move $t1, $t2
```

Move to/from hi/lo
```
mthi $t1
mfhi $t1
mtlo $t1
mflo $t1
```

Move to/from Coprocessor 0
```
mtc0 $t1, $8
mfc0 $t1, $8
```

Move to/from Coprocessor 1 (FPU)
```
mtc1 $t1, $f1
mtc1.d $t1, $f2
mfc1 $t1, $f1
mfc1.d $t1, $f2
```

Floating-point move
```
mov.d $f2, $f4
mov.s $f0, $f1
```

Conditional move (register)
```
movz $t1, $t2, $t3
movn $t1, $t2, $t3
```

Conditional move (condition flag true)
```
movt $t1, $t2, 1
movt $t1, $t2
```

Conditional move (condition flag false)
```
movf $t1, $t2, 1
movf $t1, $t2
```

Conditional floating-point move (register)
```
movz.d $f2, $f4, $t3
movz.s $f0, $f1, $t3
movn.d $f2, $f4, $t3
movn.s $f0, $f1, $t3
```

Conditional floating-point move (condition flag true)
```
movt.d $f2, $f4, 1
movt.d $f2, $f4
movt.s $f0, $f1, 1
movt.s $f0, $f1
```

Conditional floating-point move (condition flag false)
```
movf.d $f2, $f4, 1
movf.d $f2, $f4
movf.s $f0, $f1, 1
movf.s $f0, $f1
```

Other

No operation
```
nop
```

System call
```
syscall
```

Conditional trap
```
teq $t1, $t2
teqi $t1, -100
tge $t1, $t2
tgei $t1, -100
tgeiu $t1, -100
tgeu $t1, $t2
tlt $t1, $t2
tlti $t1, -100
tltiu $t1, -100
tltu $t1, $t2
tne $t1, $t2
tnei $t1, -100
```

Error return
```
eret
```

End program
```
break 100
break
```

The information in this document has been extracted from the source code of the MARS 4.5 simulator.