MIPS Instruction Format

MIPS uses a 32-bit fixed-length instruction format. There are only three different instruction word formats:

Register format

Op-code	Rs	Rt	Rd		Function code
000000	sssss	ttttt	ddddd	00000	ffffff

Immediate format

Op-code	Rs	Rt	Immediate
ffffff	sssss	ttttt	iiiiiiiiiiiiiiii

Jump format

Op-code	Target
000010	tttttttttttttttttttttttttt

The CPU examines the 6-bit *op-code* field to determine the type of instruction. Depending on the op-code, the remaining 26 bits are interpreted in three possible ways.

MIPS Instruction Categories

Arithmetic Addition, subtraction, *etc.*

Data Transfer Moving data between memory and registers.

Logical And, or, *etc.*

Shift Left and right shifts.

Conditional Branch Make decisions based on current state.

Unconditional Jump Change the flow of control.

MIPS Arithmetic Instructions

Register Format

Add

000000	sssss	ttttt	ddddd	00000	100000

`add Rd,Rs,Rt`

Add Rs to Rt placing result in Rd, trap if overflow.

Add unsigned

000000	sssss	ttttt	ddddd	00000	100001

`addu Rd,Rs,Rt`

Add Rs to Rt placing result in Rd, ignore overflow.

Subtract

000000	sssss	ttttt	ddddd	00000	100010

`sub Rd,Rs,Rt`

Subtract Rt from Rs placing result in Rd, trap if overflow.

Subtract unsigned

000000	sssss	ttttt	ddddd	00000	100011

`subu Rd,Rs,Rt`

Subtract Rt from Rs placing result in Rd, ignore overflow.

Multiply

000000	sssss	ttttt	00000	00000	011000

`mult Rs,Rt`

Two's complement multiply Rs by Rt placing 64-bit result in (Hi, Lo).

Multiply Unsigned

000000	sssss	ttttt	00000	00000	011001

`multu Rs,Rt`

Unsigned multiply Rs by Rt placing 64-bit result in (Hi, Lo).

Divide

000000	sssss	ttttt	00000	00000	011010

`div Rs,Rt`

Two's complement divide Rs by Rt placing quotient in Lo and remainder in Hi.

Divide unsigned

000000	sssss	ttttt	00000	00000	011011

`divu Rs,Rt`

Unsigned divide Rs by Rt placing quotient in Lo and remainder in Hi.

Immediate Format

Add immediate

001000	sssss	ttttt	iiiiiiiiiiiiiiii

`addi Rt,Rs,Imm`

Add Rs to sign-extended immediate value, placing result in Rt, trap if overflow.

Add immediate unsigned

001001	sssss	ttttt	iiiiiiiiiiiiiiii

`addiu Rt,Rs,Imm`

Add Rs to sign-extended immediate value, placing result in Rt, ignore overflow.

MIPS Data Transfer Instructions

Pseudo Instructions

Load Immediate

`li Rd,immed`

Move the immediate value imm into register Rd.

Load Address

`la Rd, address`

Load computed address - not the contents of the location- into register Rd.

Immediate Format

Load byte

100000	sssss	ttttt	iiiiiiiiiiiiiiii

`lb Rt,Imm(Rs)`

Add Rs to sign-extended immediate value to obtain effective address. Read 8-bit byte from memory at effective address, sign-extend, and place result in Rt.

Load halfword

100001	sssss	ttttt	iiiiiiiiiiiiiiii

`lh Rt,Imm(Rs)`

Add Rs to sign-extended immediate value to obtain effective address (trap if odd). Read 16-bit half word from memory at effective address, sign-extend, and place result in Rt.

Load word left

100010	sssss	ttttt	iiiiiiiiiiiiiiii

```
lwl Rt,Imm(Rs)
```

Add Rs to sign-extended immediate value to obtain effective address. Read 8-bit bytes from memory starting at effective address and decreasing to first byte of word. Store bytes left-justified in Rt.

Load word

100011	sssss	ttttt	iiiiiiiiiiiiiiii

```
lw Rt,Imm(Rs)
```

Add Rs to sign-extended immediate value to obtain effective address (trap if not multiple of 4). Read 32-bit word from memory at effective address and place result in Rt.

Load byte unsigned

100100	sssss	ttttt	iiiiiiiiiiiiiiii

```
lbu Rt,Imm(Rs)
```

Add Rs to sign-extended immediate value to obtain effective address. Read 8-bit byte from memory at effective address, zero-extend, and place result in Rt.

Load halfword unsigned

100101	sssss	ttttt	iiiiiiiiiiiiiiii

`lhu Rt,Imm(Rs)`

Add Rs to sign-extended immediate value to obtain effective address (trap if odd). Read 16-bit half word from memory at effective address, zero-extend, and place result in Rt.

Load word right

100110	sssss	ttttt	iiiiiiiiiiiiiiii

`lwr Rt,Imm(Rs)`

Add Rs to sign-extended immediate value to obtain effective address. Read 8-bit bytes from memory starting at effective address and increasing to last byte of word. Store bytes right-justified in Rt.

Store byte

101000	sssss	ttttt	iiiiiiiiiiiiiiii

`sb Rt,Imm(Rs)`

Add Rs to sign-extended immediate value to obtain effective address. Store least-significant 8-bit byte from Rt in memory at effective address.

Store halfword

101001	sssss	ttttt	iiiiiiiiiiiiiiii

`sh Rt,Imm(Rs)`

Add Rs to sign-extended immediate value to obtain effective address (trap if odd). Write least-significant two bytes of Rt in memory at effective address.

Store word left

101010	sssss	ttttt	iiiiiiiiiiiiiiii

`swl Rt,Imm(Rs)`

Add Rs to sign-extended immediate value to obtain effective address. Write bytes from Rt left-justified into memory, starting with the most significant byte and continuing to a word boundary.

Store word

101011	sssss	ttttt	iiiiiiiiiiiiiiii

`sw Rt,Imm(Rs)`

Add Rs to sign-extended immediate value to obtain effective address (trap if not multiple of 4). Store 32-bit contents of Rt in memory at effective address.

Store word right

101110	sssss	ttttt	iiiiiiiiiiiiiiii

```
swr Rt,Imm(Rs)
```

Add Rs to sign-extended immediate value to obtain effective address. Write bytes from Rt right-justified into memory, starting with the least significant byte and continuing to a word boundary.

Load upper immediate

001111	00000	ttttt	iiiiiiiiiiiiiiii

```
lui Rt,Imm
```

The 16-bit immediate value is shifted left 16-bits (zero-filled) and the result is stored in Rt.

Register Format

Move from high

000000	00000	00000	ddddd	00000	010000

`mfhi Rd`

The contents of the special register Hi are copied into Rd.

Move to high

000000	sssss	00000	00000	00000	010001

`mthi Rd`

The contents of Rs are copied into the special register Hi.

Move from low

000000	00000	00000	ddddd	00000	010010

`mflo Rd`

The contents of the special register Lo are copied into Rd.

Move to low

000000	sssss	00000	00000	00000	010011

`mtlo Rd`

The contents of Rs are copied into the special register Lo.

Move from control register

010000	00000	ttttt	ddddd	00000	000000

`mfc0 Rt,Cd`

The contents of coprocessor 0 control register Cd are copied into register Rt.

Move to control register

010000	00100	ttttt	ddddd	00000	000000

`mtc0 Rt,Cd`

The contents of register Rt are copied to coprocessor 0 control register Cd.

Pseudo Instruction

Move

`move Rd,Rs`

Move register Rs to register Rd.

MIPS Logical Instructions

Register Format

And

000000	sssss	ttttt	ddddd	00000	100100

`and Rd,Rs,Rt`

The contents of Rs are bitwise AND-ed with the contents of Rt and the results are placed in Rd.

Or

000000	sssss	ttttt	ddddd	00000	100101

`or Rd,Rs,Rt`

The contents of Rs are bitwise OR-ed with the contents of Rt and the results are placed in Rd.

Exclusive or

000000	sssss	ttttt	ddddd	00000	100110

`xor Rd,Rs,Rt`

The contents of Rs are bitwise XOR-ed with the contents of Rt and the results are placed in Rd.

Nor

000000	sssss	ttttt	ddddd	00000	100111

`nor Rd,Rs,Rt`

The contents of Rs are bitwise NOR-ed with the contents of Rt and the results are placed in Rd.

Set on less than

000000	sssss	ttttt	ddddd	00000	101010

`slt Rd,Rs,Rt`

The contents of Rs are two's-complement compared with the contents of Rt. If Rs is less than Rt, then Rd is set to one, otherwise Rd is set to zero.

Set on less than unsigned

000000	sssss	ttttt	ddddd	00000	101011

`sltu Rd,Rs,Rt`

The contents of Rs are compared in an unsigned manner with the contents of Rt. If Rs is less than Rt, then Rd is set to one, otherwise Rd is set to zero.

Immediate Format

Set on less than immediate

001010	sssss	ttttt	iiiiiiiiiiiiiiii

`slti Rt,Rs,Imm`

The contents of Rs are two's-complement compared with the sign-extended immediate value. If Rs is less than the immediate value, then Rt is set to one, otherwise Rt is set to zero.

Set on less than immediate unsigned

001011	sssss	ttttt	iiiiiiiiiiiiiiii

`sltiu Rt,Rs,Imm`

The contents of Rs are compared in an unsigned manner with the zero-extended immediate value. If Rs is less than the immediate value, then Rt is set to one, otherwise Rt is set to zero.

And immediate

001100	sssss	ttttt	iiiiiiiiiiiiiiii

`andi Rt,Rs,Imm`

The contents of Rs are bitwise AND-ed with the the zero-extended immediate value and the results are placed in Rt.

Or immediate

001101	sssss	ttttt	iiiiiiiiiiiiiiii

`ori Rt,Rs,Imm`

The contents of Rs are bitwise OR-ed with the the zero-extended immediate value and the results are placed in Rt.

Xor immediate

001110	sssss	ttttt	iiiiiiiiiiiiiiii

`xori Rt,Rs,Imm`

The contents of Rs are bitwise XOR-ed with the the zero-extended immediate value and the results are placed in Rt.

Pseudo Instructions

Negate

```
neg Rd,Rs
```

Put the negative value of register Rs in register Rd.

Absolute Value

```
abs Rd,Rs
```

Put the absolute value of register Rs in register Rd.

Bitwise NOT

```
not Rd,Rs
```

Put the bitwise logical negation of register Rs in register Rd.

Negate

```
neg Rd,Rs
```

Put the negative value of register Rs in register Rd.

Set on Equal

```
seq Rd,Rs,Rt
```

Set register Rd to 1 if register Rs equals register Rt, and to 0 otherwise.

Set on Greater than Equal

`sge Rd,Rs,Rt`

Set register Rd to 1 if register Rs greater than or equal to register Rt, and to 0 otherwise.

Set on Greater Than

`sgt Rd,Rs,Rt`

Set register Rd to 1 if register Rs greater than register Rt, and to 0 otherwise.

Set on Less Than Equal

`sle Rd,Rs,Rt`

Set register Rd to 1 if register Rs less than or equal to register Rt, and to 0 otherwise.

Set Not Equal

`sne Rd,Rs,Rt`

Set register Rd to 1 if register Rs not equal to register Rt, and to 0 otherwise.

MIPS Shift Instructions

Register Format

Shift left logical

000000	00000	ttttt	ddddd	iiiii	000000

`sll Rd,Rt,i`

The contents of Rt are shifted left i bits and the result is placed in Rd.

Shift right logical

000000	00000	ttttt	ddddd	iiiii	000010

`srl Rd,Rt,i`

The contents of Rt are shifted right i bits (zero-filled) and the result is placed in Rd.

Shift right arithmetic

000000	00000	ttttt	ddddd	iiiii	000011

`sra Rd,Rt,i`

The contents of Rt are shifted right i bits (sign-filled) and the result is placed in Rd.

Shift left logical variable

000000	sssss	ttttt	ddddd	00000	000100

`sllv Rd,Rt,Rs`

The contents of Rt are shifted left by a number of bits specified by the low-order five bits of Rs and the result is placed in Rd.

Shift right logical variable

000000	sssss	ttttt	ddddd	00000	000110

`srlv Rd,Rt,Rs`

The contents of Rt are shifted right (zero-filled) by a number of bits specified by the low-order five bits of Rs and the result is placed in Rd.

Shift right arithmetic variable

000000	sssss	ttttt	ddddd	00000	000111

`srav Rd,Rt,Rs`

The contents of Rt are shifted right (sign-filled) by a number of bits specified by the low-order five bits of Rs and the result is placed in Rd.

Pseudo Instructions

Rotate Right

```
ror Rd,Rs,Rt
```

Rotate register Rs right by the number of bits indicated by Rt and put the result in register Rd.

Rotate Left

```
rol Rd,Rs,Rt
```

Rotate register Rs left by the number of bits indicated by Rt and put the result in register Rd.

MIPS Conditional Branch Instructions

These instructions modify the program counter to effect a change in the flow of control. The immediate operand specifies a *word offset* from the current program counter value. Thus, the new program counter is computed by taking the sign-extended immediate operand, shifting it left by two bits, and adding it to the current program counter value, which has already been incremented at the time the calculation occurs.

Immediate Format

Branch on equal

000100	sssss	ttttt	iiiiiiiiiiiiiiii

`beq Rs,Rt,addr`

The program counter receives the new value if the contents of Rs and Rt are equal.

Branch on not equal

000101	sssss	ttttt	iiiiiiiiiiiiiiii

`bne Rs,Rt,addr`

The program counter receives the new value if the contents of Rs and Rt are not equal.

Branch on less than or equal to zero.

000110	sssss	00000	iiiiiiiiiiiiiiii

`blez Rs,addr`

The program counter receives the new value if the content of Rs is less than or equal to zero.

Branch on greater than zero

000111	sssss	00000	iiiiiiiiiiiiiiii

`bgtz Rs,addr`

The program counter receives the new value if the content of Rs is greater than zero.

Branch on less than zero

000001	sssss	00000	iiiiiiiiiiiiiiii

`bltz Rs,addr`

The program counter receives the new value if the content of Rs is less than zero.

Branch on greater than or equal to zero

000001	sssss	00001	iiiiiiiiiiiiiiii

`bgez Rs,addr`

The program counter receives the new value if the content of Rs is greater than or equal to zero.

Branch on less than zero and link

000001	sssss	10000	iiiiiiiiiiiiiiii

`bltzal Rs,addr`

The program counter receives the new value if the content of Rs is less than zero. The old program counter is saved in `$ra`.

Branch on greater than or equal to zero and link

000001	sssss	10001	iiiiiiiiiiiiiiii

`bgezal Rs,addr`

The program counter receives the new value if the content of Rs is greater than or equal to zero. The old program counter is saved in `$ra`.

Pseudo Instruction

Branch on equal to zero

`beqz Rs,addr`

The program counter receives the new value if the content of register Rs is zero.

Branch on greater than or equal

`bge Rs,Rt,addr`

The program counter receives the new value if the content of register Rs is greater than or equal to register Rt.

Branch on greater than

`bgt Rs,Rt,addr`

The program counter receives the new value if the content of register Rs is greater than to register Rt.

Branch on less than or equal

`ble Rs,Rt,addr`

The program counter receives the new value if the content of register Rs is less than or equal to register Rt.

Branch on less than or equal

`blt Rs,Rt,addr`

The program counter receives the new value if the content of register Rs is less than register Rt.

Branch on equal to zero

`bnez Rs,addr`

The program counter receives the new value if the content of register Rs is not equal to zero.

MIPS Unconditional Jump Instructions

Register Format

Jump register

000000	sssss	00000	00000	00000	001000

`jr Rs`

The program counter is replaced by the contents of Rs.

Jump and link register

000000	sssss	00000	ddddd	00000	001001

`jalr Rd,Rs`

The program counter is replaced by the contents of Rs. The old program counter is saved in Rd.

System call

000000	00000	00000	00000	00000	001100

`syscall`

A user program exception is generated, which causes a trap to the operating system.

Jump Format

These instructions replace the low-order 28-bits of the program counter with a value obtained by shifting the 26-bit immediate operand left by two bits. This can be used to effect transfer of control to anywhere within the same 256MB region in which execution is currently taking place.

Jump

000010	tttttttttttttttttttttttttt

`j addr`

The program counter is unconditionally modified as described above.

Jump and link

000011	tttttttttttttttttttttttttt

`jal addr`

The program counter is unconditionally modified as described above. The old program counter is saved in `$ra`.