Table 1

| | | | MIPS ASSEMBLY INSTRUCTIONS | TYPE | OPCODE | SOURCE 1 | SOURCE 2 | DEST | SHIFT AMT | Function | Description |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | op | rs | rt | rd | shamt | funct | |
| ALU | VALUE | | | | [31:26] (6) | [25:21] (5) | [20:16] (5) | [15:11] (5) | [10:6] (5) | [5:0] (6) | |
| X | 5’H0 | nop | No Operation | R | 6’H0 | 5'H0 | 5'H0 | 5'H0 | 5'H0 | 6’H0 | Do Nothing |
| | | sll | Logical Shift Left | R | 6’H0 | rs | rt | rd | shamt | 6’H0 | rd = rt << shamt |
| | | srl | Logical Shift Right (0-extended) | R | 6’H0 | rs | rt | rd | shamt | 6’H2 | rd = rt >>> shamt (zero fill) |
| | | sra | Arithmetic Shift Right (sign-extended) | R | 6’H0 | rs | rt | rd | shamt | 6’H3 | rd = rt >> shamt (keep sign bit) |
| | | sllv | Shift Left Logical Variable | R | 6’H0 | rs | rt | rd | 5'H0 | 6’H4 | rd = rt <<< rs |
| | | srlv | Shift Right Logical Variable | R | 6’H0 | rs | rt | rd | 5'H0 | 6’H6 | rd = rt >>> rs |
| | | srav | Shift Right Arithmetic Variable | R | 6’H0 | rs | rt | rd | 5'H0 | 6’H7 | rd = rt >> rs |
| | | jr | Jump to Address in Register | R | 6’H0 | rs | 5'H0 | 5'H0 | 5'H0 | 6’H8 | PC = nPC; nPV = $rs |
| | | jalr | Jump and Link Register | R | 6’H0 | rs | 5'H0 | 5'H0 | 5'H0 | 6’H9 | $31 = PC; PC = rs |
| | | movz | Move Conditional Zero | R | 6’H0 | rs | rt | rd | 5'H0 | 6'HA | Move rs to rd if rt == 0 |
| | | movn | Move Conditional Not Zero | R | 6’H0 | rs | rt | rd | 5'H0 | 6'HB | Move rs to rd if rt != 1 |
| | | mfhi | Move from HI Register | R | 6’H0 | 5'H0 | 5'H0 | rd | 5'H0 | 6'H12 | $d = $HI |
| | | mthi | Move to HI | R | 6’H0 | rs | 15’H0 | | | 6'H11 | Move rs to HI |
| | | mflo | Move from LO Register | R | 6’H0 | 5'H0 | 5'H0 | rd | 5'H0 | 6'H12 | $d = $LO |
| | | mtlo | Move to LO | R | 6’H0 | rs | 15’H0 | | | 6'H13 | Move rs to LO |
| | | mult | Multiply | R | 6’H0 | rs | rt | rd | shamt | 6'H18 | HI:LO = r2 * rt |
| | | multu | Unsigned Multiply | R | 6’H0 | rs | rt | rd | shamt | 6'H19 | HI:LO = r2 * rt |
| | | div | Divide | R | 6’H0 | rs | rt | rd | shamt | 6'H1A | LO = rs / rt; HI = rs % rt |
| | | divu | Unsigned Divide | R | 6’H0 | rs | rt | rd | shamt | 6'H1B | LO = rs / rt; HI = rs % rt |
| X | 5'H1 | add | Add | R | 6’H0 | rs | rt | rd | shamt | 6'H20 | rd = rs + rt (with overflow) |
| X | 5'H2 | addu | Add Unsigned | R | 6’H0 | rs | rt | rd | shamt | 6'H21 | rd = rs + rt (without overflow) |
| X | 5'H3 | sub | Subtract | R | 6’H0 | rs | rt | rd | shamt | 6'H22 | rd = rs - rt |
| X | 5'H4 | subu | Unsigned Subtract | R | 6’H0 | rs | rt | rd | shamt | 6'H23 | rd = rs - rt |
| X | 5'H5 | and | Bitwise AND | R | 6’H0 | rs | rt | rd | shamt | 6'H24 | rd = rs & rt |
| X | 5'H6 | or | Bitwise OR | R | 6’H0 | rs | rt | rd | shamt | 6'H25 | rd = rs \| rt |
| X | 5'H7 | xor | Bitwise XOR | R | 6’H0 | rs | rt | rd | shamt | 6'H26 | rd = rs ^ rt |
| X | 5'H8 | nor | Bitwise NOR | R | 6’H0 | rs | rt | rd | shamt | 6'H27 | rd = ~(rs \| rt) |
| X | 5'H9 | slt | Set to 1 if Less Than Signed | R | 6’H0 | rs | rt | rd | 5'H0 | 6'H2A | if rs < rt then rd = 1 |
| X | 5'HA | sltu | Set to 1 if Less than Unsigned | R | 6’H0 | rs | rt | rd | 5'H0 | 6'H2B | if rs < rt then rd = 1 |
| | | bltz | Branch Less Than Zero | | 6'H1 | rs | 5'H0 | address/immediate | | | If rs < 0, PC += (address << 2) |
| | | bgez | Branch Greater Than or Equal Zero | | 6'H1 | rs | 5'H1 | address/immediate | | | If rs >= 0, PC += (address << 2) |
| | | bltzal | Branch on Less Than - Link | | 6'H1 | rs | 5'H10 | address/immediate | | | if rs < 0; $31 = PC; PC += (target address << 2) |
| | | bgezal | Branch on Greater than Equal Zero - Link | | 6'H1 | rs | 5'H11 | address/immediate | | | if rs >= 0; $31 = PC; PC += (target address << 2) |
| | | j | Jump to Address | J | 6'H2 | target address | | | | | PC += (target address << 2) |
| | | jal | Jump and Link | J | 6'H3 | target address | | | | | $31 = PC; PC += (target address << 2) |
| | | beq | Branch if Equal | I | 6'H4 | rs | rt | address/immediate | | | if rs == rt advance pc (offset << 2) |
| | | bne | Branch if Not Equal | I | 6'H5 | rs | rt | address/immediate | | | if rs != rt advance pc (offset << 2) |
| | | blez | Branch on Less than or equal to zero | | 6'H6 | rs | 5'H0 | address/immediate | | | if rs <= 0 advance pc (offset << 2) |
| | | bgtz | Branch on Greater than Zero | | 6'H7 | rs | 5'H0 | address/immediate | | | if rs > 0 advance pc (offset << 2) |
| X | 5'HB | addi | Add Immediate | I | 6'H8 | rs | rt | address/immediate | | | rt = rs + immediate (with overflow) |
| X | 5'HC | adddiu | Add Unsigned immediate | I | 6'H9 | rs | rt | address/immediate | | | rt = rs + immediate (without overflow) |
| X | 5'HD | slti | Set to 1 if Less Than Immediate | I | 6'HA | rs | rt | address/immediate | | | if rs < immediate then rt = 1 |
| X | 5'HE | sltiu | Set to 1 if Less than Unsigned Immediate | I | 6'HB | rs | rt | address/immediate | | | if rs < immediate then rt = 1 |
| X | 5'HF | andi | Bitwise AND Immediate | I | 6'HC | rs | rt | address/immediate | | | rt = rs & immediate |
| X | 5'H10 | ori | Bitwise OR Immediate | I | 6'hD | rs | rt | address/immediate | | | rt = rs \| immediate |
| X | 5'H11 | xori | Bitwise XOR Immediate | I | 6'HE | rs | rt | address/immediate | | | rt = rs ^ immediate |
| | | lui | Load Upper Immediate | I | 6'HF | rs | rt | address/immediate | | | rt = (immediate << 16), zero rest |
| | | lb | Load Byte | | 6'H20 | rs | 5'H0 | offset | | | rt = MEM[rs + offset] |
| | | lh | Load Halfword | | 6'H21 | rs | 5'H0 | offset | | | rt = MEM[rs + offset] |
| | | lw | Load Word | I | 6'H23 | rs | rt | offset | | | rt = MEM[rs + offset] |
| | | lbu | Load Byte Unsigned | I | 6'H24 | rs | rt | offset | | | rt gets LSB of MEM[rs+offset] |
| | | lhu | Load Halfword Unsigned | I | 6'H25 | rs | rt | offset | | | rt gets LSHW of MEM[rs+offset] |
| | | sb | Store Byte | I | 6'H28 | rs | rt | offset | | | LSB of rt stored at MEM[rs+offset] |
| | | sh | Store Halfword | I | 6'H29 | rs | rt | offset | | | LSHW of rt stored at MEM[rs+offset] |
| | | sw | Store Word | I | 6’H2B | rs | rt | offset | | | MEM[rs+offset] = rt |