

MIPS Reference Data

①



CORE INSTRUCTION SET

| NAME, MNEMONIC | FOR-MAT | OPERATION (in Verilog) | OPCODE / FUNCT (Hex) |
|-----------------------------|---------|--|---------------------------|
| Add | add R | $R[rd] = R[rs] + R[rt]$ | (1) 0 / 20 _{hex} |
| Add Immediate | addi I | $R[rt] = R[rs] + \text{SignExtImm}$ | (1,2) 8 _{hex} |
| Add Imm. Unsigned | addiu I | $R[rt] = R[rs] + \text{SignExtImm}$ | (2) 9 _{hex} |
| Add Unsigned | addu R | $R[rd] = R[rs] + R[rt]$ | 0 / 21 _{hex} |
| And | and R | $R[rd] = R[rs] \& R[rt]$ | 0 / 24 _{hex} |
| And Immediate | andi I | $R[rt] = R[rs] \& \text{ZeroExtImm}$ | (3) c _{hex} |
| Branch On Equal | beq I | if($R[rs] == R[rt]$) $PC = PC + 4 + \text{BranchAddr}$ | (4) 4 _{hex} |
| Branch On Not Equal | bne I | if($R[rs] != R[rt]$) $PC = PC + 4 + \text{BranchAddr}$ | (4) 5 _{hex} |
| Jump | j J | $PC = \text{JumpAddr}$ | (5) 2 _{hex} |
| Jump And Link | jal J | $R[31] = PC + 8; PC = \text{JumpAddr}$ | (5) 3 _{hex} |
| Jump Register | jr R | $PC = R[rs]$ | 0 / 08 _{hex} |
| Load Byte Unsigned | lbu I | $R[rt] = \{24'b0, M[R[rs] + \text{SignExtImm}](7:0)\}$ | (2) 24 _{hex} |
| Load Halfword Unsigned | lhu I | $R[rt] = \{16'b0, M[R[rs] + \text{SignExtImm}](15:0)\}$ | (2) 25 _{hex} |
| Load Linked | ll I | $R[rt] = M[R[rs] + \text{SignExtImm}]$ | (2,7) 30 _{hex} |
| Load Upper Imm. | lui I | $R[rt] = \{\text{imm}, 16'b0\}$ | f _{hex} |
| Load Word | lw I | $R[rt] = M[R[rs] + \text{SignExtImm}]$ | (2) 23 _{hex} |
| Nor | nor R | $R[rd] = \sim (R[rs] R[rt])$ | 0 / 27 _{hex} |
| Or | or R | $R[rd] = R[rs] R[rt]$ | 0 / 25 _{hex} |
| Or Immediate | ori I | $R[rt] = R[rs] \text{ZeroExtImm}$ | (3) d _{hex} |
| Set Less Than | slt R | $R[rd] = (R[rs] < R[rt]) ? 1 : 0$ | 0 / 2a _{hex} |
| Set Less Than Imm. | slti I | $R[rt] = (R[rs] < \text{SignExtImm}) ? 1 : 0$ | (2) a _{hex} |
| Set Less Than Imm. Unsigned | sltiu I | $R[rt] = (R[rs] < \text{SignExtImm}) ? 1 : 0$ | (2,6) b _{hex} |
| Set Less Than Unsig. | sltu R | $R[rd] = (R[rs] < R[rt]) ? 1 : 0$ | (6) 0 / 2b _{hex} |
| Shift Left Logical | sll R | $R[rd] = R[rt] \ll \text{shamt}$ | 0 / 00 _{hex} |
| Shift Right Logical | srl R | $R[rd] = R[rt] \gg \text{shamt}$ | 0 / 02 _{hex} |
| Store Byte | sb I | $M[R[rs] + \text{SignExtImm}](7:0) = R[rt](7:0)$ | (2) 28 _{hex} |
| Store Conditional | sc I | $M[R[rs] + \text{SignExtImm}] = R[rt];$ $R[rt] = (\text{atomic}) ? 1 : 0$ | (2,7) 38 _{hex} |
| Store Halfword | sh I | $M[R[rs] + \text{SignExtImm}](15:0) = R[rt](15:0)$ | (2) 29 _{hex} |
| Store Word | sw I | $M[R[rs] + \text{SignExtImm}] = R[rt]$ | (2) 2b _{hex} |
| Subtract | sub R | $R[rd] = R[rs] - R[rt]$ | (1) 0 / 22 _{hex} |
| Subtract Unsigned | subu R | $R[rd] = R[rs] - R[rt]$ | 0 / 23 _{hex} |

- (1) May cause overflow exception
- (2) $\text{SignExtImm} = \{16\{\text{immediate}[15]\}, \text{immediate}\}$
- (3) $\text{ZeroExtImm} = \{16\{1'b'0\}, \text{immediate}\}$
- (4) $\text{BranchAddr} = \{14\{\text{immediate}[15]\}, \text{immediate}, 2'b'0\}$
- (5) $\text{JumpAddr} = \{PC + 4[31:28], \text{address}, 2'b'0\}$
- (6) Operands considered unsigned numbers (vs. 2's comp.)
- (7) Atomic test&set pair; $R[rt] = 1$ if pair atomic, 0 if not atomic

BASIC INSTRUCTION FORMATS

| R | opcode | | rs | rt | | rd | | shamt | funct | |
|---|--------|-------|---------|-------|--|-----------|---|-------|-------|---|
| | 31 | 26 25 | | 21 20 | | 16 15 | | 11 10 | 6 5 | 0 |
| I | opcode | | rs | rt | | immediate | | | | |
| | 31 | 26 25 | | 21 20 | | 16 15 | 0 | | | |
| J | opcode | | address | | | | | | | |
| | 31 | 26 25 | 0 | | | | | | | |

ARITHMETIC CORE INSTRUCTION SET

| NAME, MNEMONIC | FOR-MAT | OPERATION | OPCODE / FUNCT (Hex) |
|--|-----------|---|----------------------|
| Branch On FP True | bc1t FI | if(FPcond) $PC = PC + 4 + \text{BranchAddr}$ | (4) 11/8/1/-- |
| Branch On FP False | bc1f FI | if(! FPcond) $PC = PC + 4 + \text{BranchAddr}$ | (4) 11/8/0/-- |
| Divide | div R | $Lo = R[rs]/R[rt]; Hi = R[rs]\%R[rt]$ | 0/--/--/1a |
| Divide Unsigned | divu R | $Lo = R[rs]/R[rt]; Hi = R[rs]\%R[rt]$ | (6) 0/--/--/1b |
| FP Add Single | add.s FR | $F[fd] = F[fs] + F[ft]$ | 11/10/--/0 |
| FP Add Double | add.d FR | $\{F[fd], F[fd+1]\} = \{F[fs], F[fs+1]\} + \{F[ft], F[ft+1]\}$ | 11/11/--/0 |
| FP Compare Single | c.x.s* FR | $\text{FPcond} = (F[fs] \text{ op } F[ft]) ? 1 : 0$ | 11/10/--/y |
| FP Compare Double | c.x.d* FR | $\text{FPcond} = (\{F[fs], F[fs+1]\} \text{ op } \{F[ft], F[ft+1]\}) ? 1 : 0$ | 11/11/--/y |
| * (x is eq, lt, or le) (op is ==, <, or <=) (y is 32, 3c, or 3e) | | | |
| FP Divide Single | div.s FR | $F[fd] = F[fs] / F[ft]$ | 11/10/--/3 |
| FP Divide Double | div.d FR | $\{F[fd], F[fd+1]\} = \{F[fs], F[fs+1]\} / \{F[ft], F[ft+1]\}$ | 11/11/--/3 |
| FP Multiply Single | mul.s FR | $F[fd] = F[fs] * F[ft]$ | 11/10/--/2 |
| FP Multiply Double | mul.d FR | $\{F[fd], F[fd+1]\} = \{F[fs], F[fs+1]\} * \{F[ft], F[ft+1]\}$ | 11/11/--/2 |
| FP Subtract Single | sub.s FR | $F[fd] = F[fs] - F[ft]$ | 11/10/--/1 |
| FP Subtract Double | sub.d FR | $\{F[fd], F[fd+1]\} = \{F[fs], F[fs+1]\} - \{F[ft], F[ft+1]\}$ | 11/11/--/1 |
| Load FP Single | lwc1 I | $F[rt] = M[R[rs] + \text{SignExtImm}]$ | (2) 31/--/--/0 |
| Load FP Double | ldc1 I | $F[rt] = M[R[rs] + \text{SignExtImm}];$ $F[rt+1] = M[R[rs] + \text{SignExtImm} + 4]$ | (2) 35/--/--/0 |
| Move From Hi | mfhi R | $R[rd] = Hi$ | 0/--/--/10 |
| Move From Lo | mfl0 R | $R[rd] = Lo$ | 0/--/--/12 |
| Move From Control | mfc0 R | $R[rd] = CR[rs]$ | 10/00/--/0 |
| Multiply | mult R | $\{Hi, Lo\} = R[rs] * R[rt]$ | 0/--/--/18 |
| Multiply Unsigned | multu R | $\{Hi, Lo\} = R[rs] * R[rt]$ | (6) 0/--/--/19 |
| Shift Right Arith. | sra R | $R[rd] = R[rt] \gg \text{shamt}$ | 0/--/--/3 |
| Store FP Single | swc1 I | $M[R[rs] + \text{SignExtImm}] = F[rt]$ | (2) 39/--/--/0 |
| Store FP Double | sdc1 I | $M[R[rs] + \text{SignExtImm}] = F[rt];$ $M[R[rs] + \text{SignExtImm} + 4] = F[rt+1]$ | (2) 3d/--/--/0 |

FLOATING-POINT INSTRUCTION FORMATS

| FR | opcode | fmt | ft | fs | fd | funct |
|----|--------|-------|-------|-----------|-------|-------|
| | 31 | 26 25 | 21 20 | 16 15 | 11 10 | 6 5 |
| FI | opcode | fmt | ft | immediate | | |
| | 31 | 26 25 | 21 20 | 16 15 | | |

PSEUDOINSTRUCTION SET

| NAME | MNEMONIC | OPERATION |
|------------------------------|----------|--|
| Branch Less Than | b1t | if($R[rs] < R[rt]$) $PC = \text{Label}$ |
| Branch Greater Than | bgt | if($R[rs] > R[rt]$) $PC = \text{Label}$ |
| Branch Less Than or Equal | b1e | if($R[rs] \leq R[rt]$) $PC = \text{Label}$ |
| Branch Greater Than or Equal | bge | if($R[rs] \geq R[rt]$) $PC = \text{Label}$ |
| Load Immediate | li | $R[rd] = \text{immediate}$ |
| Move | move | $R[rd] = R[rs]$ |

REGISTER NAME, NUMBER, USE, CALL CONVENTION

| NAME | NUMBER | USE | PRESERVED ACROSS A CALL? |
|-----------|--------|---|--------------------------|
| \$zero | 0 | The Constant Value 0 | N.A. |
| \$at | 1 | Assembler Temporary | No |
| \$v0-\$v1 | 2-3 | Values for Function Results and Expression Evaluation | No |
| \$a0-\$a3 | 4-7 | Arguments | No |
| \$t0-\$t7 | 8-15 | Temporaries | No |
| \$s0-\$s7 | 16-23 | Saved Temporaries | Yes |
| \$t8-\$t9 | 24-25 | Temporaries | No |
| \$k0-\$k1 | 26-27 | Reserved for OS Kernel | No |
| \$gp | 28 | Global Pointer | Yes |
| \$sp | 29 | Stack Pointer | Yes |
| \$fp | 30 | Frame Pointer | Yes |
| \$ra | 31 | Return Address | Yes |

OPCODES, BASE CONVERSION, ASCII SYMBOLS

| MIPS opcode (31:26) | (1) MIPS funct (5:0) | (2) MIPS funct (5:0) | Binary | Decimal | Hexadecimal | ASCII Character | Decimal | Hexadecimal | ASCII Character |
|------------------------|----------------------------|----------------------------|---------|---------|-------------|-----------------|---------|-------------|-----------------|
| (1) | sll | add _f | 00 0000 | 0 | 0 | NUL | 64 | 40 | @ |
| | | sub _f | 00 0001 | 1 | 1 | SOH | 65 | 41 | A |
| j | srl | mul _f | 00 0010 | 2 | 2 | STX | 66 | 42 | B |
| jal | sra | div _f | 00 0011 | 3 | 3 | ETX | 67 | 43 | C |
| beq | sllv | sqrt _f | 00 0100 | 4 | 4 | EOT | 68 | 44 | D |
| bne | | abs _f | 00 0101 | 5 | 5 | ENQ | 69 | 45 | E |
| blez | srlv | mov _f | 00 0110 | 6 | 6 | ACK | 70 | 46 | F |
| bgtz | srav | neg _f | 00 0111 | 7 | 7 | BEL | 71 | 47 | G |
| addi | jr | | 00 1000 | 8 | 8 | BS | 72 | 48 | H |
| addiu | jalr | | 00 1001 | 9 | 9 | HT | 73 | 49 | I |
| slti | movz | | 00 1010 | 10 | a | LF | 74 | 4a | J |
| sltiu | movn | | 00 1011 | 11 | b | VT | 75 | 4b | K |
| andi | syscall | round.w _f | 00 1100 | 12 | c | FF | 76 | 4c | L |
| ori | break | trunc.w _f | 00 1101 | 13 | d | CR | 77 | 4d | M |
| xori | | ceil.w _f | 00 1110 | 14 | e | SO | 78 | 4e | N |
| lui | sync | floor.w _f | 00 1111 | 15 | f | SI | 79 | 4f | O |
| (2) | mfhi | | 01 0000 | 16 | 10 | DLE | 80 | 50 | P |
| | mthi | | 01 0001 | 17 | 11 | DC1 | 81 | 51 | Q |
| | mflo | movz _f | 01 0010 | 18 | 12 | DC2 | 82 | 52 | R |
| | mtlo | movn _f | 01 0011 | 19 | 13 | DC3 | 83 | 53 | S |
| | | | 01 0100 | 20 | 14 | DC4 | 84 | 54 | T |
| | | | 01 0101 | 21 | 15 | NAK | 85 | 55 | U |
| | | | 01 0110 | 22 | 16 | SYN | 86 | 56 | V |
| | | | 01 0111 | 23 | 17 | ETB | 87 | 57 | W |
| | mult | | 01 1000 | 24 | 18 | CAN | 88 | 58 | X |
| | multu | | 01 1001 | 25 | 19 | EM | 89 | 59 | Y |
| | div | | 01 1010 | 26 | 1a | SUB | 90 | 5a | Z |
| | divu | | 01 1011 | 27 | 1b | ESC | 91 | 5b | [|
| | | | 01 1100 | 28 | 1c | FS | 92 | 5c | \ |
| | | | 01 1101 | 29 | 1d | GS | 93 | 5d |] |
| | | | 01 1110 | 30 | 1e | RS | 94 | 5e | ^ |
| | | | 01 1111 | 31 | 1f | US | 95 | 5f | _ |
| lb | add | cvt.s _f | 10 0000 | 32 | 20 | Space | 96 | 60 | ` |
| | addu | cvt.d _f | 10 0001 | 33 | 21 | ! | 97 | 61 | a |
| lwl | sub | | 10 0010 | 34 | 22 | " | 98 | 62 | b |
| lw | subu | | 10 0011 | 35 | 23 | # | 99 | 63 | c |
| lbu | and | cvt.w _f | 10 0100 | 36 | 24 | \$ | 100 | 64 | d |
| lhu | or | | 10 0101 | 37 | 25 | % | 101 | 65 | e |
| lwr | xor | | 10 0110 | 38 | 26 | & | 102 | 66 | f |
| | nor | | 10 0111 | 39 | 27 | ' | 103 | 67 | g |
| sb | | | 10 1000 | 40 | 28 | (| 104 | 68 | h |
| sh | | | 10 1001 | 41 | 29 |) | 105 | 69 | i |
| swl | slt | | 10 1010 | 42 | 2a | * | 106 | 6a | j |
| sw | sltu | | 10 1011 | 43 | 2b | + | 107 | 6b | k |
| | | | 10 1100 | 44 | 2c | , | 108 | 6c | l |
| | | | 10 1101 | 45 | 2d | - | 109 | 6d | m |
| | | | 10 1110 | 46 | 2e | . | 110 | 6e | n |
| | | | 10 1111 | 47 | 2f | / | 111 | 6f | o |
| swr | | | 11 0000 | 48 | 30 | 0 | 112 | 70 | p |
| cache | | | 11 0001 | 49 | 31 | 1 | 113 | 71 | q |
| l1 | tge | c.f _f | 11 0010 | 50 | 32 | 2 | 114 | 72 | r |
| lwc1 | tgeu | c.un _f | 11 0011 | 51 | 33 | 3 | 115 | 73 | s |
| lwc2 | tlbt | c.eq _f | 11 0100 | 52 | 34 | 4 | 116 | 74 | t |
| pref | tlbtu | c.ueq _f | 11 0101 | 53 | 35 | 5 | 117 | 75 | u |
| | teq | c.lt _f | 11 0110 | 54 | 36 | 6 | 118 | 76 | v |
| ldc1 | | c.ult _f | 11 0111 | 55 | 37 | 7 | 119 | 77 | w |
| ldc2 | tne | c.ole _f | 11 1000 | 56 | 38 | 8 | 120 | 78 | x |
| | | c.cle _f | 11 1001 | 57 | 39 | 9 | 121 | 79 | y |
| sc | | c.ngl _f | 11 1010 | 58 | 3a | : | 122 | 7a | z |
| swc1 | | c.ngl _f | 11 1011 | 59 | 3b | ; | 123 | 7b | { |
| swc2 | | c.lt _f | 11 1100 | 60 | 3c | < | 124 | 7c | } |
| | | c.nge _f | 11 1101 | 61 | 3d | = | 125 | 7d | ~ |
| sdcl | | c.le _f | 11 1110 | 62 | 3e | > | 126 | 7e | ~ |
| sdcl | | c.ngt _f | 11 1111 | 63 | 3f | ? | 127 | 7f | DEL |

(1) opcode(31:26) == 0

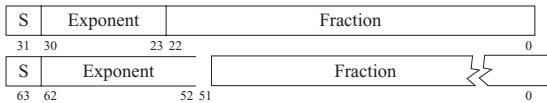
(2) opcode(31:26) == 17_{ten} (11_{hex}); if fmt(25:21) == 16_{ten} (10_{hex}) f = s (single);
if fmt(25:21) == 17_{ten} (11_{hex}) f = d (double)

IEEE 754 FLOATING-POINT STANDARD

$$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

where Single Precision Bias = 127,
Double Precision Bias = 1023.

IEEE Single Precision and Double Precision Formats:

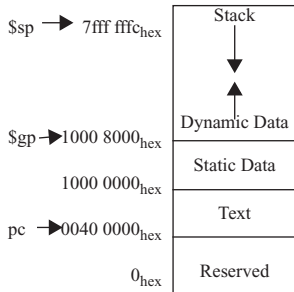


IEEE 754 Symbols

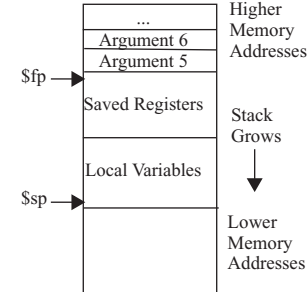
| Exponent | Fraction | Object |
|--------------|----------|----------------|
| 0 | 0 | ± 0 |
| 0 | ≠ 0 | ± Denorm |
| 1 to MAX - 1 | anything | ± Fl. Pt. Num. |
| MAX | 0 | ± ∞ |
| MAX | ≠ 0 | NaN |

S.P. MAX = 255, D.P. MAX = 2047

MEMORY ALLOCATION



STACK FRAME

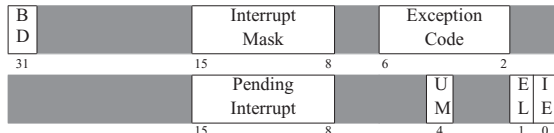


DATA ALIGNMENT

| Double Word | | | | | | | |
|-------------|------|----------|------|----------|------|----------|------|
| Word | | | | Word | | | |
| Halfword | | Halfword | | Halfword | | Halfword | |
| Byte | Byte | Byte | Byte | Byte | Byte | Byte | Byte |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Value of three least significant bits of byte address (Big Endian)

EXCEPTION CONTROL REGISTERS: CAUSE AND STATUS



BD = Branch Delay, UM = User Mode, EL = Exception Level, IE = Interrupt Enable

EXCEPTION CODES

| Number | Name | Cause of Exception | Number | Name | Cause of Exception |
|--------|------|---|--------|------|-------------------------------------|
| 0 | Int | Interrupt (hardware) | 9 | Bp | Breakpoint Exception |
| 4 | AdEL | Address Error Exception (load or instruction fetch) | 10 | RI | Reserved Instruction Exception |
| 5 | AdES | Address Error Exception (store) | 11 | CpU | Coprocessor Unimplemented Exception |
| 6 | IBE | Bus Error on Instruction Fetch | 12 | Ov | Arithmetic Overflow Exception |
| 7 | DBE | Bus Error on Load or Store | 13 | Tr | Trap |
| 8 | Sys | Syscall Exception | 15 | FPE | Floating Point Exception |

SIZE PREFIXES (10^x for Disk, Communication; 2^x for Memory)

| SIZE | PRE-FIX | SIZE | PRE-FIX | SIZE | PRE-FIX | SIZE | PRE-FIX |
|------------------------------------|---------|------------------------------------|---------|-------------------|---------|-------------------|---------|
| 10 ³ , 2 ¹⁰ | Kilo- | 10 ¹⁵ , 2 ⁵⁰ | Peta- | 10 ⁻³ | milli- | 10 ⁻¹⁵ | femto- |
| 10 ⁶ , 2 ²⁰ | Mega- | 10 ¹⁸ , 2 ⁶⁰ | Exa- | 10 ⁻⁶ | micro- | 10 ⁻¹⁸ | atto- |
| 10 ⁹ , 2 ³⁰ | Giga- | 10 ²¹ , 2 ⁷⁰ | Zetta- | 10 ⁻⁹ | nano- | 10 ⁻²¹ | zepto- |
| 10 ¹² , 2 ⁴⁰ | Tera- | 10 ²⁴ , 2 ⁸⁰ | Yotta- | 10 ⁻¹² | pico- | 10 ⁻²⁴ | yocto- |

The symbol for each prefix is just its first letter, except μ is used for micro.