<http://www.redgamingtech.com/why-ps4-and-xbox-one-moved-to-x86-64/>

MIPS Instruction Fields - R



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | Instruction | Example | Meaning | Comments |
| Arithmetic | add | add $s1, $s2, $s3 | $s1 = $s2 + $s3 | Three register operands |
| subtract | sub $s1, $s2, $s3 | $s1 = $s2 - $s3 | Three register operands |
| add immediate | addi $s1, $s2, 20 | $s1 = $s2 + 20 | Used to add constants |
| Data transfer | load word | lw $s1, 20($s2) | $s1 = Memory[$s2 + 20] | Word from memory to register |
| store word | sw $s1, 20($s2) | Memory[$s2 + 20] = $s1 | Word from register to memory |
| load half | lh $s1, 20($s2) | $s1 = Memory[$s2 + 20] | Halfword memory to register |
| load half unsigned | lhu $s1, 20($s2) | $s1 = Memory[$s2 + 20] | Halfword memory to register |
| store half | sh $s1, 20($s2) | Memory[$s2 + 20] = $s1 | Halfword register to memory |
| load byte | lb $s1, 20($s2) | $s1 = Memory[$s2 + 20] | Byte from memory to register |
| load byte unsigned | lbu $s1, 20($s2) | $s1 = Memory[$s2 + 20] | Byte from memory to register |
| store byte | sb $s1, 20($s2) | Memory[$s2 + 20] = $s1 | Byte from register to memory |
| load linked word | ll $s1, 20($s2) | $s1 = Memory[$s2 + 20] | Load word as 1st half of atomic swap |
| store condition. word | sc $s1, 20($s2) | Memory[$s2 + 20] = $s1; $s1 = 0 or 1 | Store word as 2nd half of atomic swap |
| load upper immed. | lui $s1, 20 | $s1 = 20 \* 216 | Loads constant in upper 16 bits |
| Logical | and | and $s1, $s2, $s3 | $s1 = $s2 & $s3 | Three reg. operands; bit-by-bit AND |
| or | or $s1, $s2, $s3 | $s1 = $s2 | $s3 | Three reg. operands; bit-by-bit OR |
| nor | nor $s1, $s2, $s3 | $s1 = ~($s2 | $s3) | Three reg. operands; bit-by-bit NOR |
| and immediate | andi $s1, $s2, 20 | $s1 = $s2 & 20 | Bit-by-bit AND reg with constant |
| or immediate | ori $s1, $s2, 20 | $s1 = $s2 | 20 | Bit-by-bit OR reg with constant |
| shift left logical | sll $s1, $s2, 10 | $s1 = $s2 << 10 | Shift left by constant |
| shift right logical | srl $s1, $s2, 10 | $s1 = $s2 >> 10 | Shift right by constant |
| Conditional branch | branch on equal | beq $s1, $s2, 25 | if($s1 == $s2) go to  PC + 4 + 100 | Equal test; PC-relative branch |
| branch on not equal | bne $s1, $s2, 25 | if($s1 != $s2) go to  PC + 4 + 100 | Not equal test; PC-relative branch |
| set on less than | slt $s1, $s2, $s3 | if($s2 < $s3) $s1 = 1;  else $s1 = 0 | Compare less than; for beq, bne |
| set on less than unsigned | sltu $s1, $s2, $s3 | if($s2 < $s3) $s1 = 1;  else $s1 = 0 | Compare less than unsigned |
| set less than immediate | slti $s1, $s2, 20 | if($s2 < 20) $s1 = 1;  else $s1 = 0 | Compare less than constant |
| set less than immediate unsigned | sltiu $s1, $s2, 20 | if($s2 < 20) $s1 = 1;  else $s1 = 0 | Compare less than constant unsigned |
| Unconditional jump | jump | j 2500 | go to 10000 | Jump to target address |
| jump register | jr $ra | go to $ra | For switch, procedure return |
| jump and link | jal 2500 | $ra = PC + 4; go to 10000 | For procedure call |