**Basic Instruction Formats**

| | | | |
|---|---|---|---|
| Register Format | `0000 00ss ssst tttt dddd d000 00ff ffff` | R | s, t, d are interpreted as unsigned |
| Immediate Format | `oooo ooss ssst tttt iiii iiii iiii iiii` | I | i is interpreted as two's complement |

**Instructions**

| Name | Assembly Syntax | Machine Syntax | | Semantics/Behaviour |
|---|---|---|---|---|
| Add | add $d, $s, $t | `0000 00ss ssst tttt dddd d000 0010 0000` | R | `$d = $s + $t` |
| Subtract | sub $d, $s, $t | `0000 00ss ssst tttt dddd d000 0010 0010` | R | `$d = $s - $t` |
| Multiply | mult $s, $t | `0000 00ss ssst tttt 0000 0000 0001 1000` | R | `hi:lo = $s * $t` |
| Multiply Unsigned | multu $s, $t | `0000 00ss ssst tttt 0000 0000 0001 1001` | R | `hi:lo = $s * $t` |
| Divide | div $s, $t | `0000 00ss ssst tttt 0000 0000 0001 1010` | R | `lo = $s / $t; hi = $s % $t` |
| Divide Unsigned | divu $s, $t | `0000 00ss ssst tttt 0000 0000 0001 1011` | R | `lo = $s / $t; hi = $s % $t` |
| Move From High/Remainder | mfhi $d | `0000 0000 0000 0000 dddd d000 0001 0000` | R | `$d = hi` |
| Move From Low/Quotient | mflo $d | `0000 0000 0000 0000 dddd d000 0001 0010` | R | `$d = lo` |
| Load Immediate and Skip | lis $d | `0000 0000 0000 0000 dddd d000 0001 0100` | R | `$d = MEM[pc]; pc = pc + 4` |
| Set Less Than | slt $d, $s, $t | `0000 00ss ssst tttt dddd d000 0010 1010` | R | `$d = 1 if $s < $t; 0 otherwise` |
| Set Less Than Unsigned | sltu $d, $s, $t | `0000 00ss ssst tttt dddd d000 0010 1011` | R | `$d = 1 if $s < $t; 0 otherwise` |
| Jump Register | jr $s | `0000 00ss sss0 0000 0000 0000 0000 1000` | R | `pc = $s` |
| Jump and Link Register | jalr $s | `0000 00ss sss0 0000 0000 0000 0000 1001` | R | `temp = $s; $31 = pc; pc = temp` |
| Branch On Equal | beq $s, $t, i | `0001 00ss ssst tttt iiii iiii iiii iiii` | I | `if ($s == $t) pc += i * 4` |
| Branch On Not Equal | bne $s, $t, i | `0001 01ss ssst tttt iiii iiii iiii iiii` | I | `if ($s != $t) pc += i * 4` |
| Load Word | lw $t, i($s) | `1000 11ss ssst tttt iiii iiii iiii iiii` | I | `$t = MEM [$s + i]` |
| Store Word | sw $t, i($s) | `1010 11ss ssst tttt iiii iiii iiii iiii` | I | `MEM [$s + i] = $t` |

**Directives**

| Name | Assembly Syntax | Machine Syntax | | Semantics/Behaviour |
|---|---|---|---|---|
| Encode As Word | .word i | `iiii iiii iiii iiii iiii iiii iiii iiii` | | `i is encoded as a 32-bit word` |

**Input and Output:** When a word is stored to memory location `0xffff000c`, the least significant byte of the word is sent to standard output. Loading a word from memory address `0xffff0004` places the next byte from standard input into the least significant byte of the destination register.