|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction** | **Format** | **Syntax** | **Operation** | **Opcode** | |
| **Dec** | **Hex** |
| Add | R | add $d, $s, $t | $d = $s + $t | 100000 | 20 |
| Add immediate | I | addi $t, $s, i | $t = $s + i | 001000 | 8 |
| And | R | and $d, $s, $t | $d = $s & $t | 100100 | 24 |
| And immediate | I | andi $t, $s, i | $t = $s & i | 001100 | c |
| Or | R | or $d, $s, $t | $d = $s | $t | 100101 | 25 |
| Ori | I | ori $t, $s, i | $t = $s | i | 001101 | d |
| Shift left logical | R | sll $d, $t, a | $d = $t \* 2a | 000000 | 00 |
| Shift right logical | R | Srl $d, $t, a | $d = $t / 2a | 000010 | 2 |
| Subtract | R | Sub $d, $s, $t | $d = $s - $t | 100010 | 22 |
| Branch on equal | I | Beq $s, $t, label | If ($s == $t) {label} | 000100 | 4 |
| Branch on not equal | I | Bne $s, $t, label | If ($s != $t) {label} | 000101 | 5 |
| Branch on greater than | I | Bgt $s, $t, label | If ($s > $t) {label} |  | |
| Branch on greater or equal | I | Bge $s, $t, label | If ($s >= $t) {label} |  | |
| Branch on less than | I | Blt $s, $t, label | If ($s < $t) {label} |  | |
| Branch on less or equal | I | Ble $s, $t, label | If ($s <= $t) {label} |  | |
| Jump | J | J label | Go to label | 000010 | 2 |
| Load word | I | lw $t, i($s) | $t = MEM [$s + i]  \*i is a mult of 4 | 100011 | 23 |
| Store word | I | sw $t, i($s) | MEM [$s + i] = $t  \*i is a mult of 4 | 101011 | 2b |

**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 | | | | |

**Bitwise**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **And** | | | | | | | | **Or** | | | | | | | |  |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |  | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |  | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |  | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |

**Register name and number use**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Register name** | **Dec number** | **Bin number** |  | **Register name** | **Dec number** | **Bin number** |
| $zero | 0 | 00000 |  | $t5 | 13 | 01101 |
| $at | 1 | 00001 |  | $t6 | 14 | 01110 |
| $v0 | 2 | 00010 |  | $t7 | 15 | 01111 |
| $v1 | 3 | 00011 |  | $s0 | 16 | 10000 |
| $a0 | 4 | 00100 |  | $s1 | 17 | 10001 |
| $a1 | 5 | 00101 |  | $s2 | 18 | 10010 |
| $a2 | 6 | 00110 |  | $s3 | 19 | 10011 |
| $a3 | 7 | 00111 |  | $s4 | 20 | 10100 |
| $t0 | 8 | 01000 |  | $s5 | 21 | 10101 |
| $t1 | 9 | 01001 |  | $s6 | 22 | 10110 |
| $t2 | 10 | 01010 |  | $s7 | 23 | 10111 |
| $t3 | 11 | 01011 |  | $t8 | 24 | 11000 |
| $t4 | 12 | 01100 |  | $t9 | 25 | 11001 |
| $t5 | 13 | 01101 |  |  |  |  |

**Conversions**

|  |  |
| --- | --- |
| **Binary (101110)** | |
| **Hexadecimal** | **Decimal** |
| |  |  |  | | --- | --- | --- | | Binary | 0010 | 1110 | | Decimal | 2 | 14 | | Hex | 2E | | | (25 \* 1) + ( 24 \* 0) + ( 23 \* 1) + ( 22 \* 1) + ( 21 \* 1) + ( 20 \* 0) = 46 |
| **Hexadecimal (2E)** | |
| **Binary** | **Decimal** |
| |  |  |  | | --- | --- | --- | | Hex | 2 | E | | Decimal | 2 | 14 | | Bin | 0010 | 1110 | | (161 \* 2) + ( 160 \* 14) = 46 |
| **Decimal (46)** | |
| **Hexadecimal** | **Binary** |
| |  |  |  | | --- | --- | --- | | Binary | 0010 | 1110 | | Decimal | 2 | 14 | | Hex | 2E | | | |  |  | | --- | --- | | 46 | 0 | | 23 | 1 | | 11 | 1 | | 5 | 1 | | 2 | 0 | | 1 | 1 |   101110 |

**2’s complement**

Given: FFFFFFAB = 1111 1111 1111 1111 1111 1111 1010 1011

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1111** | **1111** | **1111** | **1111** | **1111** | **1111** | **1010** | **1011** | Given |
|  | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0101 | 0100 | Switch 1 -> 0 and 0 -> 1 |
| + |  |  |  |  |  |  |  | 1 | Add 1 |
|  | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0101 | 0101 | Final answer in binary |
|  |  |  |  |  |  |  |  | = 85 | Convert |

**Examples MIPs <=> high level language <=> machine code**

|  |
| --- |
| for (i = 0; i < 100; ++i)  {  sum += A[i];  } |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | &A[0]: $s0 sum: $v0 $t0 = 1 | | | | | | |  | and | $t0, | $t0, | $zero | # let i = 0 | |  | addi | $t1, | $zero, | 100 | # intermediate 100 | | Loop: | lw | $t3, | 0($s0) |  | # temp1 = A[i] | |  | add | $v0, | $v0, | $t3 | # sum += temp1 | |  | addi | $s0, | $s0, | 4 | # address of A[i + 1] | |  | addi | $t0 | $t0 | 1 | #i = i + 1 | |  | bne | $t1, | $t0, | Loop | # if i < 100 | | Exit: |  |  |  |  |  | |

|  |
| --- |
| for (i = 0; i < 100; ++i)  {  A[i] = i;  } |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | and | $t0, | $zero, | $zero | |  | addi | $t1, | $zero, | 100 | | Loop: | sw | $t0, | 0($s2) |  | |  | addi | $s2, | $s2, | 4 | |  | addi | $t0 | $t0 | 1 | |  | beq | $t0, | $t1, | Loop | | Exit: |  |  |  |  | |