|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **add** | Add | 0 0 0 0 | Rd | Rs1 | Rs2 | 0 0 0 |  |
| Two's complement addition. Overflow is not detected.  R(Rd) ← R(Rs1) + R(Rs2) | | | | | | |  |
| **sub** | Subtract | 0 0 0 0 | Rd | Rs1 | Rs2 | 0 0 1 |  |
| Two's complement subtraction. Overflow is not detected.  R(Rd) ← R(Rs1) - R(Rs2) | | | | | | |  |
| **and** | Bitwise and | 0 0 0 0 | Rd | Rs1 | Rs2 | 0 1 0 |  |
| Bitwise and operation.  R(Rd) ← R(Rs1) & R(Rs2) | | | | | | |  |
| **or** | Bitwise or | 0 0 0 0 | Rd | Rs1 | Rs2 | 0 1 1 |  |
| Bitwise or operation.  R(Rd) ← R(Rs1) | R(Rs2) | | | | | | |  |
| **not** | Bitwise not | 0 0 0 0 | Rd | Rs1 | unused | 1 0 0 |  |
| Bitwise not operation.  R(Rd) ← ~R(Rs1) | | | | | | |  |
| **jalr** | Jump and link register | 0 0 0 1 | Rd | Rs1 | unused | unused |  |
| Jumps to the address stored in register Rd and stores PC + 1 in register Rs1. It is used for subroutine calls. It can also be used for normal jumps by using register r0 as Rs1.  R(Rs1) ← PC + 1 PC ← R(Rd) | | | | | | |  |
| **in** | Get word from input | 0 0 1 0 | Rd | unused | unused | unused |  |
| Get a word from user input.  R(Rd) ← input | | | | | | |  |
| **out** | Send word to output | 0 0 1 1 | Rd | unused | unused | unused |  |
| Send a word to output. If Rd is r0, then the processor is halted.  output ← R(Rd) | | | | | | |  |
| **addi** | Add immediate | 0 1 0 0 | Rd | Rs1 | Imm6 | |  |
| Two's complement addition with a signed immediate. Overflow is not detected.  R(Rd) ← R(Rs1) + Imm6 | | | | | | |  |
| **shf** | Bit shift | 0 1 0 1 | Rd | Rs1 | Imm6 | |  |
| Bit shift. It is either left if Imm6 is positive or right if the contents are negative. The right shift is a logical shift with zero extension.  if (Imm6 > 0) R(Rd) ← R(Rs1) << Imm6 else R(Rd) ← R(Rs1) >> Imm6 | | | | | | |  |
| **lw** | Load word from memory | 0 1 1 0 | Rd | Rs1 | Imm6 | |  |
| Loads word from memory using the effective address computed by adding Rs1 with the signed immediate.  R(Rd) ← M[R(Rs1) + Imm6] | | | | | | |  |
| **sw** | Store word to memory | 0 1 1 1 | Rd | Rs1 | Imm6 | |  |
| Stores word into memory using the effective address computed by adding Rs1 with the signed immediate.  M[R(Rs1) + Imm6] ← R(Rd) | | | | | | |  |
| **lli** | Load lower immediate | 1 0 0 0 | Rd | unused | Imm8 | | |
| The lower bits (7-0) of Rd are copied from the immediate. The upper bits (15- 8) of Rd are set to bit 7 of the immediate to produce a sign-extended result.  R(Rd[15..8]) ← Imm8[7] R(Rd[7..0]) ← Imm8 | | | | | | |  |
| **lui** | Load upper immediate | 1 0 0 1 | Rd | unused | Imm8 | | |
| The upper bits (15- 8) of Rd are copied from the immediate. The lower bits (7-0) of Rd are unchanged. The sign of the immediate does not matter – the eight bits are copied directly.  R(Rd[15..8]) ← Imm8 | | | | | | |  |
| **beq** | Branch if equal to zero | 1 0 1 0 | Rd | unused | Imm8 | | |
| Conditional branch – compares Rd to zero. If R(Rd) = 0, then branch is taken with indirect target of PC + 1 + Imm8 as next PC. Immediate is a signed value.  if (R(Rd) == 0) PC ← PC + 1 + Imm8 | | | | | | |  |
| **bne** | Branch if not equal to zero | 1 0 1 0 | Rd | unused | Imm8 | | |
| Conditional branch – compares Rd to zero. If R(Rd) ≠ 0, then branch is taken with indirect target of PC + 1 + Imm8 as next PC. Immediate is a signed value.  if (R(Rd) ≠ 0) PC ← PC + 1 + Imm8 | | | | | | |  |
| **bgt** | Branch if greater than zero | 1 1 0 0 | Rd | unused | Imm8 | | |
| Conditional branch – compares Rd to zero. If R(Rd) > 0, then branch is taken with indirect target of PC + 1 + Imm8 as next PC. Immediate is a signed value.  if (R(Rd) > 0) PC ← PC + 1 + Imm8 | | | | | | |  |
| **bge** | Branch if greater than or equal to zero | 1 1 0 1 | Rd | unused | Imm8 | | |
| Conditional branch – compares Rd to zero. If R(Rd) ≥ 0, then branch is taken with indirect target of PC + 1 + Imm8 as next PC. Immediate is a signed value.  if (R(Rd) ≥ 0) PC ← PC + 1 + Imm8 | | | | | | |  |
| **blt** | Branch if less than to zero | 1 1 1 0 | Rd | unused | Imm8 | | |
| Conditional branch – compares Rd to zero. If R(Rd) < 0, then branch is taken with indirect target of PC + 1 + Imm8 as next PC. Immediate is a signed value.  if (R(Rd) < 0) PC ← PC + 1 + Imm8 | | | | | | |  |
| **ble** | Branch if less than or equal to zero | 1 1 1 1 | Rd | unused | Imm8 | | |
| Conditional branch – compares Rd to zero. If R(Rd) ≤ 0, then branch is taken with indirect target of PC + 1 + Imm8 as next PC. Immediate is a signed value.  if (R(Rd) ≤ 0) PC ← PC + 1 + Imm8 | | | | | | |  |