## Core Instruction Formats

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Opcode** | **Function** | **English Description** |
| LW | I-Type | 0x0 | Reggie = Mem[SE(imm)] | Load mem[addr] into Reggie |
| SW | I-Type | 0x1 | Mem[SE(imm)] = Reggie | Store the value of Reggie into mem[imm] |
| LMEM | I-Type | 0x2 | Reggie = Mem[Mem[SE(imm)]] | Load MEM[MEM[addr]] into Reggie |
| SMEM | I-Type | 0x3 | MEM[MEM[SE(imm)]] = Reggie | Store the value of Reggie into MEM[MEM[addr]] |
| ADD | I-Type | 0x4 | Reggie = Reggie + Mem[SE(imm)] | Increment Reggie by mem[imm] |
| SUB | I-Type | 0x5 | Reggie = Reggie -  Mem[SE(imm)] | Decrement Reggie by mem[imm] |
| ADDI | I-Type | 0x6 | Reggie = Reggie + SE(imm) | Increment Reggie by the immediate |
| CMPI | I-Type | 0x7 | If Reggie < SE(imm): sr[1:0] = **01**  If Reggie = SE(imm): sr[1:0] = **10**  If Reggie > SE(imm): sr[1:0] = **11** | Compare Reggie and the immediate and update sr 1:0 accordingly |
| CMP | I-Type | 0x8 | If Reggie < Mem[SE(imm)] : sr[1:0] = **01**  If Reggie = Mem[SE(imm)] : sr[1:0] = **10**  If Reggie > Mem[SE(imm)] : sr[1:0] = **11** | Compare Reggie and the mem[imm] and update sr 3:0 accordingly |
| B | B-Type | 0x9 | If comp\_code == sr[1:0]  PC = PC + destination address  Comp\_code 00 == unconditional branch | Branch to dest\_addr if comp\_code equals SR[1:0] |
| BMEM | B-Type | 0xA | If comp\_code == sr[1:0]  PC = PC + destination address  Comp\_code 00 == unconditional branch | Branch to location stored at address of immediate. |
| LLI | I-Type | 0xB | Reggie[11:0] = SE(imm) | Load the immediate into the most significant bits of reggie. The remaining upper bits are 0. |
| SLLI | I-Type | 0xC | Reggie = Reggie << SE(imm)[2:0] | Logical Left shift Reggie by the immediate |
| LI | I-Type | 0xE | input = Reggie | Put the value of input into Reggie. Any immediate provided will be ignored. |
| SO | I-Type | 0xF | output = Reggie | Put the value of Reggie into output |

## Core Instruction Formats

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 15:14 | 13:12 | 11:8 | 7:4 | 3:0 |
| **I-Type** | Immediate | | | | Opcode |
| **B-Type** | Comparison Code | Memory Immediate | | | Opcode |

**Note:** the higher immediate index, the more significant the bit. Refer to LUI and LLI to see how most and least significant bits are read

## Memory Allocation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | Stack | Stack Pointer | Return Address | Text |
| **Bits** | 0xFFFF:0x0400 | 0x03FFF:0x03FFE | 0x03FFD:0x03FFC | 0x3FFB:0x0000 |

## Special Memory Locations

|  |  |
| --- | --- |
| **Address** | **Name** |
| RA | 0x03FD |
| SP | 0x03FF |
| A0 – A6 | 0x0404 – 0x040E |
| S0 – S7 | 0x0410 – 0x041E |
| T0 – T15 | 0x0420 – 0x043E |