**Document the instruction formats:**

We will use 3 types of instruction formats for simplicity: **R-type**, **I-type**, and **J-type**.

**R-type Format (for arithmetic operations like ADD, SUB):**

This format is used for instructions that perform operations between registers.

| Opcode (6 bits) | RegA (5 bits) | RegB (5 bits) | RegC (5 bits) | Unused (11 bits) |

* **Opcode**: 6 bits, defines the operation (e.g., ADD, SUB).
* **RegA, RegB, RegC**: 5 bits each, register identifiers.
* **Unused**: 11 bits, padding for alignment or unused space.

#### ****I-type Format (for LOAD, STORE, MOV, CMP)****:

This format is used for operations that involve an immediate value or memory addresses.

| Opcode (6 bits) | RegA (5 bits) | RegB (5 bits) | Immediate (16 bits) |

* **Opcode**: 6 bits, operation identifier (LOAD, STORE, etc.).
* **RegA, RegB**: 5 bits each, register identifiers.
* **Immediate**: 16 bits, a constant value or memory address.

**J-type Format (for JMP, BEQ):**

This format is used for jump and branch operations.

| Opcode (6 bits) | Address (26 bits) |

* **Opcode**: 6 bits, jump or branch operation identifier.
* **Address**: 26 bits, target memory address or offset.

### **Instruction Set Definition**

Here is a table that defines the opcode and instruction formats:

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | Opcode (6bits) | Format | Description |
| ADD | 000000 | R-type | Add two registers: R0 = R1 + R2 |
| SUB | 000001 | R-type | Subtract two registers: R0 = R1 - R2 |
| LOAD | 000010 | I-type | Load from memory into a register |
| STORE | 000011 | I-type | Store from a register into memory |
| MOV | 000100 | I-type | Move data from one register to another |
| JMP | 000101 | J-type | Unconditional jump to address |
| BEQ | 000110 | J-type | Branch if equal |