**Project Proposal**

In this project, we choose to work with Register-Register based architecture. It is very fast because only registers are involved for computing.

Format: Both R-Format and I-Format will be used.

R-Format: and $r3, $r1, $r2

Here, $r3 = destination (rd), $r1 = source 1(rs), $r2 = source 2(rt)

I-Format: andi $r3, $r1, 2

Here, $r1 = source 1(rs), $r3 = destination (rd), 2 = offset/immediate

In both the format, we allot 3 bits to the opcode. As a result, 8 operations will be used.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Bit Fields | | |  |
|  | 4 bits | 2 bits | 2 bits | 2 bits |
| R-Format | opcode | rd | rs | rt |
| I-Format | opcode | rd | rs | immediate |

The list of Opcode is given below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **List** | **Type** | **Opcode** |  | **Code** |
| Add | R | 0000 | add | add $r3, $r1, $r2 |
| Sub | R | 0001 | sub | sub $r3, $r1, $r2 |
| Load Word | I | 0010 | lw | lw $r3, imm($r1) |
| Store Word | I | 0011 | sw | sw $r3, imm($r1) |
| Bitwise And | R | 0100 | and | and $r3, $r1, $r2 |
| And immediate | I | 0101 | andi | andi $r3, $r1, imm |
| Bitwise Or | R | 0110 | or | or $r3, $r1, $r2 |
| Or immediate | I | 0111 | ori | ori $r2, $r1, imm |