**RISC 10 bit Instruction Set Architecture**

We are designing a 10 bit ISA with a 4 bit opcode and 3 operands. Thus we will have 16 operations. We will be using both R-format and I-format for this ISA. The following provides a detailed description of this ISA.

**Format**

**Bit:**

**10 9 8 7 6 5 4 3 2 1**

**RR-format :**

4 bits 2 bits 2 bits 2 bits

|  |  |  |  |
| --- | --- | --- | --- |
| **opcode** | **Reg A (rs)** | **Reg B (rt)** | **Reg C (rd)** |

**RI-format :**

4 bits 2 bits 2 bits 2 bits

|  |  |  |  |
| --- | --- | --- | --- |
| **opcode** | **Reg A (rs)** | **Reg B (rt)** | **Unsigned Immediate(00 to 11)** |

This is a register-register based ISA meaning there is no memory involved except for lw,sw and lui operations . We used 3 operands each of 2 bits because we needed 2 source registers and one destination register for the R-format and 2 source register and one immediate for the I-format.

**Instructions**

Since we used 4 bit opcode we will have 16 operations. From 0000 to 0111 all the operations are of R-format and from 1000 to 1111 the operations are of I-format.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | **Opcode** | **Name and format** | **Type** | **Action** | **Assembly format** |
| ADD | 0000 | Add  R-format | Arithmetic | $rd = $rs + $rt | add $rd, $rs ,$rt |
| SUB | 0001 | Sub  R-format | Arithmetic | $rd = $rs - $rt | sub $rd, $rs, $rt |
| AND | 0010 | AND  R-format | Logical | $rd = $rs && $rt | and $rd, $rs, $rt |
| OR | 0011 | OR  R-format | Logical | $rd = $rs || $rt | or $rd, $rs, $rt |
| XOR | 0100 | XOR  R-format | Logical | $rd = $rs ⊕ $rt | xor $rd, $rs, $rt |
| CMP | 0101 | CMP  R-format | Arithmetic | $rd = $rs ~ $rt | cmp $rd, $rs, $rt |
| NOR | 0110 | NOR  R-format | Logical | $rd = ($rs || $rt)’ | nor $rd, $rs, $rt |
| NAND | 0111 | NAND  R-format | Logical | $rd = ($rs && $rt)’ | nand $rd, $rs, $rt |
| ADDI | 1000 | ADDI  I-format | Arithmetic | $rd = $rs + immediate value | addi $rd, $rs, immediate |
| SUBI | 1001 | SUBI  I-format | Arithmetic | $rd = $rs + immediate value | subi $rd, $rs, immediate |
| SLL | 1010 | SLL  I-format | Logical shift | $rd = $rs << immediate value | sll $rd, $rs, immediate |
| SRL | 1011 | SRL  I-format | Logical shift | $rd = $rs >> immediate value | srl $rd, $rs, immediate |
| LW | 1100 | LW  I-format | Data Transfer | $rd = MEM[$rs+offset] | lw $rd, offset($rs) |
| SW | 1101 | SW  I-format | Data Transfer | MEM[$rd+offset] = $rs | sw $rs, offset($rd) |
| LUI | 1110 | LUI  I-format | Data Transfer | $rd = MEM[bin] | lui $rd, hex |
| ANDI | 1111 | ANDI  I-format | Logical | $rd = $rs && immediate value | andi $rd, $rs, immediate |

**Registers**

|  |  |  |
| --- | --- | --- |
| **Register Name** | **Register Number** | **Register Code** |
| $r0 | 0 | 00 |
| $r1 | 1 | 01 |
| $r2 | 2 | 10 |
| $r3 | 3 | 11 |