**R-Format:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **opcode** | **rs** | **rt** | **rd** | **empty field** |
| **6 bits** | **5 bits** | **5 bits** | **5 bits** | **11 bits** |

**I–Format:**

|  |  |  |  |
| --- | --- | --- | --- |
| **opcode** | **rs** | **rt** | **address/immediate** |
| **6 bits** | **5 bits** | **5 bits** | **16 bits** |

**J- Format:**

|  |  |
| --- | --- |
| **opcode** | **address** |
| **6 bits** | **26 bits** |

**Possible Instructions to implement**

-NOP

-Factorial

-Multiplication

-Division

-Branch if is a divisor

-Branch if is Multiple

-Exponent

-Modulus

**Instructions that we are definitely going to add**

-nor

-nori

-inv

-xori

-multiply

-divide

-NOP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Mnemonic** | **Operation** | **Opcode** | **Format** |
| **Add** | **add** | **$rd = $rs + $rt; $rd = $rs + $rt** | **000000** | **R** |
| **Subtract** | **sub** | **sub $rd, $rs, $rt; $rd = $rs - $rt** | **000001** | **R** |
| **AND** | **and** | **and $rd, $rs, $rt; $rd = $rs & $rt** | **000010** | **R** |
| **OR** | **or** | **or $rd, $rs, $rt; $rd = $rs | $rt** | **000011** | **R** |
| **AND (imm)** | **andi** | **andi $rt, $rs, imm; $rt = $rs & imm** | **000100** | **I** |
| **OR (imm)** | **ori** | **ori $rt, $rs, imm; $rt = $rs | imm** | **000101** | **I** |
| **Exclusive OR** | **xor** | **xor $rd, $rs, $rt; $rd = $rs ^ $rt** | **000110** | **R** |
| **Shift Left Logical** | **sll** | **sll $rt, $rs, imm; $rt = $rs << imm** | **000111** | **I** |
| **Shift Right Logical** | **srl** | **srl $rt, $rs, imm; $rt = $rs >> imm** | **001000** | **I** |
| **Set on Less Than** | **slt** | **slt $rd, $rs, $rt; if $rs < $rt $rd = 1** | **001001** | **R** |
| **Set on Greater Than** | **sgt** | **sgt $rd, $rs, $rt; if $rs > $rt $rd = 1** | **001010** | **R** |
| **Store Word** | **sw** | **sw $rt, offset($rs); MEM[$rs + offset] = $rt** | **001011** | **I** |
| **Load Word** | **lw** | **lw $rt, offset($rs); $rt = MEM[$rs + offset]** | **001100** | **I** |
| **Load Immediate** | **li** | **li $rt, imm; $rt = imm** | **001101** | **I** |
| **Add Immediate** | **addi** | **addi $rt, $rs, imm; $rt = $rs + imm** | **001110** | **I** |
| **Subtract Immediate** | **subi** | **subi $rt, $rs, imm; $rt = $rs - imm** | **001111** | **I** |
| **Branch on Not Equal** | **bne** | **bne $rs, $rt, offset; if $rs != $rt advance\_pc (offset << 2))** | **010000** | **I** |
| **Branch on Equal** | **beq** | **beq $rs, $rt, offset; if $rs == $rt advance\_pc (offset << 2))** | **010001** | **I** |
| **Branch if Greater Than** | **bgt** | **bgt $rs, $rt, offset; if $rs > $rt advance\_pc (offset << 2))** | **010010** | **I** |
| **Branch if Less Than** | **blt** | **blt $rs, $rt, offset; if $rs < $rt advance\_pc (offset << 2))** | **010011** | **I** |
| **Jump** | **j** | **j target; PC = nPC; nPC = (PC & 0xf0000000) | (target << 2)** | **010100** | **J** |

**Factorial Diagram Suggestion:**

