**Custom Instruction Set Architecture (ISA) Design Report**

**Computer Architecture Project**

**Team Members  
Srujan Patel: 230001063**

**Prayag Lakhani: 230001045**

**1. Introduction**

**This report presents the design of a customized Reduced Instruction Set Architecture (ISA) using a 24-bit instruction format. The goal is to optimize instruction size while retaining essential arithmetic, logic, memory, and control operations. The architecture supports three instruction types: R-type, I-type, and J-type, and utilizes compact encoding with 4-bit opcodes and 5-bit register identifiers.**

**2. Instruction Formats**

**2.1 R-type Format**

**Used for register-to-register arithmetic and logic operations.**

| **Field** | **Bits** | **Description** |
| --- | --- | --- |
| **opcode** | **4** | **Operation code** |
| **rs** | **5** | **Source register** |
| **rt** | **5** | **Target register** |
| **rd** | **5** | **Destination register** |
| **pad** | **5** | **Reserved/unused** |

**Total: 24 bits**

**2.2 I-type Format**

**Used for immediate arithmetic operations, memory access, and branches.**

| **Field** | **Bits** | **Description** |
| --- | --- | --- |
| **opcode** | **4** | **Operation code** |
| **rs/rt** | **5** | **Register (source or base)** |
| **rd** | **5** | **Destination register** |
| **immediate** | **10** | **Immediate value** |

**Total: 24 bits**

**2.3 J-type Format**

**Used for jump operations.**

| **Field** | **Bits** | **Description** |
| --- | --- | --- |
| **opcode** | **4** | **Operation code** |
| **address** | **20** | **Jump address** |

**Total: 24 bits**

**3. Instruction Set Table**

| **Instruction** | **Meaning** | **Type** | **Opcode** | **Format** |
| --- | --- | --- | --- | --- |
| **ADD** | **RD ← RS + RT** | **R** | **0000** | **[op, rs, rt, rd, pad]** |
| **SUB** | **RD ← RS - RT** | **R** | **0001** | **[op, rs, rt, rd, pad]** |
| **AND** | **RD ← RS & RT** | **R** | **0010** | **[op, rs, rt, rd, pad]** |
| **OR** | **RD ← RS|RT** | **R** | **0011** | **[op, rs, rt, rd, pad]** |
| **XOR** | **RD ← RS ^ RT** | **R** | **0100** | **[op, rs, rt, rd, pad]** |
| **SLT** | **RD ← (RS < RT)** | **R** | **0101** | **[op, rs, rt, rd, pad]** |
| **SLL** | **RD ← RT << IMM** | **I** | **0110** | **[op, rt, rd, imm(10)]** |
| **SRL** | **RD ← RT >> IMM** | **I** | **0111** | **[op, rt, rd, imm(10)]** |
| **ADDI** | **RD ← RS + IMM** | **I** | **1000** | **[op, rs, rd, imm(10)]** |
| **ANDI** | **RD ← RS & IMM** | **I** | **1001** | **[op, rs, rd, imm(10)]** |
| **ORI** | **RD ← RS+ IMM** | **I** | **1010** | **[op, rs, rd, imm(10)]** |
| **XORI** | **RD ← RS ^ IMM** | **I** | **1011** | **[op, rs, rd, imm(10)]** |
| **LW** | **RD ← MEM[RS + IMM]** | **I** | **1100** | **[op, rs, rd, imm(10)]** |
| **SW** | **MEM[RS + IMM] ← RD** | **I** | **1101** | **[op, rs, rd, imm(10)]** |
| **BEQ** | **if RS == RT → PC += IMM** | **I** | **1110** | **[op, rs, rt, imm(10)]** |
| **JUMP** | **PC ← address** | **J** | **1111** | **[op, address(20)]** |

**4. Block Diagram (IF + ID Stages)**

**Instruction Fetch (IF)**

* **Program Counter (PC): Holds address of next instruction**
* **Instruction Memory: Returns 24-bit instruction**
* **PC + 1 (for sequential execution)**

**Instruction Decode (ID)**

* **Instruction Register: Holds the current instruction**
* **Opcode Decoder: Decodes 4-bit opcode**
* **Register File: 16 or 32 general purpose registers**
* **Immediate Extractor: Gets 10-bit constant**
* **Control Unit: Sends control signals to next stages (EX/MEM/WB)**

***A detailed diagram can be drawn using tools like draw.io based on this architecture.***

***(View the block diagram on https://app.diagrams.net/#G1BdUpafhVzY2oNjWFEkNSKtUkjK42PiDH#%7B%22pageId%22%3A%22TrvupmfSEGupCLfrMqsl%22%7D)***

