# RISC PROCESSOR
# using Verilog

**Group: 17**

**Group Members:** DARAPU ADHITHYA SHIVA KUMAR REDDY-22CS30019

NAGALLA DEVI SRI PRASAD-22CS10045

**Semester:** Autumn Session- 2024

**Project link:** RISC_FINAL_2k24

Design of Instruction Set Architecture:

In the 32-bit instruction to be given, We divide the 32 bit in the following way

1. Opcode: 6 bits
2. Source Register 1 (rs) : 4 bits
3. Source Register 2 (rt): 4 bits
4. Destination Register (rd) : 4 bits
5. Shift amount bit: 1 bit (encoded as imm[0])
6. Immediate Data: 14bits (both immediate constants and branching address)
7. R[15] is taken as stack pointer

Opcode corresponding to various instructions

a. Arithmetic and Logic Instructions:

| **Operation** | **Opcode** |
|---|---|
| ADD | 000000 |
| ADDI | 010000 |
| SUB | 000001 |
| SUBI | 010001 |
| AND | 000010 |
| ANDI | 010010 |
| OR | 000011 |
| ORI | 010011 |
| XOR | 000100 |
| XORI | 010100 |
| NOT | 000101 |
| SLA | 000110 |
| SLAI | 010110 |
| SRA | 000111 |
| SRAI | 010111 |
| HAM | 001010 |
| SRL | 001000 |

| | |
|---|---|
| SRLI | 011000 |
| LOAD | 011001 |

b. Load and Store

| **Operation** | **Opcode** |
|---|---|
| LD | 110000 |
| ST | 110001 |

c. Branch instruction

| **Operations** | **Opcode** |
|---|---|
| BR | 100000 |
| BMI | 100001 |
| BPL | 100010 |
| BZ | 100011 |

d. Register to Register Transfer

MOVE: Implemented as:

MOVE R2, R1 as : ADD R1, R0, R2

CMOV:

CMOV R1,R2,R3 : if(R1<R2) R3=R1 else R3=R2

| Instruction | Opcode |
|---|---|
| CMOV | 001011 |

Other operations:

HALT: Implemented as a branch instruction to the end of memory

NOP: Implemented as R0 = R0 + R0

**Instruction Type Encoding:**

a. Immediate-Type like ADDI, SUBI, etc:

| Opcode - 6bits | rs - 4bits | rt - 4bits | rd – 4bits | Immediate – 14 bits |
|---|---|---|---|---|

The opcode starts with 01

The first 6 bits are reserved for the operation code. There are 35 operations. Therefore 6 bits.

The next 4 bits are for the source register.

The next 4 bits are don't care inputs

The next 4 bits are the target register

The next 14 bits are for immediate data

The range of immediate data is -16,384 to 16,383

b. Register-Type like ADD, SUB, etc:

| Opcode - 6bits | rs - 4bits | rt - 4bits | rd – 4bits | Immediate – 14 bits |
|---|---|---|---|---|

The opcode starts with 00

The first 6 bits are reserved for the operation code. There are 35 operations. Therefore 6 bits.

The next 4 bits are for the source register 1.

The next 4 bits are for the source register 2.

The next 4 bits are for the destination register.

The next 14 bits are don't care inputs

c. Load:

| Opcode - 6bits | rs - 4bits | rt - 4bits | rd – 4bits | Immediate – 14 bits |
|---|---|---|---|---|

The opcode starts with 11

The first 6 bits are reserved for the operation code.

The next 4 bits are for the source register.

The next 4 bits are don't care inputs.

The next 4 bits are the final target register.

The next 14 bits are for immediate address.

Rd = Mem[rs + Imm]

d. Store:

| **Opcode - 6bits** | **rs - 4bits** | **rt - 4bits** | **rd – 4bits** | **Immediate – 14 bits** |
|---|---|---|---|---|

Opcode starts with 11

The first 6 bits are reserved for the operation code.

The next 4 bits are for the source register.

The next 4 bits are the final target register.

The next 4 bits are don't care inputs.

The next 14 bits are for immediate address.

Mem[rs + imm] = rt

e. Branch:

| **Opcode - 6bits** | **rs - 4bits** | **rt - 4bits** | **rd – 4bits** | **Immediate – 14 bits** |
|---|---|---|---|---|

Opcode starts with 10

The first 6 bits are reserved for the operation code.

The next 4 bits are for the source register.

The next 4 bits are don't care inputs.

The next 4 bits are don't care inputs.

The next 14 bits are for immediate address.

rs is used only for BPL, BMI and BZ to check the condition for branching

MUXES used in the design:

1. Between A(rs) and NPC(next instruction stored)
2. Between B(rt) and Immediate(14 bits)
3. Between Aluout(answer calculated in ALU) and NPC
4. Between Aluout and LMD(memory related read and write data)
5. Between various immediate values

Muxes of type 1 to 4 are 2:1 and last is 4:1 mux

Control signals depending upon the instruction type:

**Control Signals for each instruction:**

**ADD, SUB, AND, OR, XOR, NOT, SLA, SRA, SRL**

| opcode | Mux-1 | Mux-2 | Mux-3 | Mux-4 | Mux-5 |
|---|---|---|---|---|---|
| 000000 | 0 | 0 | 1 | 1 | 0 |
| 000001 | 0 | 0 | 1 | 1 | 0 |
| 000010 | 0 | 0 | 1 | 1 | 0 |
| 000011 | 0 | 0 | 1 | 1 | 0 |
| 000100 | 0 | 0 | 1 | 1 | 0 |
| 000101 | 0 | 0 | 1 | 1 | 0 |
| 000110 | 0 | 0 | 1 | 1 | 0 |
| 000111 | 0 | 0 | 1 | 1 | 0 |
| 001000 | 0 | 0 | 1 | 1 | 0 |

**ADDI, SUBI, ANDI, ORI, XORI, SLAI, SRAI, SRLI, LOAD**

| opcode | Mux-1 | Mux-2 | Mux-3 | Mux-4 | Mux-5 |
|---|---|---|---|---|---|
| 010000 | 0 | 1 | 1 | 1 | 0 |
| 010001 | 0 | 1 | 1 | 1 | 0 |
| 010010 | 0 | 1 | 1 | 1 | 0 |
| 010011 | 0 | 1 | 1 | 1 | 0 |
| 010100 | 0 | 1 | 1 | 1 | 0 |
| 010110 | 0 | 1 | 1 | 1 | 0 |
| 010111 | 0 | 1 | 1 | 1 | 0 |
| 011001 | 0 | 1 | 1 | 1 | 0 |

**LOAD, STORE**

| opcode | Mux-1 | Mux-2 | Mux-3 | Mux-4 | Mux-5 |
|---|---|---|---|---|---|
| 110000 | 0 | 1 | 1 | 0 | 0 |
| 110001 | 0 | 1 | 1 | anything | 0 |

**BRANCH**

| opcode | Mux-1 | Mux-2 | Mux-3 | Mux-4 | Mux-5 |
|---|---|---|---|---|---|
| 100000 | 1 | 1 | 1 | 0 | 0 |
| 100001 | 1 | 1 | depends | 0 | 0 |
| 100010 | 1 | 1 | depends | 0 | 0 |
| 100011 | 1 | 1 | depends | 0 | 0 |

The Mux-3 value depends on the condition being satisfied (whether A <0 || A > 0 || A == 0)

Control unit:

The various Muxes function as follows:

1. **Mux-1**: The first mux-1 selects between A and NPC. For the instructions which involve register operations which fetches rs value A is selected as the instruction involves usage of A.
2. **Mux-2** : The second mux selectes between B and immediate value. For the instructions involving immediate values either a branch address or a shift amount or a const immediate, the immediate value is selected else B is selected.
3. **Mux-3**: The third mux selects between NPC and Aluout. For the branch instructions of conditional type , if the condition is satisfied Aluout is selected which holds the address of where the instruction goes else NPC( the next instruction ) is selected. For the unconditional branch always Aluout is selected as there is a need for a jump always.

   The checking of condition will be done by a module which finally gives the output based on which the value in the mux-3 is selected.

4. **Mux-4** : The $4^{th}$ mux selects the value between LMD and Aluout. For the load type instruction the value which is to be loaded from memory is fetched and is sent to the LMD register. Then this mux selects this value and then is written back into the destination register. If the operation involves simple register-register or register-immediate instruction then Aluout is selected and is written back into the destination register.
5. **Mux-5** : The $5^{th}$ mux selects between 4 types of immediates which might come into instruction based on its type. But in our architecture as every type of immediate is of 14 bits anything can be choosen here as per instruction only 1 immediate value can exist and that to every such immediate is of 14 bits.