CS39001: COMPUTER ORGANIZATION LABORATORY

32-BIT PROCESSOR DESIGN USING VERILOG

Tuhin Mondal (22CS10087) | Diganta Mandal (22CS30062)

1) Introduction

This report documents the design and implementation of a 32-bit RISC-like processor in Verilog, with a focus on the specified architecture and instruction set. The processor supports basic arithmetic, logic, load/store, branch, and program control instructions. A hardwired control unit is employed to manage the flow of instructions through the data path.

2) Processor Architecture

2.1) General Purpose Registers

Registers: The processor consists of 16 general-purpose registers, R0 to R15.
Register R0: Special read-only register, always containing the value 0.
Read/Write Ports: The register file has two read ports and one write port.
Program Counter (PC): 32-bit SPR responsible for holding the current instruction address.
Stack Pointer (SP): 32-bit SPR which is stack pointer for stack operations.

2.2) Memory and Addressing

Memory Addressing: Memory is byte-addressable using 32-bit addresses. All operations are on 32-bit data, and loads/stores access addresses that are multiples of 8.

2.3) Addressing Modes

a) Register Addressing: Operands are in registers.
b) Immediate Addressing: Operand is an immediate constant.
c) Base Addressing: Register is used as the base address for memory access.
d) PC Relative Addressing: Used for branch instructions, the operand is relative to the PC.
e) Indirect Addressing: Memory address is provided indirectly via a register.

3) Instruction Set

3.1) Instruction Encoding

There are 3 types of Instruction encoding in our processor, namely R-type, I-type and J-type:

Type	**31-**			**Encoding (bits)**		**-0**
R-Type	opcode (6)	rs (4)	rt (4)	rd (4)	shamt(10)	funct (4)
I-Type	opcode (6)	rs (4)	rt (4)	immediate (18)		
J-Type	opcode (6)	address (26)				

3.2) Instruction Types and Opcode Assignment

Instruction Category	Instructions	Encoding Type	Opcode	Function	Usage
Arithmetic & Logic Instructions (Non-Immediate Type)	ADD	R-type	000000	0000	add rd, rs, rt
	SUB	R-type	000000	0001	sub rd, rs, rt
	NOT	R-type	000000	0010	not rt, rs
	SLL	R-type	000000	0011	sll rd, rs, rt
	AND	R-type	000000	0100	and rd, rs, rt
	OR	R-type	000000	0101	or rd, rs, rt
	SRL	R-type	000000	0110	srl rd, rs, rt
	SRA	R-type	000000	0111	sra rd, rs, rt
	XOR	R-type	000000	1000	xor rd, rs, rt
	NOR	R-type	000000	1001	nor rd, rs, rt
	INC	R-type	000000	1010	inc rs
	DEC	R-type	000000	1011	dec rs
	SLT	R-type	000000	1100	slt rd, rs, rt
	SGT	R-type	000000	1101	sgt rd, rs, rt
	HAM	R-type	000000	1110	ham rt, rs

Instruction Category	Instructions	Encoding Type	Opcode	Function	Usage
Arithmetic & Logic Instructions (Immediate Type)	LUI	I-type	111111	XXXX	lui rs, (imm)
	ADDI	I-type	110000	XXXX	addi rt, rs, (imm)
	SUBI	I-type	110001	XXXX	subi rt, rs, (imm)
	NOTI	I-type	110010	XXXX	noti rs, (imm)
	SLLI	I-type	110011	XXXX	slli rt, rs, (imm)
	ANDI	I-type	110100	XXXX	andi rt, rs, (imm)
	ORI	I-type	110101	XXXX	ori rt, rs, (imm)
	SRLI	I-type	110110	XXXX	srli rt, rs, (imm)
	SRAI	I-type	110111	XXXX	srai rt, rs, (imm)
	XORI	I-type	111000	XXXX	xori rt, rs, (imm)
	NORI	I-type	111001	XXXX	nori rt, rs, (imm)

Instruction Category	Instructions	Encoding Type	Opcode	Function	Usage
Load & Store Instructions	LD	I-type	000001	XXXX	ld rt, rs, (imm)
	ST	I-type	000010	XXXX	st rt, rs, (imm)
Branch Instructions	BR	J-type	000011	XXXX	br (imm)
	BMI	I-type	000100	XXXX	bmi rs, (imm)
	BPL	I-type	000101	XXXX	bpl rs, (imm)
	BZ	I-type	000110	XXXX	bz rs, (imm)
Register to Register Transfer Instructions	MOVE	R-type	000111	0010	move rt, rs
	CMOV	R-type	000111	0001	cmov rd, rs, rt
Program Control Instructions	HALT	J-type	001000	XXXX	halt
	NOP	J-type	001001	XXXX	nop

4) Design Implementation

4.1) ALU Design

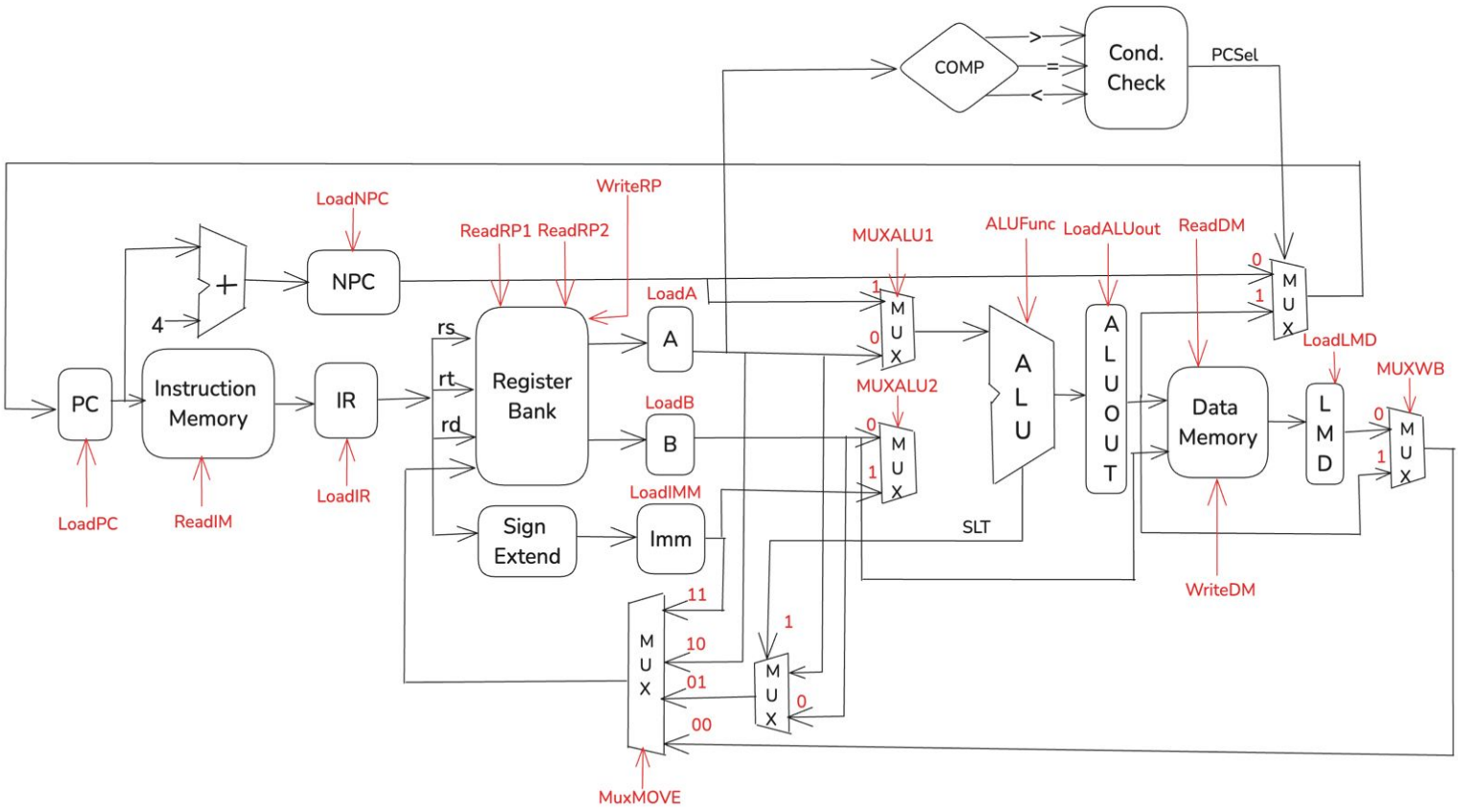
ALU Inputs:

1) Operand A: 32-bit input for the first operand from ALU MUX-1
2) Operand B: 32-bit input for the second operand from ALU MUX-2
3) Control Signal: N-bit control signal to specify the ALU Function

ALU Outputs:

1) ALU Result: 32-bit result of the operation.
2) SLT Flag: 1-bit flag, set to 1 if the result of SLT is 1.
3) SGT Flag: 1-bit flag, set to 1 , set to 1 if the result of SGT is 1.

4.2) Data Path Design:



4.3) Control Path Design:

4.3.1) Control Signals:

1) **LoadNPC -**Load NPC with the immediate next address of instruction stored in PC
2) **ReadIM** – The address from PC used to fetch the instruction from Instruction memory
3) **LoadIR** - Loads the Instruction Register with the instruction fetched from instruction memory
4) **ReadRP1** – Read content of register address mentioned in source register 1
5) **ReadRP2** – Read content of register address mentioned in source register 2
6) **LoadA** – Load the content read form register bank of source register 1 intro register A
7) **LoadB** – Load the content read from register bank of source register 2 intro register 2
8) **LoadIMM** – Load any Immediate value from the IR into register storing immediate value
9) **MuxALU1** – Set MUX to 0 to load A, 1 to load NPC into ALU as first operand,
10) **MuxALU2** – Set MUX to 0 to load B, 1 to load immediate value into ALU as second operand
11) **ALUfunc** – Specifies the arithmetic operation to be performed by the ALU
12) **LoadALUOut** – Load the output obtained from ALU into the register ALUOut
13) **LoadPC** - Load the Program counter with the next instruction address
14) **ReadDM** – Read a value from the data memory
15) **WriteDM** - Write a value into the Data Memory
16) **LoadLMD** – Load the LMD with value fetched from the Data memory
17) **MuxWB** – Set it to 0 to select value stored in LMD or set it to 1 select value stored in ALUOut
18) **WriteRP** – Writes a new value into the register bank as pointed by the destination register
19) **MuxMOVE** – 10 for MOVE, 01 for CMOV and 00 for other instructions
20) **HALT** – Set to 1 when HALT instruction is executed, otherwise 0

4.3.2) Control Signals with Timestamps:

A) Arithmetic & Logic Instructions (Non-Immediate Type)	
T1	LoadNPC, ReadIM, WMFC
T2	LoadIR
T3	LoadA, LoadB, MuxALU1 = 0, MuxALU2 = 0, ALUFunc = (func), LoadALUOut
T4	LoadPC
T5	MuxWB = 1, WriteRP, MuxMOVE = 00

B) Arithmetic & Logic Instructions (Immediate Type)	
T1	LoadNPC, ReadIM, WMFC
T2	LoadIR
T3	LoadA, LoadIMM, MuxALU1 = 0, MuxALU2 = 1, ALUFunc = opcode[3:0], LoadALUOut
T4	LoadPC
T5	MuxWB = 1, WriteRP, MuxMOVE = 00

C) Load & Store Instructions	
1) LOAD	
T1	LoadNPC, ReadIM, WMFC
T2	LoadIR
T3	LoadA, LoadB, LoadIMM, MuxALU1 = 0, MuxALU2 = 1, ALUFunc = 0000, LoadALUOut
T4	LoadPC, ReadDM, WMFC
T5	LoadLMD, MuxWB = 0, WriteRP, MuxMOVE = 00

2) STORE	
T1	LoadNPC, ReadIM, WMFC
T2	LoadIR
T3	LoadA, LoadB, LoadIMM, MuxALU1 = 0, MuxALU2 = 1, ALUFunc = 0000, LoadALUOut
T4	LoadPC, WriteDM, WMFC
T5	MuxWB = 0, MuxMOVE = 00

D) Branch Instructions	
1) BR	
T1	LoadNPC, ReadIM, WMFC
T2	LoadIR
T3	LoadIMM, MuxALU1 = 1, MuxALU2 = 1, ALUFunc = 0000, LoadALUOut
T4	LoadPC
T5	MuxWB = 0, MuxMOVE = 00

2) BPL, BMI, BZ	
T1	LoadNPC, ReadIM, WMFC
T2	LoadIR
T3	LoadA, LoadIMM, MuxALU1 = 1, MuxALU2 = 1, ALUFunc = 0000, LoadALUOut
T4	LoadPC
T5	MuxWB = 0, MuxMOVE = 00

E) Move Instructions	
1) MOVE	
T1	LoadNPC, ReadIM, WMFC
T2	LoadIR
T3	LoadA
T4	LoadPC
T5	MuxMOVE = 10, WriteRP
2) CMOV	
T1	LoadNPC, ReadIM, WMFC
T2	LoadIR
T3	LoadA, LoadB
T4	LoadPC
T5	MuxMOVE = 01, WriteRP

F) Program Control Instructions	
1) HALT	
T1	HALT
T2	-
T3	-
T4	-
T5	-
2) NOP	
T1	LoadNPC, ReadIM, WMFC
T2	LoadIR
T3	-
T4	LoadPC
T5	-

4.3.3) Control Path Design: