LAB REPORT
PROJECT

BS-Computer Engineering

Session: 2023-2027

Course Instructor: Mr. Aftab Alam
Lab Instructor: Sir Usama Riaz

Submitted By: Ashba Khaliq **Roll No:** BSCE23027
Submitted By: Minahil Imran **Roll No:** BSCE23030

Block diagram of the pipelined datapath:

Single-Cycle Data path:

Type	Instructions
arithmetic (unsigned)	addu, subu, addiu
arithmetic (signed)	add, sub, addi
Logical	and, andi, or, ori, xor, xori
Shift	sll, sra, srl
Compare	slt, slti, sltu
Control	beq, bne, blez,
data transfer	lw, sw,

RISC-V Instructions Format:

Instruction	Format	funct7	rs2	rs1	funct3	rd	opcode
add (add)	R	0000000	reg	reg	000	reg	0110011
sub (sub)	R	0100000	reg	reg	000	reg	0110011
Instruction	**Format**	**immediate**		**rs1**	**funct3**	**rd**	**opcode**
addi (add immediate)	I	constant		reg	000	reg	0010011
ld (load doubleword)	I	address		reg	011	reg	0000011
Instruction	**Format**	**immed -iate**	**rs2**	**rs1**	**funct3**	**immed -iate**	**opcode**
sd (store doubleword)	S	address	reg	reg	011	address	0100011

Name (Bit position)	Fields 31:25	24:20	19:15	14:12	11:7	6:0
(a) R-type	funct7	rs2	rs1	funct3	rd	opcode
(b) I-type	immediate[11:0]		rs1	funct3	rd	opcode
(c) S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode
(d) SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode

RISC-V Instructions Opcodes:

Instruction Type	Opcode Value
R-Type	0110011
I-Type	0000011
S-Type	0100011
SB-Type	1100011

ALU Control Lines:

ALU control lines	Function
0000	ADD
0001	SLL
0010	SLT
0011	SUB
0100	XOR
0101	SRL
0110	OR
0111	AND
1000	BEQ
1001	BNE
1010	BLT
1011	BGE
1100	BLTU
1101	BGEU
1110	SLLA

RISC-V Instructions Opcodes:

R Type:

Format	Instruction	Opcode	Funct3	Funct6/7
R-type	add	0110011	000	0000000
	sub	0110011	000	0100000
	sll	0110011	001	0000000
	xor	0110011	100	0000000
	srl	0110011	101	0000000
	sra	0110011	101	0000000
	or	0110011	110	0000000
	and	0110011	111	0000000
	lr.d	0110011	011	0001000
	sc.d	0110011	011	0001100

I Type:

I-type	lb	0000011	000	n.a.
	lh	0000011	001	n.a.
	lw	0000011	010	n.a.
	ld	0000011	011	n.a.
	lbu	0000011	100	n.a.
	lhu	0000011	101	n.a.
	lwu	0000011	110	n.a.
	addi	0010011	000	n.a.
	slli	0010011	001	000000
	xori	0010011	100	n.a.
	srli	0010011	101	000000
	srai	0010011	101	010000
	ori	0010011	110	n.a.
	andi	0010011	111	n.a.
	jalr	1100111	000	n.a.

S/SBType:

S-type	sb	0100011	000	n.a.
	sh	0100011	001	n.a.
	sw	0100011	010	n.a.
	sd	0100011	111	n.a.
SB-type	beq	1100111	000	n.a.
	bne	1100111	001	n.a.
	blt	1100111	100	n.a.
	bge	1100111	101	n.a.
	bltu	1100111	110	n.a.
	bgeu	1100111	111	n.a.

Instruction	ALUOp	Operation	funct7	funct3	ALU Action	ALU_Cntl
lw	00	Load word	X	X	Add	0000
sw	00	Store word	X	X	Add	0000
beq	01	Branch if equal	X	000	Subtract + compare	1000
bne	01	Branch if not equal	X	001	Subtract + compare	1001
blt	01	Branch if less than	X	100	Set if less than	1010
bge	01	Branch if greater than equal to	X	101	Set if greater equal	1011
bltu	01	Branch if less than(U)	X	110	Unsigned less than	1100
bgeu	01	Branch if greater than equal to (U)	X	111	Unsigned greater equal	1101
add, addi	10	add	0000000	000	Add	0000
sub	10	sub	0100000	000	Subtract	0011
and, andi	10	and	0000000	111	And	0111
or, ori	10	or	0000000	110	Or	0110
xor, xori	10	xor	0000000	100	Xor	0100
sll, slli	10	Shift left logical	0000000	001	Shift left logical	0001
srl, srli	10	Shift right logical	0000000	101	Shift right logical	0101
sra, srai	10	Shift rigth arithematic	0100000	101	Shift right arithmetic	0101
slt, slti	10	Set less than	0000000	010	Set less than	0010
sltu, sltiu	10	Set less than (U)	0000000	011	Set less than unsigned	0010

EXPLANATION:

1. Fetch Stage:
The fetch stage retrieves the instruction from memory using the program counter (PC), calculates the next PC and PC + 4, and outputs the instruction, current PC, and PC + 4 to the decode stage.

2. Decode Stage:
The decode stage extracts the instruction's opcode, funct3, and funct7, reads source registers (rs1, rs2) from the register file, generates an immediate value and control signals, including ALU controls, and outputs operands (rd1, rd2), control signals, and the destination register (rd).

3. Execute Stage:
The execute stage performs ALU operations using operands,which may be forwarded from memory or write-back stages, computes branch targets, determines branching,and outputs the ALU result, memory write data, branch decision, and destination register.

4. Memory Stage:
The execute stage performs ALU operations using operands ,which may be forwarded from memory or write-back stages, computes branch targets, determines branching and outputs the ALU result, memory write data, branch decision, and destination register.

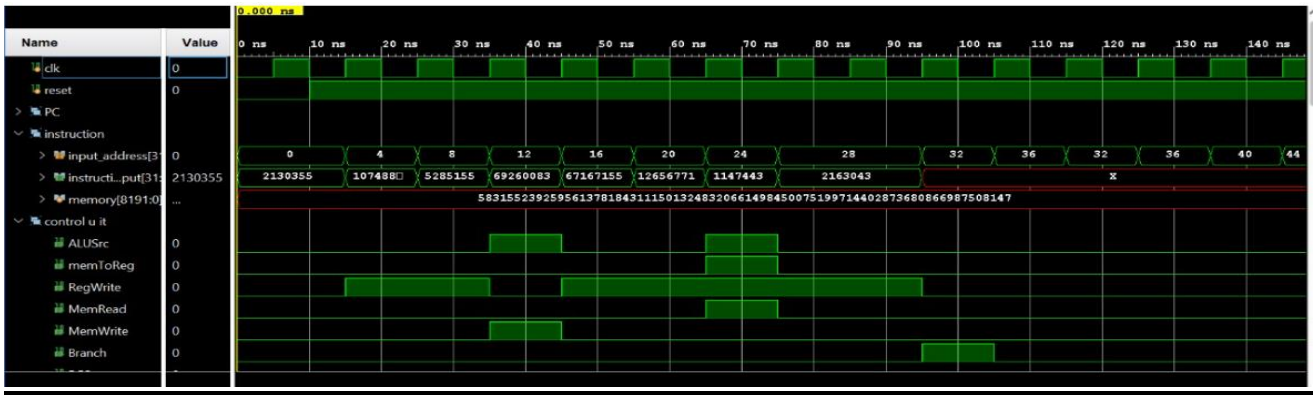
5. Write Back Stage:
The write-back stage selects memory read data or the ALU result and writes it to the register file if RegWrite is active, finalizing the instruction's execution.

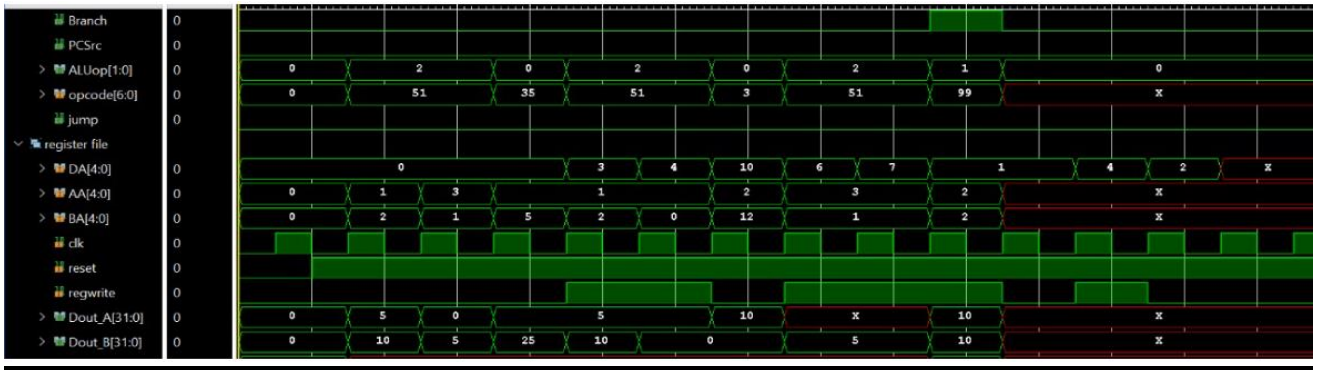
Code:

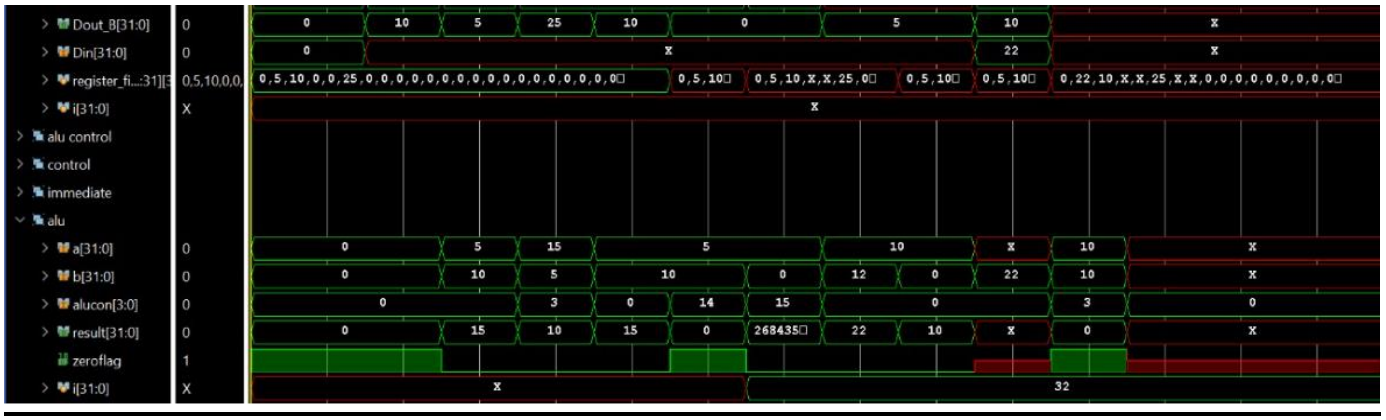
```
memory[0*8+:32] = 32'b0000000_00010_00001_000_00011_0110011; // add x3, x1, x2

memory[4*8+:32] = 32'b0100000_00001_00011_000_00100_0110011; // sub x4, x3, x1

memory[8*8+:32] = 32'b0000000_00101_00001_010_01010_0100011; // sw x5, 10(x1)

memory[12*8+:32] = 32'b0000010_00010_00001_101_00110_0110011; // gt x6, x1, x2

memory[16*8+:32] = 32'b0000010_00000_00001_110_00111_0110011; // reverse x7, x1

memory[20*8+:32] = 32'b000000001100_00001_010_01000_0000011; // lw x8, 12(x1)

memory[24*8+:32] = 32'b0000000_00010_00001_000_00010_1100011; // beq x1, x2, +2
```

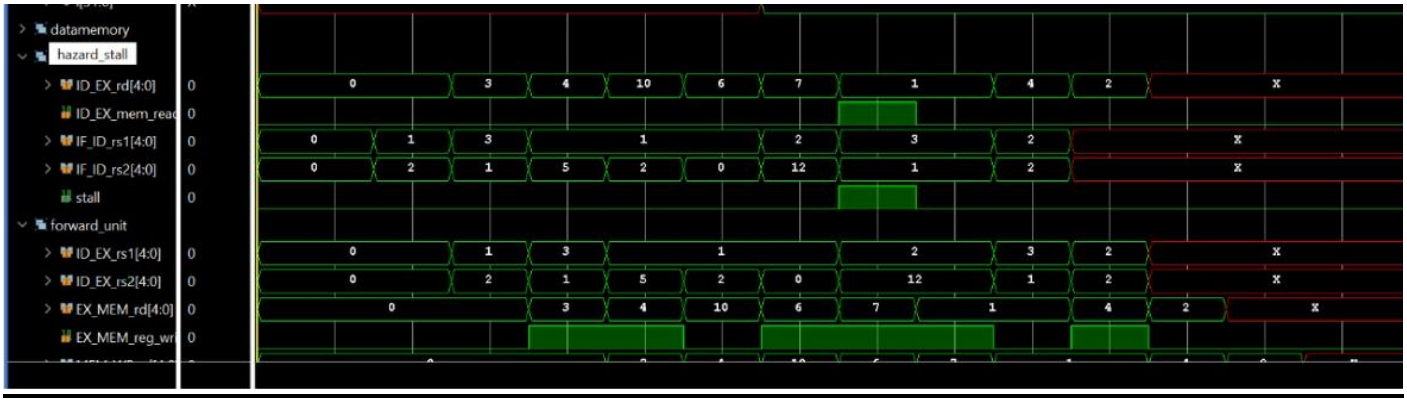
memory[28*8+:32] = 32'b0000000_00010_00010_000_00010_1100011; // beq x2, x2, +2

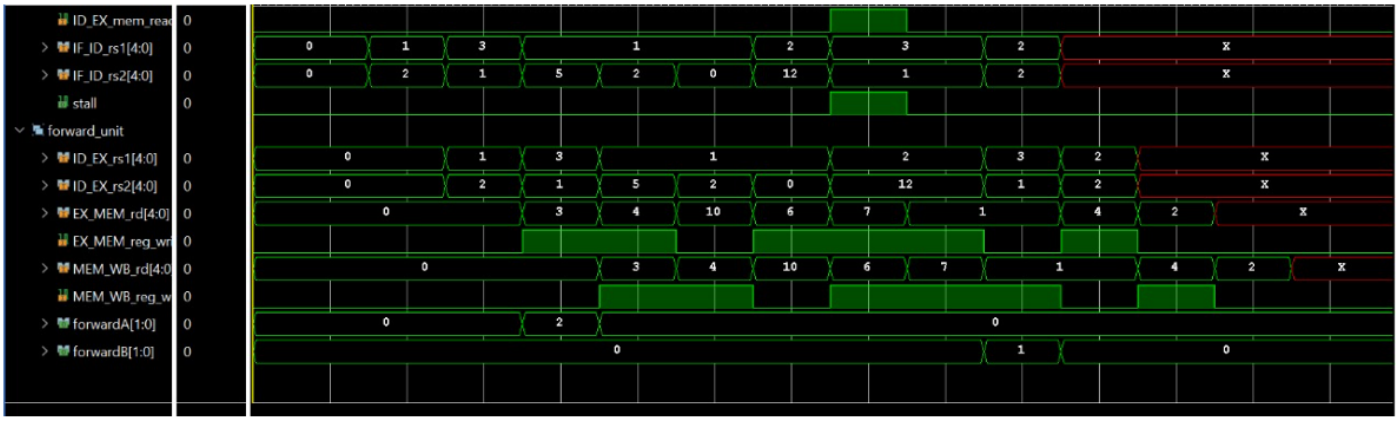
Waveforms:









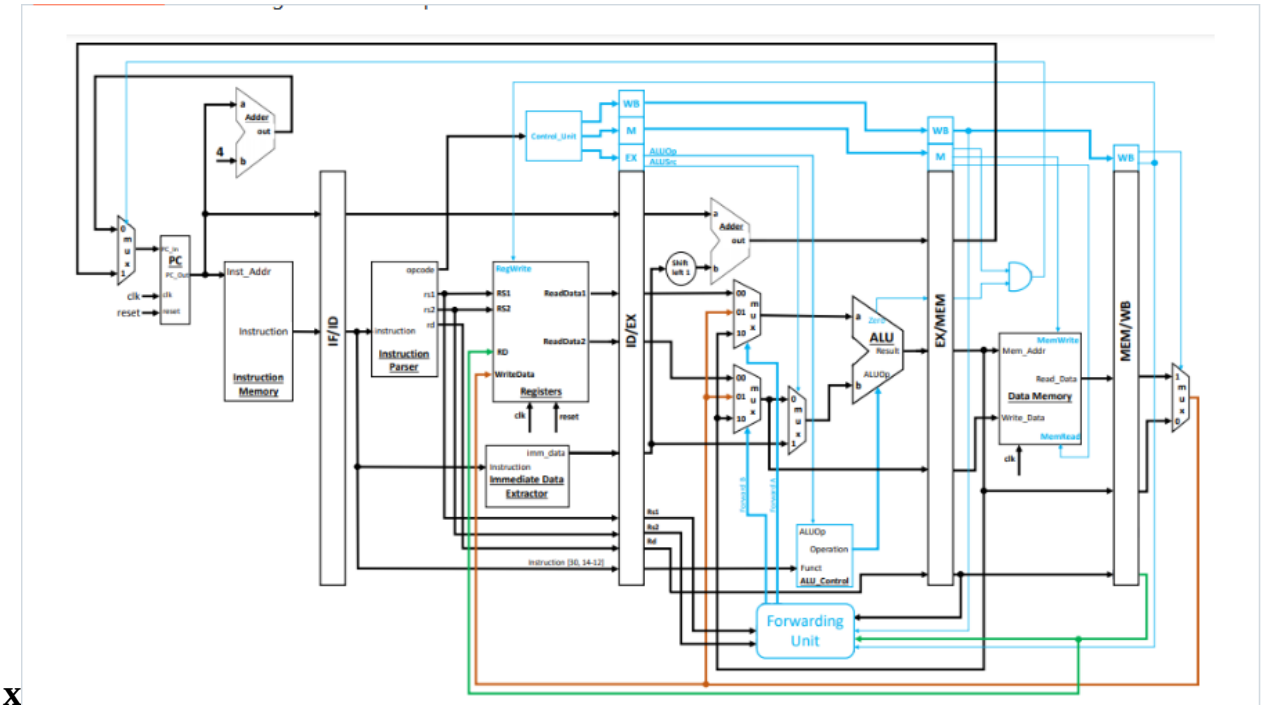


Hazard Handling in a 5-Stage RISC-V Pipeline:

In a 5-stage RISC-V pipeline (fetch, decode, execute, memory, write-back), data hazards are addressed by forwarding results from the execute or memory stages to the execute stage's operands, with stalls inserted for unresolved load-use dependencies. Control hazards from branches are managed by predicting outcomes and flushing the pipeline if incorrect, updating the PC with the branch target.Structural hazards are avoided using separate instruction and data memories, ensuring efficient pipeline execution with minimal delays.

Block Diagram:

PIPELINED PROCESSOR:


Adder
4
PC
Inst_Addr
Instruction
Instruction Memory
IF/ID
Instruction Parser
RegWrite
RS1
RS2
RD
WriteData
ReadData1
ReadData2
Registers
Immediate Data Extractor
Control_Unit
WB
M
EX
ID/EX
Shift left 1
ALU
Result
ALUOp
EX/MEM
MemWrite
Mem_Addr
Read_Data
Data Memory
Write_Data
MemRead
MEM/WB
ALU_Control
Operation
Funct
Forwarding Unit

x