IPA Project Report

Team - 5

Sanjana Sheela - 2023102027

Snigdha Stp - 2023102036

Khyathi Sri Basireddy - 2023102065

TASK - 1: Sequential Implementation

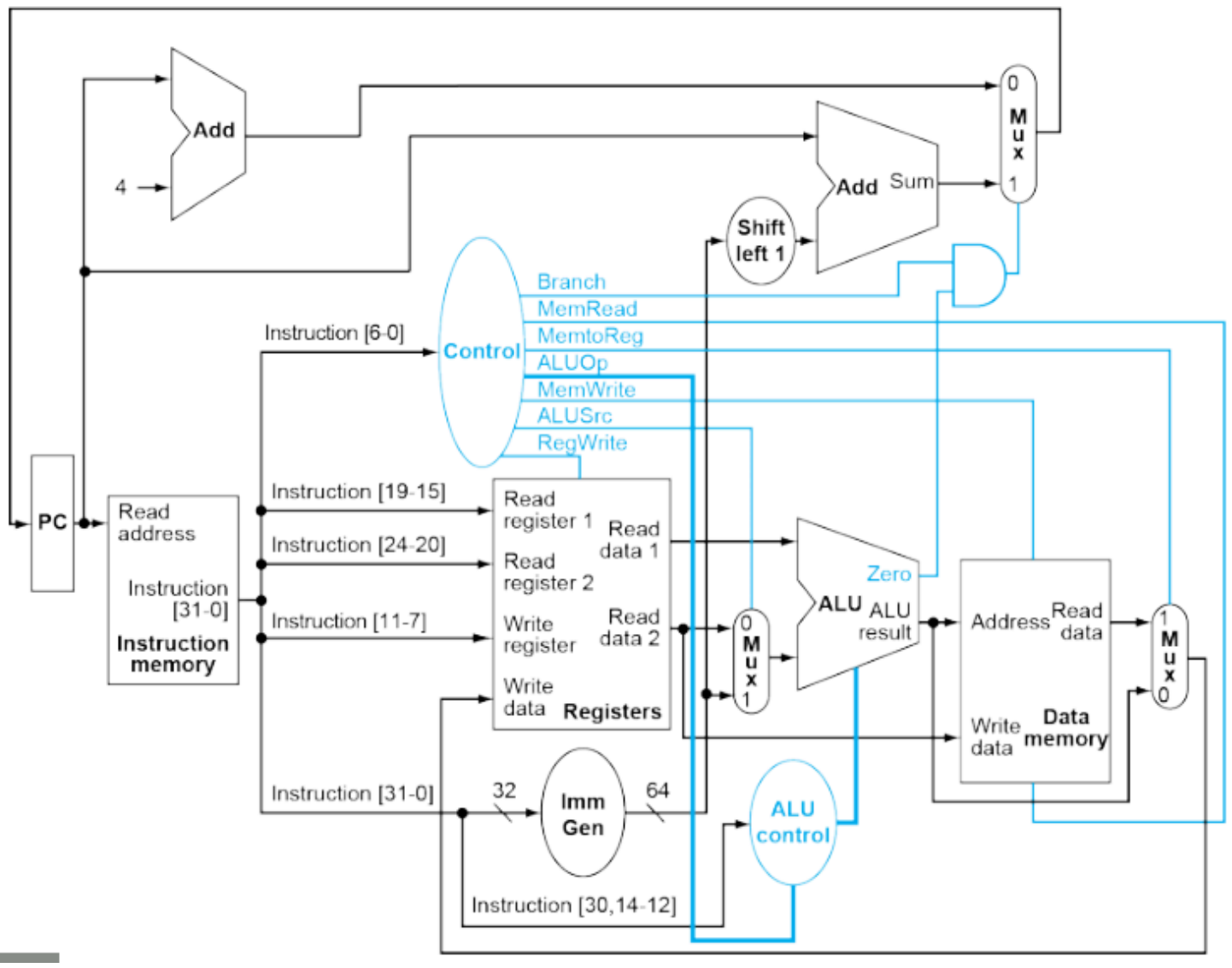
A sequential processor executes instructions one after the other in a structured manner, ensuring that each instruction completes before the next begins. Unlike Pipeline Implementation, where multiple instructions can be processed simultaneously, a sequential processor follows a strict step-by-step execution.

To improve efficiency and maintain an organized execution flow, the processor can be differentiated into five stages.

These five stages are

1. **Instruction Fetch (IF)**
2. **Instruction Decode (ID)**
3. **Execute (EX)**
4. **Memory Access (MEM)**
5. **Write Back (WB)**

The Data path of the complete implementation is shown below:



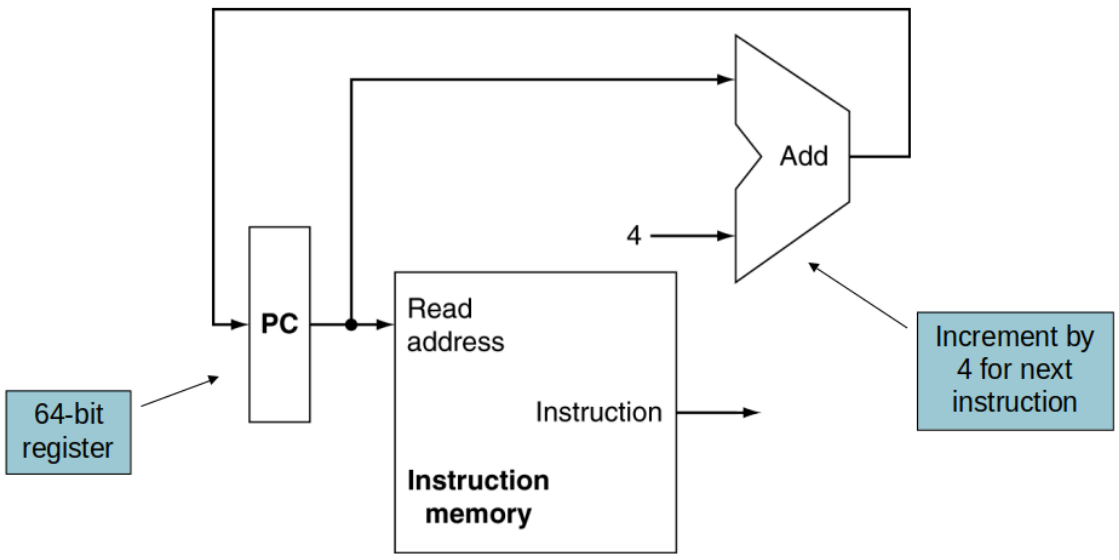
We have 32 bit Instructions in a RISC-V ISA.

We will be executing the following instructions from RISC-V ISA: add, sub, and, or, ld, sd and beq.

1. Instruction Fetch (IF) Stage

The Fetch is responsible for retrieving the next instruction from memory

- The **Program Counter (PC)** is a 64 bit register that holds the address of the next instruction to be executed.
- This address is sent to **Instruction Memory**, which fetches the 32-bit instruction.
- The PC is updated by adding **4** to point to the next instruction. This is done using an **adder** (top left).
- A multiplexer (**MUX**) determines whether to fetch the next sequential instruction (PC + 4) or a branch target address based on the branch control signal.
- The output is sent to the **Instruction Decode (ID) stage**.



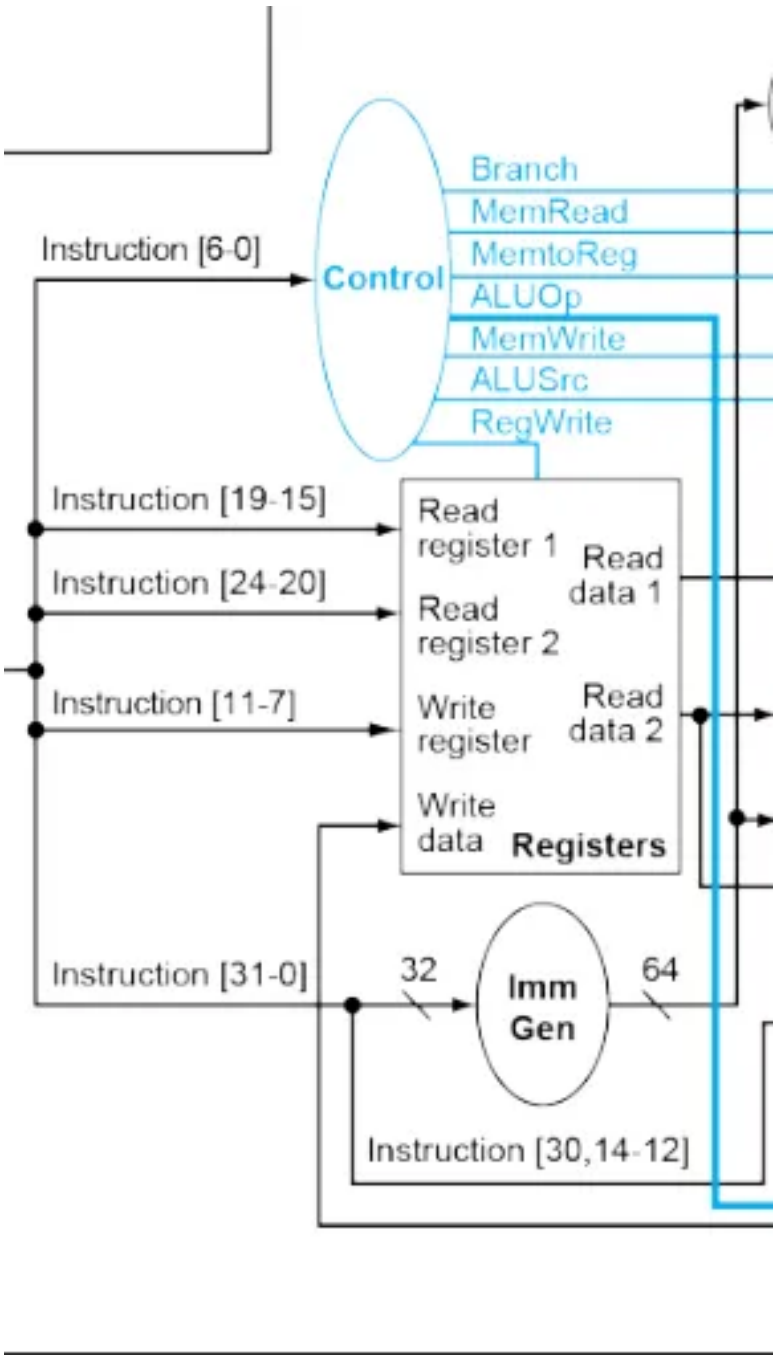
2. Instruction Decode (ID) Stage

The **Instruction Decode (ID) stage** interprets the fetched instruction by extracting its opcode, register addresses, and immediate values, allowing the processor to identify the required operands for execution.

1. **Opcode (bits 6-0)** → Sent to the **Control Unit** to generate control signals. The Opcodes are as follows:

- R-type - 0110011
- Load - 0000011
- Store - 0100011
- Branch - 1100011

2. **Register Addresses (bits 19:15 & 24:20)** → Used to read values from the **Register File**.
3. **Destination Register (bits 11:7)**→ Used to store result of an operation in the **Register File**.
4. **Immediate Generator (Imm Gen)** extracts **immediate values** (for instructions like load/store).
5. The **Control Unit** generates signals for the **ALU, memory, and multiplexers (MUX)**. These are highlighted in **blue**.



3. Execute (EX) Stage

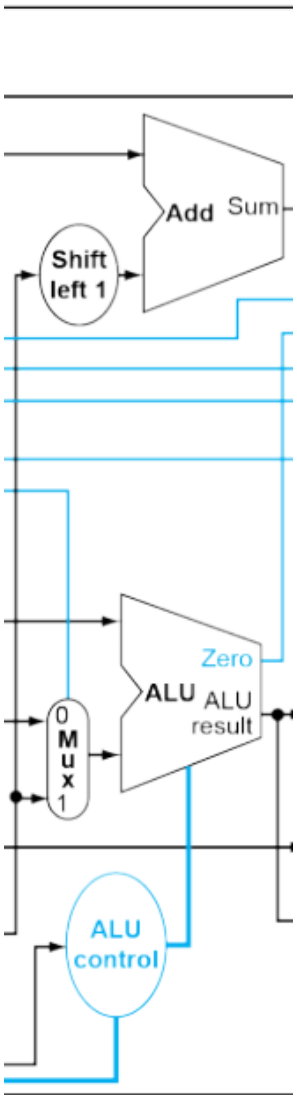
- The ALU performs operations based on the instruction:
- Takes two inputs: **register data** or an **immediate value** (selected using a MUX controlled by `AluSrc`).
- The **ALU Control Unit** (bottom right) determines the operation (e.g., addition, subtraction, AND, OR) which is decided on the basis of `AluOp`.

Alu Control	Function
0000	AND
0001	OR
0010	add
0110	subtract

- The **Branch Adder** (top right) calculates the branch target address if the instruction is a branch.

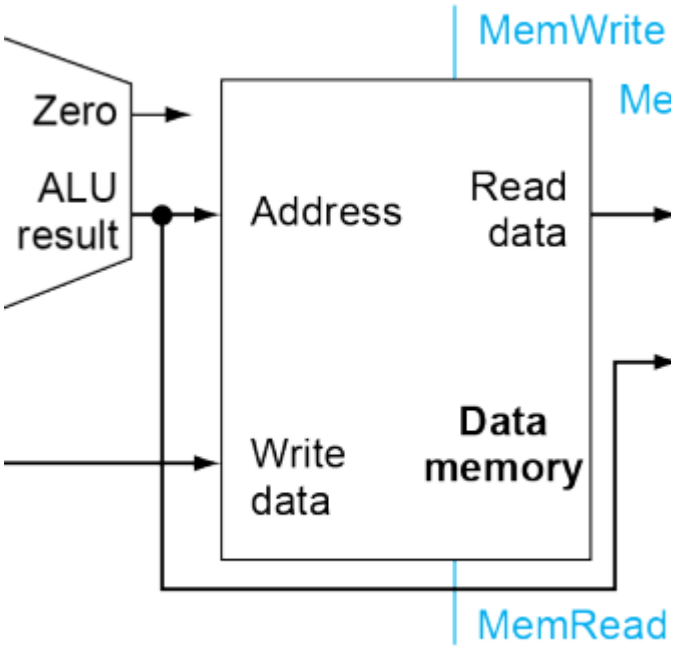
  Branch Address = PC + 2(Imm)

opcode	ALUOp	Operation	Opcode field	ALU function	ALU control
ld	00	load register	XXXXXXXXXXX	add	0010
sd	00	store register	XXXXXXXXXXX	add	0010
beq	01	branch on equal	XXXXXXXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001



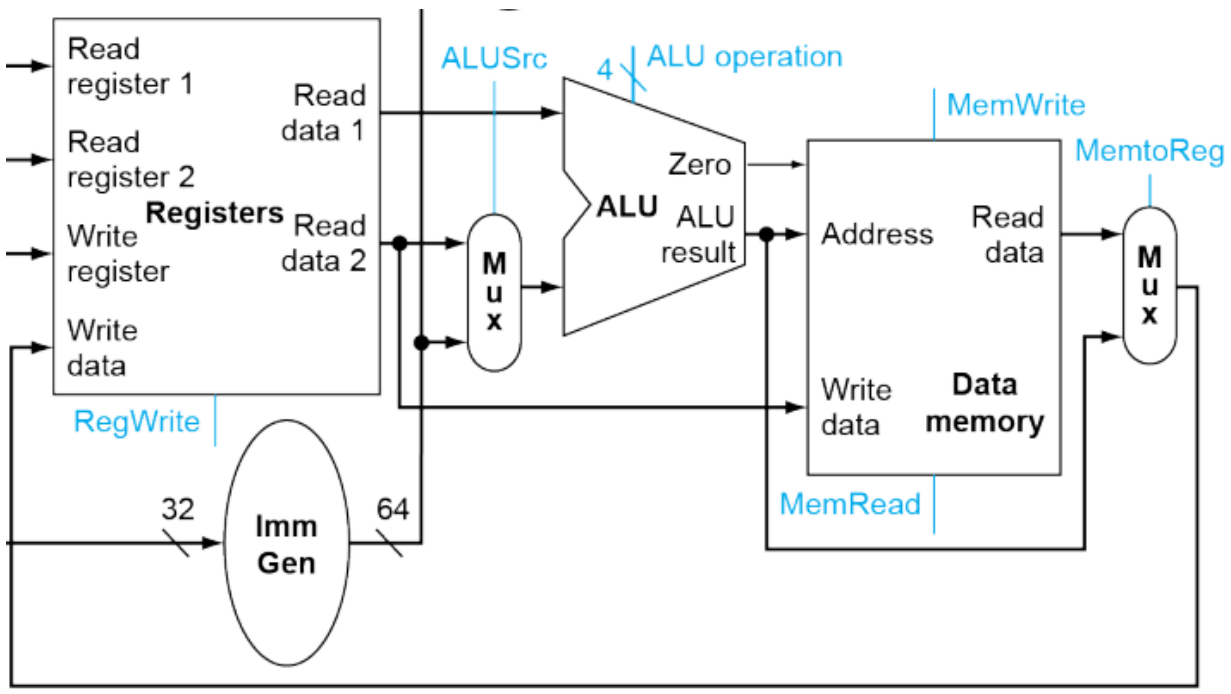
4. Memory Access (MEM) Stage

- If the instruction involves memory access:
  - A **load instruction** reads data from **Data Memory**.
  - A **store instruction** writes data into **Data Memory**.
- The `MemRead` and `MemWrite` signals from the **Control Unit** determine memory operations.
- The **MUX (bottom right)** decides whether the next data comes from memory or the ALU result.



5. Write Back (WB) Stage

- The final result is written back to the **Register File**:
  - If the instruction is an ALU operation, the result is written from the ALU.
  - If it is a load instruction, the data is written from **Data Memory**.
- The **MUX at the end** (bottom right) decides what is written back.
- The `RegWrite` signal from the **Control Unit** enables writing to registers.



OUTPUTS OF SEQUENTIAL IMPLEMENTATION

ADD Operation

```
002081B3  // add x3, x1, x2
005202B3  // add x5, x5, x4
00328333  // add x6, x3, x5
003181B3  // add x3, x3, x3
```

$x_3 = x_2 + x_1$

```
Time: 10000
Current PC =                    0 instruction = 002081b3
rs1= 1, rs2= 2, rd= 3
ReadData1 = 00000000000000000000000000000000000000000000000000000000000000001, ReadData2 = 00000000000000000000000000000000000000000000000000000000000000010
ALU Output (valE) = 00000000000000000000000000000000000000000000000000000000000000011, Zero Flag = 0
Memory Read Data = 00000000000000000000000000000000000000000000000000000000000000000, Write Data = 00000000000000000000000000000000000000000000000000000000000000011
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=0
reg[ 0] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 00000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 00000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 00000000000000000000000000000000000000000000000000000000000000011  reg[ 4] = 00000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 00000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 00000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 00000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 00000000000000000000000000000000000000000000000000000000000000000  reg[10] = 00000000000000000000000000000000000000000000000000000000000001010  reg[11] = 00000000000000000000000000000000000000000000000000000000000001011
reg[12] = 00000000000000000000000000000000000000000000000000000000000001100  reg[13] = 00000000000000000000000000000000000000000000000000000000000000000  reg[14] = 00000000000000000000000000000000000000000000000000000000000000000
reg[15] = 00000000000000000000000000000000000000000000000000000000000000000  reg[16] = 00000000000000000000000000000000000000000000000000000000000000000  reg[17] = 01111111111111111111111111111111111111111111111111111111111111111
reg[18] = 10000000000000000000000000000000000000000000000000000000000000000  reg[19] = 11111111111111111111111111111111111111111111111111111111111111111  reg[20] = 00000000000000000000000000000000000000000000000000000000000000000
reg[21] = 00000000000000000000000000000000000000000000000000000000000000000  reg[22] = 00000000000000000000000000000000000000000000000000000000000000000  reg[23] = 00000000000000000000000000000000000000000000000000000000000000000
reg[24] = 00000000000000000000000000000000000000000000000000000000000000000  reg[25] = 00000000000000000000000000000000000000000000000000000000000000000  reg[26] = 00000000000000000000000000000000000000000000000000000000000000000
reg[27] = 00000000000000000000000000000000000000000000000000000000000000000  reg[28] = 00000000000000000000000000000000000000000000000000000000000000000  reg[29] = 00000000000000000000000000000000000000000000000000000000000000000
reg[30] = 10101010101010101010101010101010101010101010101010101010101010101  reg[31] = 11111111111111111111111111111111111111111111111111111111111111100
```

$$x_5 = x_5 + x_4$$

```
Time: 30000
Current PC =                    4 instruction = 005202b3
rs1= 4, rs2= 5, rd= 5
ReadData1 = 00000000000000000000000000000000000000000000000000000000000000100, ReadData2 = 00000000000000000000000000000000000000000000000000000000000000101
ALU Output (valE) = 00000000000000000000000000000000000000000000000000000000000001001, Zero Flag = 0
Memory Read Data = 00000000000000000000000000000000000000000000000000000000000000000, Write Data = 00000000000000000000000000000000000000000000000000000000000001001
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=0
reg[ 0] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 00000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 00000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 00000000000000000000000000000000000000000000000000000000000000011  reg[ 4] = 00000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 00000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 00000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 00000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 00000000000000000000000000000000000000000000000000000000000000000  reg[10] = 00000000000000000000000000000000000000000000000000000000000001010  reg[11] = 00000000000000000000000000000000000000000000000000000000000001011
reg[12] = 00000000000000000000000000000000000000000000000000000000000001100  reg[13] = 00000000000000000000000000000000000000000000000000000000000000000  reg[14] = 00000000000000000000000000000000000000000000000000000000000000000
reg[15] = 00000000000000000000000000000000000000000000000000000000000000000  reg[16] = 00000000000000000000000000000000000000000000000000000000000000000  reg[17] = 01111111111111111111111111111111111111111111111111111111111111111
reg[18] = 10000000000000000000000000000000000000000000000000000000000000000  reg[19] = 11111111111111111111111111111111111111111111111111111111111111111  reg[20] = 00000000000000000000000000000000000000000000000000000000000000000
reg[21] = 00000000000000000000000000000000000000000000000000000000000000000  reg[22] = 00000000000000000000000000000000000000000000000000000000000000000  reg[23] = 00000000000000000000000000000000000000000000000000000000000000000
reg[24] = 00000000000000000000000000000000000000000000000000000000000000000  reg[25] = 00000000000000000000000000000000000000000000000000000000000000000  reg[26] = 00000000000000000000000000000000000000000000000000000000000000000
reg[27] = 00000000000000000000000000000000000000000000000000000000000000000  reg[28] = 00000000000000000000000000000000000000000000000000000000000000000  reg[29] = 00000000000000000000000000000000000000000000000000000000000000000
reg[30] = 10101010101010101010101010101010101010101010101010101010101010101  reg[31] = 11111111111111111111111111111111111111111111111111111111111111100
```

$$x_6 = x_5 + x_3$$

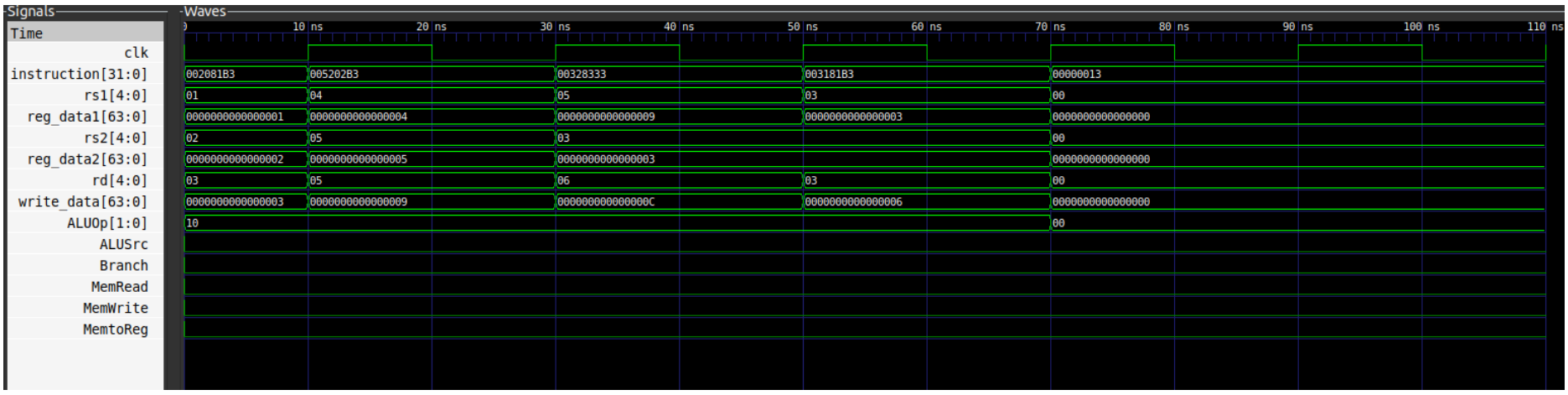
```
Time: 50000
Current PC =                    8 instruction = 00328333
rs1= 5, rs2= 3, rd= 6
ReadData1 = 00000000000000000000000000000000000000000000000000000000000001001, ReadData2 = 00000000000000000000000000000000000000000000000000000000000000011
ALU Output (valE) = 00000000000000000000000000000000000000000000000000000000000001100, Zero Flag = 0
Memory Read Data = 00000000000000000000000000000000000000000000000000000000000000000, Write Data = 00000000000000000000000000000000000000000000000000000000000001100
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=0
reg[ 0] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 00000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 00000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 00000000000000000000000000000000000000000000000000000000000000011  reg[ 4] = 00000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 00000000000000000000000000000000000000000000000000000000000001001
reg[ 6] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 00000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 00000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 00000000000000000000000000000000000000000000000000000000000000000  reg[10] = 00000000000000000000000000000000000000000000000000000000000001010  reg[11] = 00000000000000000000000000000000000000000000000000000000000001011
reg[12] = 00000000000000000000000000000000000000000000000000000000000001100  reg[13] = 00000000000000000000000000000000000000000000000000000000000000000  reg[14] = 00000000000000000000000000000000000000000000000000000000000000000
reg[15] = 00000000000000000000000000000000000000000000000000000000000000000  reg[16] = 00000000000000000000000000000000000000000000000000000000000000000  reg[17] = 01111111111111111111111111111111111111111111111111111111111111111
reg[18] = 10000000000000000000000000000000000000000000000000000000000000000  reg[19] = 11111111111111111111111111111111111111111111111111111111111111111  reg[20] = 00000000000000000000000000000000000000000000000000000000000000000
reg[21] = 00000000000000000000000000000000000000000000000000000000000000000  reg[22] = 00000000000000000000000000000000000000000000000000000000000000000  reg[23] = 00000000000000000000000000000000000000000000000000000000000000000
reg[24] = 00000000000000000000000000000000000000000000000000000000000000000  reg[25] = 00000000000000000000000000000000000000000000000000000000000000000  reg[26] = 00000000000000000000000000000000000000000000000000000000000000000
reg[27] = 00000000000000000000000000000000000000000000000000000000000000000  reg[28] = 00000000000000000000000000000000000000000000000000000000000000000  reg[29] = 00000000000000000000000000000000000000000000000000000000000000000
reg[30] = 10101010101010101010101010101010101010101010101010101010101010101  reg[31] = 11111111111111111111111111111111111111111111111111111111111111100
```

$$x_3 = x_3 + x_3$$

```
Time: 70000
Current PC =                   12 instruction = 003181b3
rs1= 3, rs2= 3, rd= 3
ReadData1 = 00000000000000000000000000000000000000000000000000000000000000011, ReadData2 = 00000000000000000000000000000000000000000000000000000000000000011
ALU Output (valE) = 00000000000000000000000000000000000000000000000000000000000000110, Zero Flag = 0
Memory Read Data = 00000000000000000000000000000000000000000000000000000000000000000, Write Data = 00000000000000000000000000000000000000000000000000000000000000110
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=0
reg[ 0] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 00000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 00000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 00000000000000000000000000000000000000000000000000000000000000011  reg[ 4] = 00000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 00000000000000000000000000000000000000000000000000000000000001001
reg[ 6] = 00000000000000000000000000000000000000000000000000000000000001100  reg[ 7] = 00000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 00000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 00000000000000000000000000000000000000000000000000000000000000000  reg[10] = 00000000000000000000000000000000000000000000000000000000000001010  reg[11] = 00000000000000000000000000000000000000000000000000000000000001011
reg[12] = 00000000000000000000000000000000000000000000000000000000000001100  reg[13] = 00000000000000000000000000000000000000000000000000000000000000000  reg[14] = 00000000000000000000000000000000000000000000000000000000000000000
reg[15] = 00000000000000000000000000000000000000000000000000000000000000000  reg[16] = 00000000000000000000000000000000000000000000000000000000000000000  reg[17] = 01111111111111111111111111111111111111111111111111111111111111111
reg[18] = 10000000000000000000000000000000000000000000000000000000000000000  reg[19] = 11111111111111111111111111111111111111111111111111111111111111111  reg[20] = 00000000000000000000000000000000000000000000000000000000000000000
reg[21] = 00000000000000000000000000000000000000000000000000000000000000000  reg[22] = 00000000000000000000000000000000000000000000000000000000000000000  reg[23] = 00000000000000000000000000000000000000000000000000000000000000000
reg[24] = 00000000000000000000000000000000000000000000000000000000000000000  reg[25] = 00000000000000000000000000000000000000000000000000000000000000000  reg[26] = 00000000000000000000000000000000000000000000000000000000000000000
reg[27] = 00000000000000000000000000000000000000000000000000000000000000000  reg[28] = 00000000000000000000000000000000000000000000000000000000000000000  reg[29] = 00000000000000000000000000000000000000000000000000000000000000000
reg[30] = 10101010101010101010101010101010101010101010101010101010101010101  reg[31] = 11111111111111111111111111111111111111111111111111111111111111100
```

$$x_3 = x_3 + x_3$$

```
Time: 70000
Current PC =                  12 instruction = 003181b3
rs1= 3, rs2= 3, rd= 3
ReadData1 = 00000000000000000000000000000000000000000000000000000000000000011, ReadData2 = 00000000000000000000000000000000000000000000000000000000000000011
ALU Output (valE) = 00000000000000000000000000000000000000000000000000000000000000110, Zero Flag = 0
Memory Read Data = 00000000000000000000000000000000000000000000000000000000000000000, Write Data = 00000000000000000000000000000000000000000000000000000000000000110
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=0
reg[ 0] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 00000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 00000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 00000000000000000000000000000000000000000000000000000000000000011  reg[ 4] = 00000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 00000000000000000000000000000000000000000000000000000000000001001
reg[ 6] = 00000000000000000000000000000000000000000000000000000000000001100  reg[ 7] = 00000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 00000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 00000000000000000000000000000000000000000000000000000000000000000  reg[10] = 00000000000000000000000000000000000000000000000000000000000001010  reg[11] = 00000000000000000000000000000000000000000000000000000000000001011
reg[12] = 00000000000000000000000000000000000000000000000000000000000001100  reg[13] = 00000000000000000000000000000000000000000000000000000000000000000  reg[14] = 00000000000000000000000000000000000000000000000000000000000000000
reg[15] = 00000000000000000000000000000000000000000000000000000000000000000  reg[16] = 00000000000000000000000000000000000000000000000000000000000000000  reg[17] = 01111111111111111111111111111111111111111111111111111111111111111
reg[18] = 10000000000000000000000000000000000000000000000000000000000000000  reg[19] = 11111111111111111111111111111111111111111111111111111111111111111  reg[20] = 00000000000000000000000000000000000000000000000000000000000000000
reg[21] = 00000000000000000000000000000000000000000000000000000000000000000  reg[22] = 00000000000000000000000000000000000000000000000000000000000000000  reg[23] = 00000000000000000000000000000000000000000000000000000000000000000
reg[24] = 00000000000000000000000000000000000000000000000000000000000000000  reg[25] = 00000000000000000000000000000000000000000000000000000000000000000  reg[26] = 00000000000000000000000000000000000000000000000000000000000000000
reg[27] = 00000000000000000000000000000000000000000000000000000000000000000  reg[28] = 00000000000000000000000000000000000000000000000000000000000000000  reg[29] = 00000000000000000000000000000000000000000000000000000000000000000
reg[30] = 10101010101010101010101010101010101010101010101010101010101010100  reg[31] = 11111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
Time: 90000
Current PC =                  16 instruction = 408384b3
rs1= 7, rs2= 8, rd= 9
ReadData1 = 00000000000000000000000000000000000000000000000000000000000000111, ReadData2 = 00000000000000000000000000000000000000000000000000000000000001000
ALU Output (valE) = 11111111111111111111111111111111111111111111111111111111111111111, Zero Flag = 0
Memory Read Data = 00000000000000000000000000000000000000000000000000000000000000000, Write Data = 11111111111111111111111111111111111111111111111111111111111111111
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=0
reg[ 0] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 00000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 00000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 00000000000000000000000000000000000000000000000000000000000000110  reg[ 4] = 00000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 00000000000000000000000000000000000000000000000000000000000001001
reg[ 6] = 00000000000000000000000000000000000000000000000000000000000001100  reg[ 7] = 00000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 00000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 00000000000000000000000000000000000000000000000000000000000000000  reg[10] = 00000000000000000000000000000000000000000000000000000000000001010  reg[11] = 00000000000000000000000000000000000000000000000000000000000001011
reg[12] = 00000000000000000000000000000000000000000000000000000000000001100  reg[13] = 00000000000000000000000000000000000000000000000000000000000000000  reg[14] = 00000000000000000000000000000000000000000000000000000000000000000
reg[15] = 00000000000000000000000000000000000000000000000000000000000000000  reg[16] = 00000000000000000000000000000000000000000000000000000000000000000  reg[17] = 01111111111111111111111111111111111111111111111111111111111111111
reg[18] = 10000000000000000000000000000000000000000000000000000000000000000  reg[19] = 11111111111111111111111111111111111111111111111111111111111111111  reg[20] = 00000000000000000000000000000000000000000000000000000000000000000
reg[21] = 00000000000000000000000000000000000000000000000000000000000000000  reg[22] = 00000000000000000000000000000000000000000000000000000000000000000  reg[23] = 00000000000000000000000000000000000000000000000000000000000000000
reg[24] = 00000000000000000000000000000000000000000000000000000000000000000  reg[25] = 00000000000000000000000000000000000000000000000000000000000000000  reg[26] = 00000000000000000000000000000000000000000000000000000000000000000
reg[27] = 00000000000000000000000000000000000000000000000000000000000000000  reg[28] = 00000000000000000000000000000000000000000000000000000000000000000  reg[29] = 00000000000000000000000000000000000000000000000000000000000000000
reg[30] = 10101010101010101010101010101010101010101010101010101010101010100  reg[31] = 11111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------------------------------------------------------
```



SUB Operation

```
408384B3  // sub x9, x8, x7
40B505B3  // sub x11, x10, x11
40B48633  // sub x12, x9, x11
409484b3  // sub x9 ,x9, x9
```

$x_9 = x_7 - x_8$

```
Time: 10000
Current PC =                   0 instruction = 408384b3
rs1= 7, rs2= 8, rd= 9
ReadData1 = 00000000000000000000000000000000000000000000000000000000000000111, ReadData2 = 00000000000000000000000000000000000000000000000000000000000001000
ALU Output (valE) = 11111111111111111111111111111111111111111111111111111111111111111, Zero Flag = 0
Memory Read Data = 00000000000000000000000000000000000000000000000000000000000000000, Write Data = 11111111111111111111111111111111111111111111111111111111111111111
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=0
reg[ 0] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 00000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 00000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 00000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 00000000000000000000000000000000000000000000000000000000000001010
reg[ 6] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 00000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 00000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 00000000000000000000000000000000000000000000000000000000000000000  reg[10] = 00000000000000000000000000000000000000000000000000000000000001010  reg[11] = 00000000000000000000000000000000000000000000000000000000000001011
reg[12] = 00000000000000000000000000000000000000000000000000000000000001100  reg[13] = 00000000000000000000000000000000000000000000000000000000000000000  reg[14] = 00000000000000000000000000000000000000000000000000000000000000000
reg[15] = 00000000000000000000000000000000000000000000000000000000000000000  reg[16] = 00000000000000000000000000000000000000000000000000000000000000000  reg[17] = 01111111111111111111111111111111111111111111111111111111111111111
reg[18] = 10000000000000000000000000000000000000000000000000000000000000000  reg[19] = 11111111111111111111111111111111111111111111111111111111111111111  reg[20] = 00000000000000000000000000000000000000000000000000000000000000000
reg[21] = 00000000000000000000000000000000000000000000000000000000000000000  reg[22] = 00000000000000000000000000000000000000000000000000000000000000000  reg[23] = 00000000000000000000000000000000000000000000000000000000000000000
reg[24] = 00000000000000000000000000000000000000000000000000000000000000000  reg[25] = 00000000000000000000000000000000000000000000000000000000000000000  reg[26] = 00000000000000000000000000000000000000000000000000000000000000000
reg[27] = 00000000000000000000000000000000000000000000000000000000000000000  reg[28] = 00000000000000000000000000000000000000000000000000000000000000000  reg[29] = 00000000000000000000000000000000000000000000000000000000000000000
reg[30] = 10101010101010101010101010101010101010101010101010101010101010100  reg[31] = 11111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
Time: 30000
Current PC =                   4 instruction = 40b505b3
rs1=10, rs2=11, rd=11
ReadData1 = 00000000000000000000000000000000000000000000000000000000000001010, ReadData2 = 00000000000000000000000000000000000000000000000000000000000001011
ALU Output (valE) = 11111111111111111111111111111111111111111111111111111111111111111, Zero Flag = 0
Memory Read Data = 00000000000000000000000000000000000000000000000000000000000000000, Write Data = 11111111111111111111111111111111111111111111111111111111111111111
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=0
reg[ 0] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 00000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 00000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 00000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 00000000000000000000000000000000000000000000000000000000000001010
reg[ 6] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 00000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 00000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 11111111111111111111111111111111111111111111111111111111111111111  reg[10] = 00000000000000000000000000000000000000000000000000000000000001010  reg[11] = 00000000000000000000000000000000000000000000000000000000000001011
reg[12] = 00000000000000000000000000000000000000000000000000000000000001100  reg[13] = 00000000000000000000000000000000000000000000000000000000000000000  reg[14] = 00000000000000000000000000000000000000000000000000000000000000000
reg[15] = 00000000000000000000000000000000000000000000000000000000000000000  reg[16] = 00000000000000000000000000000000000000000000000000000000000000000  reg[17] = 01111111111111111111111111111111111111111111111111111111111111111
reg[18] = 10000000000000000000000000000000000000000000000000000000000000000  reg[19] = 11111111111111111111111111111111111111111111111111111111111111111  reg[20] = 00000000000000000000000000000000000000000000000000000000000000000
reg[21] = 00000000000000000000000000000000000000000000000000000000000000000  reg[22] = 00000000000000000000000000000000000000000000000000000000000000000  reg[23] = 00000000000000000000000000000000000000000000000000000000000000000
reg[24] = 00000000000000000000000000000000000000000000000000000000000000000  reg[25] = 00000000000000000000000000000000000000000000000000000000000000000  reg[26] = 00000000000000000000000000000000000000000000000000000000000000000
reg[27] = 00000000000000000000000000000000000000000000000000000000000000000  reg[28] = 00000000000000000000000000000000000000000000000000000000000000000  reg[29] = 00000000000000000000000000000000000000000000000000000000000000000
reg[30] = 10101010101010101010101010101010101010101010101010101010101010100  reg[31] = 11111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------------------------------------------------------
```

$x_{11} = x_{10} - x_{11}$

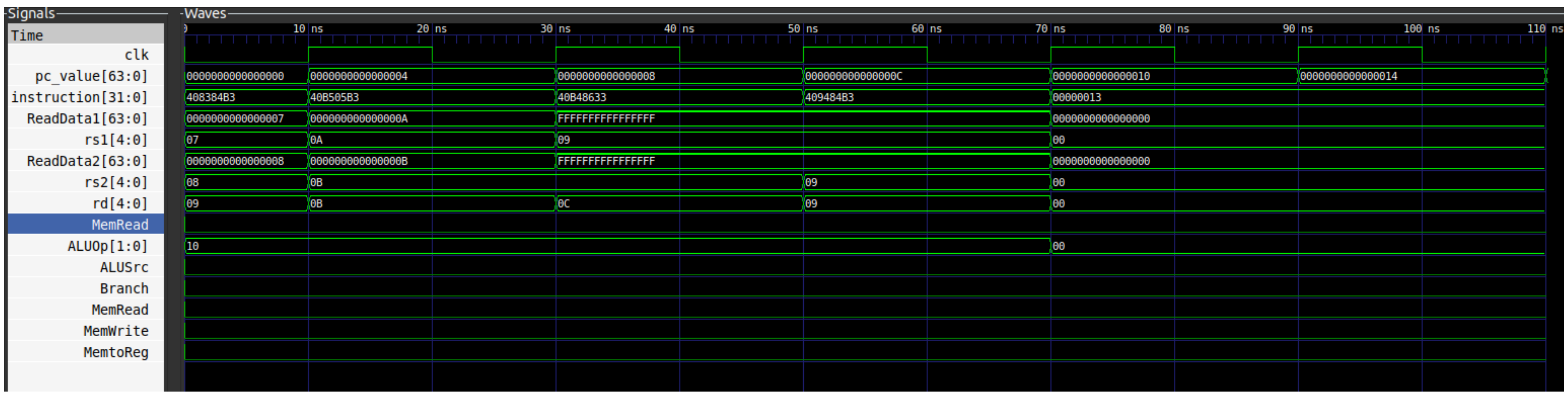
```
--------------------------------------------------------------------------------
Time: 30000
Current PC =                    4 instruction = 40b505b3
rs1=10, rs2=11, rd=11
ReadData1 = 0000000000000000000000000000000000000000000000000000000000001010, ReadData2 = 0000000000000000000000000000000000000000000000000000000000001011
ALU Output (valE) = 1111111111111111111111111111111111111111111111111111111111111111, Zero Flag = 0
Memory Read Data = 0000000000000000000000000000000000000000000000000000000000000000, Write Data = 1111111111111111111111111111111111111111111111111111111111111111
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=0
reg[ 0] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 0000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 0000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 0000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 0000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 0000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 0000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 1111111111111111111111111111111111111111111111111111111111111111  reg[10] = 0000000000000000000000000000000000000000000000000000000000001010  reg[11] = 0000000000000000000000000000000000000000000000000000000000001011
reg[12] = 0000000000000000000000000000000000000000000000000000000000001100  reg[13] = 0000000000000000000000000000000000000000000000000000000000000000  reg[14] = 0000000000000000000000000000000000000000000000000000000000000000
reg[15] = 0000000000000000000000000000000000000000000000000000000000000000  reg[16] = 0000000000000000000000000000000000000000000000000000000000000000  reg[17] = 0111111111111111111111111111111111111111111111111111111111111111
reg[18] = 1000000000000000000000000000000000000000000000000000000000000000  reg[19] = 1111111111111111111111111111111111111111111111111111111111111111  reg[20] = 0000000000000000000000000000000000000000000000000000000000000000
reg[21] = 0000000000000000000000000000000000000000000000000000000000000000  reg[22] = 0000000000000000000000000000000000000000000000000000000000000000  reg[23] = 0000000000000000000000000000000000000000000000000000000000000000
reg[24] = 0000000000000000000000000000000000000000000000000000000000000000  reg[25] = 0000000000000000000000000000000000000000000000000000000000000000  reg[26] = 0000000000000000000000000000000000000000000000000000000000000000
reg[27] = 0000000000000000000000000000000000000000000000000000000000000000  reg[28] = 0000000000000000000000000000000000000000000000000000000000000000  reg[29] = 0000000000000000000000000000000000000000000000000000000000000000
reg[30] = 1010101010101010101010101010101010101010101010101010101010101010  reg[31] = 1111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Time: 50000
Current PC =                    8 instruction = 40b48633
rs1= 9, rs2=11, rd=12
ReadData1 = 1111111111111111111111111111111111111111111111111111111111111111, ReadData2 = 1111111111111111111111111111111111111111111111111111111111111111
ALU Output (valE) = 0000000000000000000000000000000000000000000000000000000000000000, Zero Flag = 1
Memory Read Data = 0000000000000000000000000000000000000000000000000000000000000000, Write Data = 0000000000000000000000000000000000000000000000000000000000000000
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=1, overflow=0
reg[ 0] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 0000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 0000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 0000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 0000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 0000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 0000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 1111111111111111111111111111111111111111111111111111111111111111  reg[10] = 0000000000000000000000000000000000000000000000000000000000001010  reg[11] = 1111111111111111111111111111111111111111111111111111111111111111
reg[12] = 0000000000000000000000000000000000000000000000000000000000001100  reg[13] = 0000000000000000000000000000000000000000000000000000000000000000  reg[14] = 0000000000000000000000000000000000000000000000000000000000000000
reg[15] = 0000000000000000000000000000000000000000000000000000000000000000  reg[16] = 0000000000000000000000000000000000000000000000000000000000000000  reg[17] = 0111111111111111111111111111111111111111111111111111111111111111
reg[18] = 1000000000000000000000000000000000000000000000000000000000000000  reg[19] = 1111111111111111111111111111111111111111111111111111111111111111  reg[20] = 0000000000000000000000000000000000000000000000000000000000000000
reg[21] = 0000000000000000000000000000000000000000000000000000000000000000  reg[22] = 0000000000000000000000000000000000000000000000000000000000000000  reg[23] = 0000000000000000000000000000000000000000000000000000000000000000
reg[24] = 0000000000000000000000000000000000000000000000000000000000000000  reg[25] = 0000000000000000000000000000000000000000000000000000000000000000  reg[26] = 0000000000000000000000000000000000000000000000000000000000000000
reg[27] = 0000000000000000000000000000000000000000000000000000000000000000  reg[28] = 0000000000000000000000000000000000000000000000000000000000000000  reg[29] = 0000000000000000000000000000000000000000000000000000000000000000
reg[30] = 1010101010101010101010101010101010101010101010101010101010101010  reg[31] = 1111111111111111111111111111111111111111111111111111111111111100
```

$x_{12} = x_9 - x_{11}$

```
Time: 50000
Current PC =                    8 instruction = 40b48633
rs1= 9, rs2=11, rd=12
ReadData1 = 1111111111111111111111111111111111111111111111111111111111111111, ReadData2 = 1111111111111111111111111111111111111111111111111111111111111111
ALU Output (valE) = 0000000000000000000000000000000000000000000000000000000000000000, Zero Flag = 1
Memory Read Data = 0000000000000000000000000000000000000000000000000000000000000000, Write Data = 0000000000000000000000000000000000000000000000000000000000000000
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=1, overflow=0
reg[ 0] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 0000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 0000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 0000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 0000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 0000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 0000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 1111111111111111111111111111111111111111111111111111111111111111  reg[10] = 0000000000000000000000000000000000000000000000000000000000001010  reg[11] = 1111111111111111111111111111111111111111111111111111111111111111
reg[12] = 0000000000000000000000000000000000000000000000000000000000001100  reg[13] = 0000000000000000000000000000000000000000000000000000000000000000  reg[14] = 0000000000000000000000000000000000000000000000000000000000000000
reg[15] = 0000000000000000000000000000000000000000000000000000000000000000  reg[16] = 0000000000000000000000000000000000000000000000000000000000000000  reg[17] = 0111111111111111111111111111111111111111111111111111111111111111
reg[18] = 1000000000000000000000000000000000000000000000000000000000000000  reg[19] = 1111111111111111111111111111111111111111111111111111111111111111  reg[20] = 0000000000000000000000000000000000000000000000000000000000000000
reg[21] = 0000000000000000000000000000000000000000000000000000000000000000  reg[22] = 0000000000000000000000000000000000000000000000000000000000000000  reg[23] = 0000000000000000000000000000000000000000000000000000000000000000
reg[24] = 0000000000000000000000000000000000000000000000000000000000000000  reg[25] = 0000000000000000000000000000000000000000000000000000000000000000  reg[26] = 0000000000000000000000000000000000000000000000000000000000000000
reg[27] = 0000000000000000000000000000000000000000000000000000000000000000  reg[28] = 0000000000000000000000000000000000000000000000000000000000000000  reg[29] = 0000000000000000000000000000000000000000000000000000000000000000
reg[30] = 1010101010101010101010101010101010101010101010101010101010101010  reg[31] = 1111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Time: 70000
Current PC =                   12 instruction = 409484b3
rs1= 9, rs2= 9, rd= 9
ReadData1 = 1111111111111111111111111111111111111111111111111111111111111111, ReadData2 = 1111111111111111111111111111111111111111111111111111111111111111
ALU Output (valE) = 0000000000000000000000000000000000000000000000000000000000000000, Zero Flag = 1
Memory Read Data = 0000000000000000000000000000000000000000000000000000000000000000, Write Data = 0000000000000000000000000000000000000000000000000000000000000000
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=1, overflow=0
reg[ 0] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 0000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 0000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 0000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 0000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 0000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 0000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 1111111111111111111111111111111111111111111111111111111111111111  reg[10] = 0000000000000000000000000000000000000000000000000000000000001010  reg[11] = 1111111111111111111111111111111111111111111111111111111111111111
reg[12] = 0000000000000000000000000000000000000000000000000000000000000000  reg[13] = 0000000000000000000000000000000000000000000000000000000000000000  reg[14] = 0000000000000000000000000000000000000000000000000000000000000000
reg[15] = 0000000000000000000000000000000000000000000000000000000000000000  reg[16] = 0000000000000000000000000000000000000000000000000000000000000000  reg[17] = 0111111111111111111111111111111111111111111111111111111111111111
reg[18] = 1000000000000000000000000000000000000000000000000000000000000000  reg[19] = 1111111111111111111111111111111111111111111111111111111111111111  reg[20] = 0000000000000000000000000000000000000000000000000000000000000000
reg[21] = 0000000000000000000000000000000000000000000000000000000000000000  reg[22] = 0000000000000000000000000000000000000000000000000000000000000000  reg[23] = 0000000000000000000000000000000000000000000000000000000000000000
reg[24] = 0000000000000000000000000000000000000000000000000000000000000000  reg[25] = 0000000000000000000000000000000000000000000000000000000000000000  reg[26] = 0000000000000000000000000000000000000000000000000000000000000000
reg[27] = 0000000000000000000000000000000000000000000000000000000000000000  reg[28] = 0000000000000000000000000000000000000000000000000000000000000000  reg[29] = 0000000000000000000000000000000000000000000000000000000000000000
reg[30] = 1010101010101010101010101010101010101010101010101010101010101010  reg[31] = 1111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------
```

$x_{12} = x_{12} - x_{12}$

```
--------------------------------------------------------------------------------
Time: 70000
Current PC =                   12 instruction = 409484b3
rs1= 9, rs2= 9, rd= 9
ReadData1 = 1111111111111111111111111111111111111111111111111111111111111111, ReadData2 = 1111111111111111111111111111111111111111111111111111111111111111
ALU Output (valE) = 0000000000000000000000000000000000000000000000000000000000000000, Zero Flag = 1
Memory Read Data = 0000000000000000000000000000000000000000000000000000000000000000, Write Data = 0000000000000000000000000000000000000000000000000000000000000000
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=1, overflow=0
reg[ 0] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 0000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 0000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 0000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 0000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 0000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 0000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 1111111111111111111111111111111111111111111111111111111111111111  reg[10] = 0000000000000000000000000000000000000000000000000000000000001010  reg[11] = 1111111111111111111111111111111111111111111111111111111111111111
reg[12] = 0000000000000000000000000000000000000000000000000000000000000000  reg[13] = 0000000000000000000000000000000000000000000000000000000000000000  reg[14] = 0000000000000000000000000000000000000000000000000000000000000000
reg[15] = 0000000000000000000000000000000000000000000000000000000000000000  reg[16] = 0000000000000000000000000000000000000000000000000000000000000000  reg[17] = 0111111111111111111111111111111111111111111111111111111111111111
reg[18] = 1000000000000000000000000000000000000000000000000000000000000000  reg[19] = 1111111111111111111111111111111111111111111111111111111111111111  reg[20] = 0000000000000000000000000000000000000000000000000000000000000000
reg[21] = 0000000000000000000000000000000000000000000000000000000000000000  reg[22] = 0000000000000000000000000000000000000000000000000000000000000000  reg[23] = 0000000000000000000000000000000000000000000000000000000000000000
reg[24] = 0000000000000000000000000000000000000000000000000000000000000000  reg[25] = 0000000000000000000000000000000000000000000000000000000000000000  reg[26] = 0000000000000000000000000000000000000000000000000000000000000000
reg[27] = 0000000000000000000000000000000000000000000000000000000000000000  reg[28] = 0000000000000000000000000000000000000000000000000000000000000000  reg[29] = 0000000000000000000000000000000000000000000000000000000000000000
reg[30] = 1010101010101010101010101010101010101010101010101010101010101010  reg[31] = 1111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Time: 90000
Current PC =                   16 instruction = 00000013
rs1= 0, rs2= 0, rd= 0
ReadData1 = 0000000000000000000000000000000000000000000000000000000000000000, ReadData2 = 0000000000000000000000000000000000000000000000000000000000000000
ALU Output (valE) = 0000000000000000000000000000000000000000000000000000000000000000, Zero Flag = 1
Memory Read Data = 0000000000000000000000000000000000000000000000000000000000000000, Write Data = 0000000000000000000000000000000000000000000000000000000000000000
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=0, ALUOp=00, carry=0, overflow=0
reg[ 0] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 0000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 0000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 0000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 0000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 0000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 0000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 0000000000000000000000000000000000000000000000000000000000000000  reg[10] = 0000000000000000000000000000000000000000000000000000000000001010  reg[11] = 1111111111111111111111111111111111111111111111111111111111111111
reg[12] = 0000000000000000000000000000000000000000000000000000000000000000  reg[13] = 0000000000000000000000000000000000000000000000000000000000000000  reg[14] = 0000000000000000000000000000000000000000000000000000000000000000
reg[15] = 0000000000000000000000000000000000000000000000000000000000000000  reg[16] = 0000000000000000000000000000000000000000000000000000000000000000  reg[17] = 0111111111111111111111111111111111111111111111111111111111111111
reg[18] = 1000000000000000000000000000000000000000000000000000000000000000  reg[19] = 1111111111111111111111111111111111111111111111111111111111111111  reg[20] = 0000000000000000000000000000000000000000000000000000000000000000
reg[21] = 0000000000000000000000000000000000000000000000000000000000000000  reg[22] = 0000000000000000000000000000000000000000000000000000000000000000  reg[23] = 0000000000000000000000000000000000000000000000000000000000000000
reg[24] = 0000000000000000000000000000000000000000000000000000000000000000  reg[25] = 0000000000000000000000000000000000000000000000000000000000000000  reg[26] = 0000000000000000000000000000000000000000000000000000000000000000
reg[27] = 0000000000000000000000000000000000000000000000000000000000000000  reg[28] = 0000000000000000000000000000000000000000000000000000000000000000  reg[29] = 0000000000000000000000000000000000000000000000000000000000000000
reg[30] = 1010101010101010101010101010101010101010101010101010101010101010  reg[31] = 1111111111111111111111111111111111111111111111111111111111111100
```



Signals
Waves
Time
clk
pc_value[63:0]
instruction[31:0]
ReadData1[63:0]
rs1[4:0]
ReadData2[63:0]
rs2[4:0]
rd[4:0]
MemRead
ALUOp[1:0]
ALUSrc
Branch
MemRead
MemWrite
MemtoReg
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns
110 ns
0000000000000000
0000000000000004
0000000000000008
000000000000000C
0000000000000010
0000000000000014
408384B3
40B505B3
40B48633
409484B3
00000013
0000000000000007
000000000000000A
FFFFFFFFFFFFFFFF
0000000000000000
07
0A
09
00
0000000000000008
000000000000000B
FFFFFFFFFFFFFFFF
0000000000000000
08
0B
09
00
09
0B
0C
09
00
10
00

AND Operation

01ff7eb3 //AND x29 ×30 ×31
01fefe33 //AND x28 ×29 ×31
01de7db3 //AND x27 ×28 ×29

$x_{29} = x_{30}.x_{31}$

```
--------------------------------------------------------------------------------------------------------------------------------
Time: 10000
Current PC =                    0 instruction = 01ff7eb3
rs1=30, rs2=31, rd=29
ReadData1 = 10101010101010101010101010101010101010101010101010101010101010, ReadData2 = 11111111111111111111111111111111111111111111111111111111111111100
ALU Output (valE) = 10101010101010101010101010101010101010101010101010101010101000, Zero Flag = 0
Memory Read Data = 00000000000000000000000000000000000000000000000000000000000000000, Write Data = 10101010101010101010101010101010101010101010101010101010101000
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=0
reg[ 0] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 00000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 00000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 00000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 00000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 00000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 00000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 00000000000000000000000000000000000000000000000000000000000000000  reg[10] = 00000000000000000000000000000000000000000000000000000000000001010  reg[11] = 00000000000000000000000000000000000000000000000000000000000001011
reg[12] = 00000000000000000000000000000000000000000000000000000000000001100  reg[13] = 00000000000000000000000000000000000000000000000000000000000000000  reg[14] = 00000000000000000000000000000000000000000000000000000000000000000
reg[15] = 00000000000000000000000000000000000000000000000000000000000000000  reg[16] = 00000000000000000000000000000000000000000000000000000000000000000  reg[17] = 01111111111111111111111111111111111111111111111111111111111111111
reg[18] = 10000000000000000000000000000000000000000000000000000000000000000  reg[19] = 11111111111111111111111111111111111111111111111111111111111111111  reg[20] = 00000000000000000000000000000000000000000000000000000000000000000
reg[21] = 00000000000000000000000000000000000000000000000000000000000000000  reg[22] = 00000000000000000000000000000000000000000000000000000000000000000  reg[23] = 00000000000000000000000000000000000000000000000000000000000000000
reg[24] = 00000000000000000000000000000000000000000000000000000000000000000  reg[25] = 00000000000000000000000000000000000000000000000000000000000000000  reg[26] = 00000000000000000000000000000000000000000000000000000000000000000
reg[27] = 00000000000000000000000000000000000000000000000000000000000000000  reg[28] = 00000000000000000000000000000000000000000000000000000000000000000  reg[29] = 00000000000000000000000000000000000000000000000000000000000000000
reg[30] = 10101010101010101010101010101010101010101010101010101010101010101  reg[31] = 11111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
Time: 30000
Current PC =                    4 instruction = 01fefe33
rs1=29, rs2=31, rd=28
ReadData1 = 10101010101010101010101010101010101010101010101010101010101010101000, ReadData2 = 11111111111111111111111111111111111111111111111111111111111111100
ALU Output (valE) = 10101010101010101010101010101010101010101010101010101010101001000, Zero Flag = 0
Memory Read Data = 00000000000000000000000000000000000000000000000000000000000000000, Write Data = 10101010101010101010101010101010101010101010101010101010101001000
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=0
reg[ 0] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 00000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 00000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 00000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 00000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 00000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 00000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 00000000000000000000000000000000000000000000000000000000000000000  reg[10] = 00000000000000000000000000000000000000000000000000000000000001010  reg[11] = 00000000000000000000000000000000000000000000000000000000000001011
reg[12] = 00000000000000000000000000000000000000000000000000000000000001100  reg[13] = 00000000000000000000000000000000000000000000000000000000000000000  reg[14] = 00000000000000000000000000000000000000000000000000000000000000000
reg[15] = 00000000000000000000000000000000000000000000000000000000000000000  reg[16] = 00000000000000000000000000000000000000000000000000000000000000000  reg[17] = 01111111111111111111111111111111111111111111111111111111111111111
reg[18] = 10000000000000000000000000000000000000000000000000000000000000000  reg[19] = 11111111111111111111111111111111111111111111111111111111111111111  reg[20] = 00000000000000000000000000000000000000000000000000000000000000000
reg[21] = 00000000000000000000000000000000000000000000000000000000000000000  reg[22] = 00000000000000000000000000000000000000000000000000000000000000000  reg[23] = 00000000000000000000000000000000000000000000000000000000000000000
reg[24] = 00000000000000000000000000000000000000000000000000000000000000000  reg[25] = 00000000000000000000000000000000000000000000000000000000000000000  reg[26] = 00000000000000000000000000000000000000000000000000000000000000000
reg[27] = 00000000000000000000000000000000000000000000000000000000000000000  reg[28] = 00000000000000000000000000000000000000000000000000000000000000000  reg[29] = 10101010101010101010101010101010101010101010101010101010101001000
reg[30] = 10101010101010101010101010101010101010101010101010101010101010101  reg[31] = 11111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------------------------------------------------------
```

$x_{28} = x_{29}.x_{31}$

```
--------------------------------------------------------------------------------------------------------------------------------
Time: 30000
Current PC =                    4 instruction = 01fefe33
rs1=29, rs2=31, rd=28
ReadData1 = 10101010101010101010101010101010101010101010101010101010101010101000, ReadData2 = 11111111111111111111111111111111111111111111111111111111111111100
ALU Output (valE) = 10101010101010101010101010101010101010101010101010101010101001000, Zero Flag = 0
Memory Read Data = 00000000000000000000000000000000000000000000000000000000000000000, Write Data = 10101010101010101010101010101010101010101010101010101010101001000
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=0
reg[ 0] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 00000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 00000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 00000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 00000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 00000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 00000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 00000000000000000000000000000000000000000000000000000000000000000  reg[10] = 00000000000000000000000000000000000000000000000000000000000001010  reg[11] = 00000000000000000000000000000000000000000000000000000000000001011
reg[12] = 00000000000000000000000000000000000000000000000000000000000001100  reg[13] = 00000000000000000000000000000000000000000000000000000000000000000  reg[14] = 00000000000000000000000000000000000000000000000000000000000000000
reg[15] = 00000000000000000000000000000000000000000000000000000000000000000  reg[16] = 00000000000000000000000000000000000000000000000000000000000000000  reg[17] = 01111111111111111111111111111111111111111111111111111111111111111
reg[18] = 10000000000000000000000000000000000000000000000000000000000000000  reg[19] = 11111111111111111111111111111111111111111111111111111111111111111  reg[20] = 00000000000000000000000000000000000000000000000000000000000000000
reg[21] = 00000000000000000000000000000000000000000000000000000000000000000  reg[22] = 00000000000000000000000000000000000000000000000000000000000000000  reg[23] = 00000000000000000000000000000000000000000000000000000000000000000
reg[24] = 00000000000000000000000000000000000000000000000000000000000000000  reg[25] = 00000000000000000000000000000000000000000000000000000000000000000  reg[26] = 00000000000000000000000000000000000000000000000000000000000000000
reg[27] = 00000000000000000000000000000000000000000000000000000000000000000  reg[28] = 00000000000000000000000000000000000000000000000000000000000000000  reg[29] = 10101010101010101010101010101010101010101010101010101010101001000
reg[30] = 10101010101010101010101010101010101010101010101010101010101010101  reg[31] = 11111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
Time: 50000
Current PC =                    8 instruction = 01de7db3
rs1=28, rs2=29, rd=27
ReadData1 = 10101010101010101010101010101010101010101010101010101010101001000, ReadData2 = 10101010101010101010101010101010101010101010101010101010101001000
ALU Output (valE) = 10101010101010101010101010101010101010101010101010101010101001000, Zero Flag = 0
Memory Read Data = 00000000000000000000000000000000000000000000000000000000000000000, Write Data = 10101010101010101010101010101010101010101010101010101010101001000
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=0
reg[ 0] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 00000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 00000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 00000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 00000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 00000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 00000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 00000000000000000000000000000000000000000000000000000000000000000  reg[10] = 00000000000000000000000000000000000000000000000000000000000001010  reg[11] = 00000000000000000000000000000000000000000000000000000000000001011
reg[12] = 00000000000000000000000000000000000000000000000000000000000001100  reg[13] = 00000000000000000000000000000000000000000000000000000000000000000  reg[14] = 00000000000000000000000000000000000000000000000000000000000000000
reg[15] = 00000000000000000000000000000000000000000000000000000000000000000  reg[16] = 00000000000000000000000000000000000000000000000000000000000000000  reg[17] = 01111111111111111111111111111111111111111111111111111111111111111
reg[18] = 10000000000000000000000000000000000000000000000000000000000000000  reg[19] = 11111111111111111111111111111111111111111111111111111111111111111  reg[20] = 00000000000000000000000000000000000000000000000000000000000000000
reg[21] = 00000000000000000000000000000000000000000000000000000000000000000  reg[22] = 00000000000000000000000000000000000000000000000000000000000000000  reg[23] = 00000000000000000000000000000000000000000000000000000000000000000
reg[24] = 00000000000000000000000000000000000000000000000000000000000000000  reg[25] = 00000000000000000000000000000000000000000000000000000000000000000  reg[26] = 00000000000000000000000000000000000000000000000000000000000000000
reg[27] = 00000000000000000000000000000000000000000000000000000000000000000  reg[28] = 10101010101010101010101010101010101010101010101010101010101001000  reg[29] = 10101010101010101010101010101010101010101010101010101010101001000
reg[30] = 10101010101010101010101010101010101010101010101010101010101010101  reg[31] = 11111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------------------------------------------------------
```

$x_{27} = x_{28}.x_{29}$

```
--------------------------------------------------------------------------------------------------------------------------------
Time: 50000
Current PC =                    8 instruction = 01de7db3
rs1=28, rs2=29, rd=27
ReadData1 = 10101010101010101010101010101010101010101010101010101010101001000, ReadData2 = 10101010101010101010101010101010101010101010101010101010101001000
ALU Output (valE) = 10101010101010101010101010101010101010101010101010101010101001000, Zero Flag = 0
Memory Read Data = 00000000000000000000000000000000000000000000000000000000000000000, Write Data = 10101010101010101010101010101010101010101010101010101010101001000
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=0
reg[ 0] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 00000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 00000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 00000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 00000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 00000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 00000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 00000000000000000000000000000000000000000000000000000000000000000  reg[10] = 00000000000000000000000000000000000000000000000000000000000001010  reg[11] = 00000000000000000000000000000000000000000000000000000000000001011
reg[12] = 00000000000000000000000000000000000000000000000000000000000001100  reg[13] = 00000000000000000000000000000000000000000000000000000000000000000  reg[14] = 00000000000000000000000000000000000000000000000000000000000000000
reg[15] = 00000000000000000000000000000000000000000000000000000000000000000  reg[16] = 00000000000000000000000000000000000000000000000000000000000000000  reg[17] = 01111111111111111111111111111111111111111111111111111111111111111
reg[18] = 10000000000000000000000000000000000000000000000000000000000000000  reg[19] = 11111111111111111111111111111111111111111111111111111111111111111  reg[20] = 00000000000000000000000000000000000000000000000000000000000000000
reg[21] = 00000000000000000000000000000000000000000000000000000000000000000  reg[22] = 00000000000000000000000000000000000000000000000000000000000000000  reg[23] = 00000000000000000000000000000000000000000000000000000000000000000
reg[24] = 00000000000000000000000000000000000000000000000000000000000000000  reg[25] = 00000000000000000000000000000000000000000000000000000000000000000  reg[26] = 00000000000000000000000000000000000000000000000000000000000000000
reg[27] = 00000000000000000000000000000000000000000000000000000000000000000  reg[28] = 10101010101010101010101010101010101010101010101010101010101001000  reg[29] = 10101010101010101010101010101010101010101010101010101010101001000
reg[30] = 10101010101010101010101010101010101010101010101010101010101010101  reg[31] = 11111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
Time: 70000
Current PC =                   12 instruction = 00000013
rs1= 0, rs2= 0, rd= 0
ReadData1 = 00000000000000000000000000000000000000000000000000000000000000000, ReadData2 = 00000000000000000000000000000000000000000000000000000000000000000
ALU Output (valE) = 00000000000000000000000000000000000000000000000000000000000000000, Zero Flag = 1
Memory Read Data = 00000000000000000000000000000000000000000000000000000000000000000, Write Data = 00000000000000000000000000000000000000000000000000000000000000000
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=0, ALUOp=00, carry=0, overflow=0
reg[ 0] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 00000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 00000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 00000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 00000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 00000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 00000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 00000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 00000000000000000000000000000000000000000000000000000000000000000  reg[10] = 00000000000000000000000000000000000000000000000000000000000001010  reg[11] = 00000000000000000000000000000000000000000000000000000000000001011
reg[12] = 00000000000000000000000000000000000000000000000000000000000001100  reg[13] = 00000000000000000000000000000000000000000000000000000000000000000  reg[14] = 00000000000000000000000000000000000000000000000000000000000000000
reg[15] = 00000000000000000000000000000000000000000000000000000000000000000  reg[16] = 00000000000000000000000000000000000000000000000000000000000000000  reg[17] = 01111111111111111111111111111111111111111111111111111111111111111
reg[18] = 10000000000000000000000000000000000000000000000000000000000000000  reg[19] = 11111111111111111111111111111111111111111111111111111111111111111  reg[20] = 00000000000000000000000000000000000000000000000000000000000000000
reg[21] = 00000000000000000000000000000000000000000000000000000000000000000  reg[22] = 00000000000000000000000000000000000000000000000000000000000000000  reg[23] = 00000000000000000000000000000000000000000000000000000000000000000
reg[24] = 00000000000000000000000000000000000000000000000000000000000000000  reg[25] = 00000000000000000000000000000000000000000000000000000000000000000  reg[26] = 00000000000000000000000000000000000000000000000000000000000000000
reg[27] = 10101010101010101010101010101010101010101010101010101010101001000  reg[28] = 10101010101010101010101010101010101010101010101010101010101001000  reg[29] = 10101010101010101010101010101010101010101010101010101010101001000
reg[30] = 10101010101010101010101010101010101010101010101010101010101010101  reg[31] = 11111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------------------------------------------------------
```

OR Operation

01ff6db3 // AND x27 ×30 ×31

```
Time: 10000
Current PC =                    0 instruction = 01ff6db3
rs1=30, rs2=31, rd=27
ReadData1 = 10101010101010101010101010101010101010101010101010101010101010, ReadData2 = 11111111111111111111111111111111111111111111111111111111111111100
ALU Output (valE) = 11111111111111111111111111111111111111111111111111111111111111110, Zero Flag = 0
Memory Read Data = 0000000000000000000000000000000000000000000000000000000000000000, Write Data = 11111111111111111111111111111111111111111111111111111111111111110
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=0
reg[ 0] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 0000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 0000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 0000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 0000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 0000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 0000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 0000000000000000000000000000000000000000000000000000000000000000  reg[10] = 0000000000000000000000000000000000000000000000000000000000001010  reg[11] = 0000000000000000000000000000000000000000000000000000000000001011
reg[12] = 0000000000000000000000000000000000000000000000000000000000001100  reg[13] = 0000000000000000000000000000000000000000000000000000000000000000  reg[14] = 0000000000000000000000000000000000000000000000000000000000000000
reg[15] = 0000000000000000000000000000000000000000000000000000000000000000  reg[16] = 0000000000000000000000000000000000000000000000000000000000000000  reg[17] = 0111111111111111111111111111111111111111111111111111111111111111
reg[18] = 1000000000000000000000000000000000000000000000000000000000000000  reg[19] = 1111111111111111111111111111111111111111111111111111111111111111  reg[20] = 0000000000000000000000000000000000000000000000000000000000000000
reg[21] = 0000000000000000000000000000000000000000000000000000000000000000  reg[22] = 0000000000000000000000000000000000000000000000000000000000000000  reg[23] = 0000000000000000000000000000000000000000000000000000000000000000
reg[24] = 0000000000000000000000000000000000000000000000000000000000000000  reg[25] = 0000000000000000000000000000000000000000000000000000000000000000  reg[26] = 0000000000000000000000000000000000000000000000000000000000000000
reg[27] = 0000000000000000000000000000000000000000000000000000000000000000  reg[28] = 0000000000000000000000000000000000000000000000000000000000000000  reg[29] = 0000000000000000000000000000000000000000000000000000000000000000
reg[30] = 1010101010101010101010101010101010101010101010101010101010101010  reg[31] = 1111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
Time: 30000
Current PC =                    4 instruction = 00000013
rs1= 0, rs2= 0, rd= 0
ReadData1 = 0000000000000000000000000000000000000000000000000000000000000000, ReadData2 = 0000000000000000000000000000000000000000000000000000000000000000
ALU Output (valE) = 0000000000000000000000000000000000000000000000000000000000000000, Zero Flag = 1
Memory Read Data = 0000000000000000000000000000000000000000000000000000000000000000, Write Data = 0000000000000000000000000000000000000000000000000000000000000000
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=0, ALUOp=00, carry=0, overflow=0
reg[ 0] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 0000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 0000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 0000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 0000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 0000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 0000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 0000000000000000000000000000000000000000000000000000000000000000  reg[10] = 0000000000000000000000000000000000000000000000000000000000001010  reg[11] = 0000000000000000000000000000000000000000000000000000000000001011
reg[12] = 0000000000000000000000000000000000000000000000000000000000001100  reg[13] = 0000000000000000000000000000000000000000000000000000000000000000  reg[14] = 0000000000000000000000000000000000000000000000000000000000000000
reg[15] = 0000000000000000000000000000000000000000000000000000000000000000  reg[16] = 0000000000000000000000000000000000000000000000000000000000000000  reg[17] = 0111111111111111111111111111111111111111111111111111111111111111
reg[18] = 1000000000000000000000000000000000000000000000000000000000000000  reg[19] = 1111111111111111111111111111111111111111111111111111111111111111  reg[20] = 0000000000000000000000000000000000000000000000000000000000000000
reg[21] = 0000000000000000000000000000000000000000000000000000000000000000  reg[22] = 0000000000000000000000000000000000000000000000000000000000000000  reg[23] = 0000000000000000000000000000000000000000000000000000000000000000
reg[24] = 0000000000000000000000000000000000000000000000000000000000000000  reg[25] = 0000000000000000000000000000000000000000000000000000000000000000  reg[26] = 0000000000000000000000000000000000000000000000000000000000000000
reg[27] = 0000000000000000000000000000000000000000000000000000000000000000  reg[28] = 0000000000000000000000000000000000000000000000000000000000000000  reg[29] = 0000000000000000000000000000000000000000000000000000000000000000
reg[30] = 1010101010101010101010101010101010101010101010101010101010101010  reg[31] = 1111111111111111111111111111111111111111111111111111111111111100
--------------------------------------------------------------------------------------------------------------------------------
```

LOAD Operation

00612183 // LB x3, 6(x2)

```
--------------------------------------------------------------------------------------------------------------------------------
Time: 10000
Current PC =                    0 instruction = 00612183
rs1= 2, rs2= 6, rd= 3
ReadData1 = 0000000000000000000000000000000000000000000000000000000000000010, ReadData2 = 0000000000000000000000000000000000000000000000000000000000000000
ALU Output (valE) = 0000000000000000000000000000000000000000000000000000000000001000, Zero Flag = 0
Memory Read Data = 0000000000000000000000000000000000000000000000000000000000001000, Write Data = 0000000000000000000000000000000000000000000000000000000000001000
Control Signals: Branch=0, MemRead=1, MemtoReg=1, MemWrite=0, ALUSrc=1, RegWrite=1, ALUOp=00, carry=0, overflow=0
reg[ 0] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 0000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 0000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 0000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 0000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 0000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 0000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 0000000000000000000000000000000000000000000000000000000000000000  reg[10] = 0000000000000000000000000000000000000000000000000000000000001010  reg[11] = 0000000000000000000000000000000000000000000000000000000000001011
reg[12] = 0000000000000000000000000000000000000000000000000000000000001100  reg[13] = 0000000000000000000000000000000000000000000000000000000000000000  reg[14] = 0000000000000000000000000000000000000000000000000000000000000000
reg[15] = 0000000000000000000000000000000000000000000000000000000000000000  reg[16] = 0000000000000000000000000000000000000000000000000000000000000000  reg[17] = 0111111111111111111111111111111111111111111111111111111111111111
reg[18] = 1000000000000000000000000000000000000000000000000000000000000000  reg[19] = 1111111111111111111111111111111111111111111111111111111111111111  reg[20] = 0000000000000000000000000000000000000000000000000000000000000000
reg[21] = 0000000000000000000000000000000000000000000000000000000000000000  reg[22] = 0000000000000000000000000000000000000000000000000000000000000000  reg[23] = 0000000000000000000000000000000000000000000000000000000000000000
reg[24] = 0000000000000000000000000000000000000000000000000000000000000000  reg[25] = 0000000000000000000000000000000000000000000000000000000000000000  reg[26] = 0000000000000000000000000000000000000000000000000000000000000000
reg[27] = 0000000000000000000000000000000000000000000000000000000000000000  reg[28] = 0000000000000000000000000000000000000000000000000000000000000000  reg[29] = 0000000000000000000000000000000000000000000000000000000000000000
reg[30] = 1010101010101010101010101010101010101010101010101010101010101010  reg[31] = 1111111111111111111111111111111111111111111111111111111111111100
memory[0] = 0000000000000000000000000000000000000000000000000000000000000000, memory[1] = 0000000000000000000000000000000000000000000000000000000000000001, memory[2] = 0000000000000000000000000000000000000000000000000000000000000010
memory[3] = 0000000000000000000000000000000000000000000000000000000000000011, memory[4] = 0000000000000000000000000000000000000000000000000000000000000100, memory[5] = 0000000000000000000000000000000000000000000000000000000000000101
memory[6] = 0000000000000000000000000000000000000000000000000000000000000110, memory[7] = 0000000000000000000000000000000000000000000000000000000000000111, memory[8] = 0000000000000000000000000000000000000000000000000000000000001000
memory[9] = 0000000000000000000000000000000000000000000000000000000000001001, memory[10] = 0000000000000000000000000000000000000000000000000000000000001010, memory[11] = 0000000000000000000000000000000000000000000000000000000000001011
memory[12] = 0000000000000000000000000000000000000000000000000000000000001100, memory[13] = 0000000000000000000000000000000000000000000000000000000000001101, memory[14] = 0000000000000000000000000000000000000000000000000000000000001110
memory[15] = 0000000000000000000000000000000000000000000000000000000000001111, memory[16] = 0000000000000000000000000000000000000000000000000000000000010000, memory[17] = 0000000000000000000000000000000000000000000000000000000000010001
memory[18] = 0000000000000000000000000000000000000000000000000000000000010010, memory[19] = 0000000000000000000000000000000000000000000000000000000000010011, memory[20] = 0000000000000000000000000000000000000000000000000000000000010100
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
Time: 30000
Current PC =                    4 instruction = 0020a023
rs1= 1, rs2= 2, rd= 0
ReadData1 = 0000000000000000000000000000000000000000000000000000000000000001, ReadData2 = 0000000000000000000000000000000000000000000000000000000000000010
ALU Output (valE) = 0000000000000000000000000000000000000000000000000000000000000001, Zero Flag = 0
Memory Read Data = 0000000000000000000000000000000000000000000000000000000000000000, Write Data = 0000000000000000000000000000000000000000000000000000000000000001
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=1, ALUSrc=1, RegWrite=0, ALUOp=00, carry=0, overflow=0
reg[ 0] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 0000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 0000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 0000000000000000000000000000000000000000000000000000000000001000  reg[ 4] = 0000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 0000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 0000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 0000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 0000000000000000000000000000000000000000000000000000000000000000  reg[10] = 0000000000000000000000000000000000000000000000000000000000001010  reg[11] = 0000000000000000000000000000000000000000000000000000000000001011
reg[12] = 0000000000000000000000000000000000000000000000000000000000001100  reg[13] = 0000000000000000000000000000000000000000000000000000000000000000  reg[14] = 0000000000000000000000000000000000000000000000000000000000000000
reg[15] = 0000000000000000000000000000000000000000000000000000000000000000  reg[16] = 0000000000000000000000000000000000000000000000000000000000000000  reg[17] = 0111111111111111111111111111111111111111111111111111111111111111
reg[18] = 1000000000000000000000000000000000000000000000000000000000000000  reg[19] = 1111111111111111111111111111111111111111111111111111111111111111  reg[20] = 0000000000000000000000000000000000000000000000000000000000000000
reg[21] = 0000000000000000000000000000000000000000000000000000000000000000  reg[22] = 0000000000000000000000000000000000000000000000000000000000000000  reg[23] = 0000000000000000000000000000000000000000000000000000000000000000
reg[24] = 0000000000000000000000000000000000000000000000000000000000000000  reg[25] = 0000000000000000000000000000000000000000000000000000000000000000  reg[26] = 0000000000000000000000000000000000000000000000000000000000000000
reg[27] = 0000000000000000000000000000000000000000000000000000000000000000  reg[28] = 0000000000000000000000000000000000000000000000000000000000000000  reg[29] = 0000000000000000000000000000000000000000000000000000000000000000
reg[30] = 1010101010101010101010101010101010101010101010101010101010101010  reg[31] = 1111111111111111111111111111111111111111111111111111111111111100
memory[0] = 0000000000000000000000000000000000000000000000000000000000000000, memory[1] = 0000000000000000000000000000000000000000000000000000000000000010, memory[2] = 0000000000000000000000000000000000000000000000000000000000000010
memory[3] = 0000000000000000000000000000000000000000000000000000000000000011, memory[4] = 0000000000000000000000000000000000000000000000000000000000000100, memory[5] = 0000000000000000000000000000000000000000000000000000000000000101
memory[6] = 0000000000000000000000000000000000000000000000000000000000000110, memory[7] = 0000000000000000000000000000000000000000000000000000000000000111, memory[8] = 0000000000000000000000000000000000000000000000000000000000000010
memory[9] = 0000000000000000000000000000000000000000000000000000000000000010, memory[10] = 0000000000000000000000000000000000000000000000000000000000001010, memory[11] = 0000000000000000000000000000000000000000000000000000000000001011
memory[12] = 0000000000000000000000000000000000000000000000000000000000001100, memory[13] = 0000000000000000000000000000000000000000000000000000000000001101, memory[14] = 0000000000000000000000000000000000000000000000000000000000001110
memory[15] = 0000000000000000000000000000000000000000000000000000000000001111, memory[16] = 0000000000000000000000000000000000000000000000000000000000010000, memory[17] = 0000000000000000000000000000000000000000000000000000000000010001
memory[18] = 0000000000000000000000000000000000000000000000000000000000010010, memory[19] = 0000000000000000000000000000000000000000000000000000000000010011, memory[20] = 0000000000000000000000000000000000000000000000000000000000010100
--------------------------------------------------------------------------------------------------------------------------------
```

STORE Operation

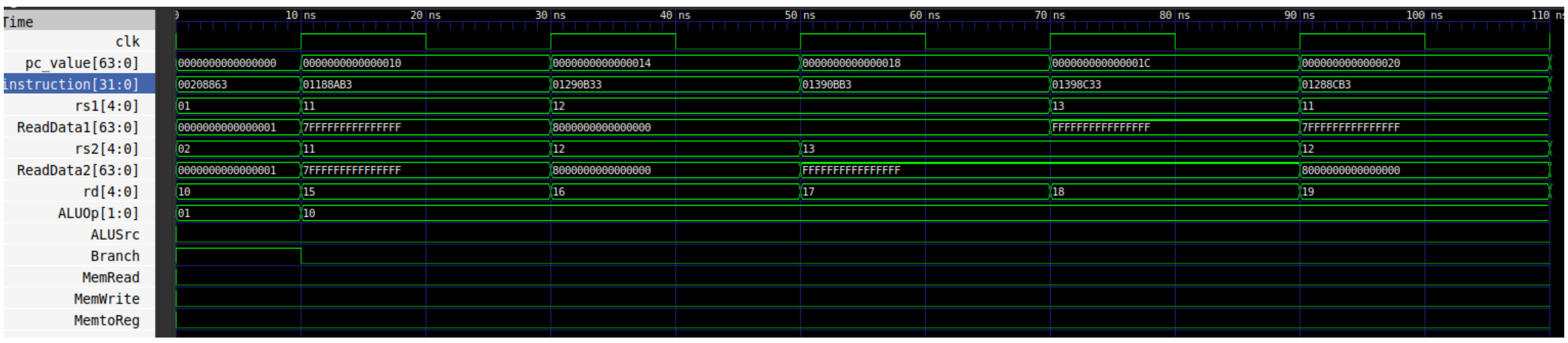
```
0020A023  // sd x4, 0(x2)
```

```
Time: 30000
Current PC =                    4 instruction = 0020a023
rs1= 1, rs2= 2, rd= 0
ReadData1 = 0000000000000000000000000000000000000000000000000000000000000001, ReadData2 = 0000000000000000000000000000000000000000000000000000000000000010
ALU Output (valE) = 0000000000000000000000000000000000000000000000000000000000000001, Zero Flag = 0
Memory Read Data = 0000000000000000000000000000000000000000000000000000000000000000, Write Data = 0000000000000000000000000000000000000000000000000000000000000001
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=1, ALUSrc=1, RegWrite=0, ALUOp=00, carry=0, overflow=0
reg[ 0] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 0000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 0000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 0000000000000000000000000000000000000000000000000000000000001000  reg[ 4] = 0000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 0000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 0000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 0000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 0000000000000000000000000000000000000000000000000000000000000000  reg[10] = 0000000000000000000000000000000000000000000000000000000000001010  reg[11] = 0000000000000000000000000000000000000000000000000000000000001011
reg[12] = 0000000000000000000000000000000000000000000000000000000000001100  reg[13] = 0000000000000000000000000000000000000000000000000000000000000000  reg[14] = 0000000000000000000000000000000000000000000000000000000000000000
reg[15] = 0000000000000000000000000000000000000000000000000000000000000000  reg[16] = 0000000000000000000000000000000000000000000000000000000000000000  reg[17] = 0111111111111111111111111111111111111111111111111111111111111111
reg[18] = 1000000000000000000000000000000000000000000000000000000000000000  reg[19] = 1111111111111111111111111111111111111111111111111111111111111111  reg[20] = 0000000000000000000000000000000000000000000000000000000000000000
reg[21] = 0000000000000000000000000000000000000000000000000000000000000000  reg[22] = 0000000000000000000000000000000000000000000000000000000000000000  reg[23] = 0000000000000000000000000000000000000000000000000000000000000000
reg[24] = 0000000000000000000000000000000000000000000000000000000000000000  reg[25] = 0000000000000000000000000000000000000000000000000000000000000000  reg[26] = 0000000000000000000000000000000000000000000000000000000000000000
reg[27] = 0000000000000000000000000000000000000000000000000000000000000000  reg[28] = 0000000000000000000000000000000000000000000000000000000000000000  reg[29] = 0000000000000000000000000000000000000000000000000000000000000000
reg[30] = 1010101010101010101010101010101010101010101010101010101010101010  reg[31] = 1111111111111111111111111111111111111111111111111111111111111100
memory[0] = 0000000000000000000000000000000000000000000000000000000000000000, memory[1] = 0000000000000000000000000000000000000000000000000000000000000010, memory[2] = 0000000000000000000000000000000000000000000000000000000000000010
memory[3] = 0000000000000000000000000000000000000000000000000000000000000011, memory[4] = 0000000000000000000000000000000000000000000000000000000000000100, memory[5] = 0000000000000000000000000000000000000000000000000000000000000101
memory[6] = 0000000000000000000000000000000000000000000000000000000000000110, memory[7] = 0000000000000000000000000000000000000000000000000000000000000111, memory[8] = 0000000000000000000000000000000000000000000000000000000000000010
memory[9] = 0000000000000000000000000000000000000000000000000000000000000010, memory[10] = 0000000000000000000000000000000000000000000000000000000000001010, memory[11] = 0000000000000000000000000000000000000000000000000000000000000010
memory[12] = 0000000000000000000000000000000000000000000000000000000000001100, memory[13] = 0000000000000000000000000000000000000000000000000000000000001101, memory[14] = 0000000000000000000000000000000000000000000000000000000000001110
memory[15] = 0000000000000000000000000000000000000000000000000000000000001111, memory[16] = 0000000000000000000000000000000000000000000000000000000000010000, memory[17] = 0000000000000000000000000000000000000000000000000000000000010001
memory[18] = 0000000000000000000000000000000000000000000000000000000000010010, memory[19] = 0000000000000000000000000000000000000000000000000000000000010011, memory[20] = 0000000000000000000000000000000000000000000000000000000000010100
----------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------
Time: 50000
Current PC =                    8 instruction = 00000013
rs1= 0, rs2= 0, rd= 0
ReadData1 = 0000000000000000000000000000000000000000000000000000000000000000, ReadData2 = 0000000000000000000000000000000000000000000000000000000000000000
ALU Output (valE) = 0000000000000000000000000000000000000000000000000000000000000000, Zero Flag = 1
Memory Read Data = 0000000000000000000000000000000000000000000000000000000000000000, Write Data = 0000000000000000000000000000000000000000000000000000000000000000
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=0, ALUOp=00, carry=0, overflow=0
reg[ 0] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 0000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 0000000000000000000000000000000000000000000000000000000000000010
reg[ 3] = 0000000000000000000000000000000000000000000000000000000000001000  reg[ 4] = 0000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 0000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 0000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 0000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 0000000000000000000000000000000000000000000000000000000000000000  reg[10] = 0000000000000000000000000000000000000000000000000000000000001010  reg[11] = 0000000000000000000000000000000000000000000000000000000000001011
reg[12] = 0000000000000000000000000000000000000000000000000000000000001100  reg[13] = 0000000000000000000000000000000000000000000000000000000000000000  reg[14] = 0000000000000000000000000000000000000000000000000000000000000000
reg[15] = 0000000000000000000000000000000000000000000000000000000000000000  reg[16] = 0000000000000000000000000000000000000000000000000000000000000000  reg[17] = 0111111111111111111111111111111111111111111111111111111111111111
reg[18] = 1000000000000000000000000000000000000000000000000000000000000000  reg[19] = 1111111111111111111111111111111111111111111111111111111111111111  reg[20] = 0000000000000000000000000000000000000000000000000000000000000000
reg[21] = 0000000000000000000000000000000000000000000000000000000000000000  reg[22] = 0000000000000000000000000000000000000000000000000000000000000000  reg[23] = 0000000000000000000000000000000000000000000000000000000000000000
reg[24] = 0000000000000000000000000000000000000000000000000000000000000000  reg[25] = 0000000000000000000000000000000000000000000000000000000000000000  reg[26] = 0000000000000000000000000000000000000000000000000000000000000000
reg[27] = 0000000000000000000000000000000000000000000000000000000000000000  reg[28] = 0000000000000000000000000000000000000000000000000000000000000000  reg[29] = 0000000000000000000000000000000000000000000000000000000000000000
reg[30] = 1010101010101010101010101010101010101010101010101010101010101010  reg[31] = 1111111111111111111111111111111111111111111111111111111111111100
memory[0] = 0000000000000000000000000000000000000000000000000000000000000000, memory[1] = 0000000000000000000000000000000000000000000000000000000000000010, memory[2] = 0000000000000000000000000000000000000000000000000000000000000010
memory[3] = 0000000000000000000000000000000000000000000000000000000000000011, memory[4] = 0000000000000000000000000000000000000000000000000000000000000100, memory[5] = 0000000000000000000000000000000000000000000000000000000000000101
memory[6] = 0000000000000000000000000000000000000000000000000000000000000110, memory[7] = 0000000000000000000000000000000000000000000000000000000000000111, memory[8] = 0000000000000000000000000000000000000000000000000000000000000010
memory[9] = 0000000000000000000000000000000000000000000000000000000000000010, memory[10] = 0000000000000000000000000000000000000000000000000000000000001010, memory[11] = 0000000000000000000000000000000000000000000000000000000000000010
memory[12] = 0000000000000000000000000000000000000000000000000000000000001100, memory[13] = 0000000000000000000000000000000000000000000000000000000000001101, memory[14] = 0000000000000000000000000000000000000000000000000000000000001110
memory[15] = 0000000000000000000000000000000000000000000000000000000000001111, memory[16] = 0000000000000000000000000000000000000000000000000000000000010000, memory[17] = 0000000000000000000000000000000000000000000000000000000000010001
memory[18] = 0000000000000000000000000000000000000000000000000000000000010010, memory[19] = 0000000000000000000000000000000000000000000000000000000000010011, memory[20] = 0000000000000000000000000000000000000000000000000000000000010100
```

BRANCH Operation

```
00208863 //beq x1 ×2 16
```

```
Time: 10000
Current PC =                    0 instruction = 00208863
rs1= 1, rs2= 2, rd=16
ReadData1 = 0000000000000000000000000000000000000000000000000000000000000001, ReadData2 = 0000000000000000000000000000000000000000000000000000000000000001
ALU Output (valE) = 0000000000000000000000000000000000000000000000000000000000000000, Zero Flag = 1
Memory Read Data = 0000000000000000000000000000000000000000000000000000000000000000, Write Data = 0000000000000000000000000000000000000000000000000000000000000000
Control Signals: Branch=1, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=0, ALUOp=01, carry=1, overflow=0
reg[ 0] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 0000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 0000000000000000000000000000000000000000000000000000000000000001
reg[ 3] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 0000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 0000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 0000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 0000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 0000000000000000000000000000000000000000000000000000000000000000  reg[10] = 0000000000000000000000000000000000000000000000000000000000001010  reg[11] = 0000000000000000000000000000000000000000000000000000000000001011
reg[12] = 0000000000000000000000000000000000000000000000000000000000001100  reg[13] = 0000000000000000000000000000000000000000000000000000000000000000  reg[14] = 0000000000000000000000000000000000000000000000000000000000000000
reg[15] = 0000000000000000000000000000000000000000000000000000000000000000  reg[16] = 0000000000000000000000000000000000000000000000000000000000000000  reg[17] = 0111111111111111111111111111111111111111111111111111111111111111
reg[18] = 1000000000000000000000000000000000000000000000000000000000000000  reg[19] = 1111111111111111111111111111111111111111111111111111111111111111  reg[20] = 0000000000000000000000000000000000000000000000000000000000000000
reg[21] = 0000000000000000000000000000000000000000000000000000000000000000  reg[22] = 0000000000000000000000000000000000000000000000000000000000000000  reg[23] = 0000000000000000000000000000000000000000000000000000000000000000
reg[24] = 0000000000000000000000000000000000000000000000000000000000000000  reg[25] = 0000000000000000000000000000000000000000000000000000000000000000  reg[26] = 0000000000000000000000000000000000000000000000000000000000000000
reg[27] = 0000000000000000000000000000000000000000000000000000000000000000  reg[28] = 0000000000000000000000000000000000000000000000000000000000000000  reg[29] = 0000000000000000000000000000000000000000000000000000000000000000
reg[30] = 1010101010101010101010101010101010101010101010101010101010101010  reg[31] = 1111111111111111111111111111111111111111111111111111111111111100
----------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------
Time: 30000
Current PC =                   16 instruction = 01188ab3
rs1=17, rs2=17, rd=21
ReadData1 = 0111111111111111111111111111111111111111111111111111111111111111, ReadData2 = 0111111111111111111111111111111111111111111111111111111111111111
ALU Output (valE) = 1111111111111111111111111111111111111111111111111111111111111110, Zero Flag = 0
Memory Read Data = 0000000000000000000000000000000000000000000000000000000000000000, Write Data = 1111111111111111111111111111111111111111111111111111111111111110
Control Signals: Branch=0, MemRead=0, MemtoReg=0, MemWrite=0, ALUSrc=0, RegWrite=1, ALUOp=10, carry=0, overflow=1
reg[ 0] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 1] = 0000000000000000000000000000000000000000000000000000000000000001  reg[ 2] = 0000000000000000000000000000000000000000000000000000000000000001
reg[ 3] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 4] = 0000000000000000000000000000000000000000000000000000000000000100  reg[ 5] = 0000000000000000000000000000000000000000000000000000000000000101
reg[ 6] = 0000000000000000000000000000000000000000000000000000000000000000  reg[ 7] = 0000000000000000000000000000000000000000000000000000000000000111  reg[ 8] = 0000000000000000000000000000000000000000000000000000000000001000
reg[ 9] = 0000000000000000000000000000000000000000000000000000000000000000  reg[10] = 0000000000000000000000000000000000000000000000000000000000001010  reg[11] = 0000000000000000000000000000000000000000000000000000000000001011
reg[12] = 0000000000000000000000000000000000000000000000000000000000001100  reg[13] = 0000000000000000000000000000000000000000000000000000000000000000  reg[14] = 0000000000000000000000000000000000000000000000000000000000000000
reg[15] = 0000000000000000000000000000000000000000000000000000000000000000  reg[16] = 0000000000000000000000000000000000000000000000000000000000000000  reg[17] = 0111111111111111111111111111111111111111111111111111111111111111
reg[18] = 1000000000000000000000000000000000000000000000000000000000000000  reg[19] = 1111111111111111111111111111111111111111111111111111111111111111  reg[20] = 0000000000000000000000000000000000000000000000000000000000000000
reg[21] = 0000000000000000000000000000000000000000000000000000000000000000  reg[22] = 0000000000000000000000000000000000000000000000000000000000000000  reg[23] = 0000000000000000000000000000000000000000000000000000000000000000
reg[24] = 0000000000000000000000000000000000000000000000000000000000000000  reg[25] = 0000000000000000000000000000000000000000000000000000000000000000  reg[26] = 0000000000000000000000000000000000000000000000000000000000000000
reg[27] = 0000000000000000000000000000000000000000000000000000000000000000  reg[28] = 0000000000000000000000000000000000000000000000000000000000000000  reg[29] = 0000000000000000000000000000000000000000000000000000000000000000
reg[30] = 1010101010101010101010101010101010101010101010101010101010101010  reg[31] = 1111111111111111111111111111111111111111111111111111111111111100
```


Time
clk
pc_value[63:0]
instruction[31:0]
rs1[4:0]
ReadData1[63:0]
rs2[4:0]
ReadData2[63:0]
rd[4:0]
ALUOp[1:0]
ALUSrc
Branch
MemRead
MemWrite
MemtoReg
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns
110 ns
0000000000000000
0000000000000010
0000000000000014
0000000000000018
000000000000001C
0000000000000020
00208863
01188AB3
01290B33
01390BB3
01398C33
01288CB3
01
11
12
13
11
0000000000000001
7FFFFFFFFFFFFFFF
8000000000000000
FFFFFFFFFFFFFFFF
7FFFFFFFFFFFFFFF
02
11
12
13
12
0000000000000001
7FFFFFFFFFFFFFFF
8000000000000000
FFFFFFFFFFFFFFFF
8000000000000000
10
15
16
17
18
19
01
10

TASK - 2: Pipeline Implementation

In pipeline implementation, the architecture improves instruction throughput by dividing the execution process into multiple stages, allowing multiple instructions to be executed concurrently in different stages. This approach significantly increases performance compared to a sequential execution model. A five stage pipeline commonly consists of:

- **Instruction Fetch (IF)**
- **Instruction Decode & Register Read (ID)**
- **Execute (EX)**
- **Memory Access (MEM)**
- **Write-Back (WB)**

In a pipelined processor, each stage must pass data to the next stage in the following cycle. To facilitate this, **pipeline registers** are used between pipeline stages to store intermediate values. These registers help maintain the separation of instruction processing across clock cycles. Pipeline registers store the instruction being processed, register values, ALU results, memory addresses and control signals required and the control signals required by the next stage. The pipeline registers are named on the basis of the stages they connect:

- **IF/ID Register** (Between Instruction Fetch and Instruction Decode):

  It stores the fetched information and the updated program counter. From here the instruction is passed to the **decode** stage for interpretation.

- **ID/EX Register** (Between Instruction Decode and Execute):

  It stores the decoded instruction fields(operands and control signals). It ensures that the operands are available for the ALU in the **execute** stage.

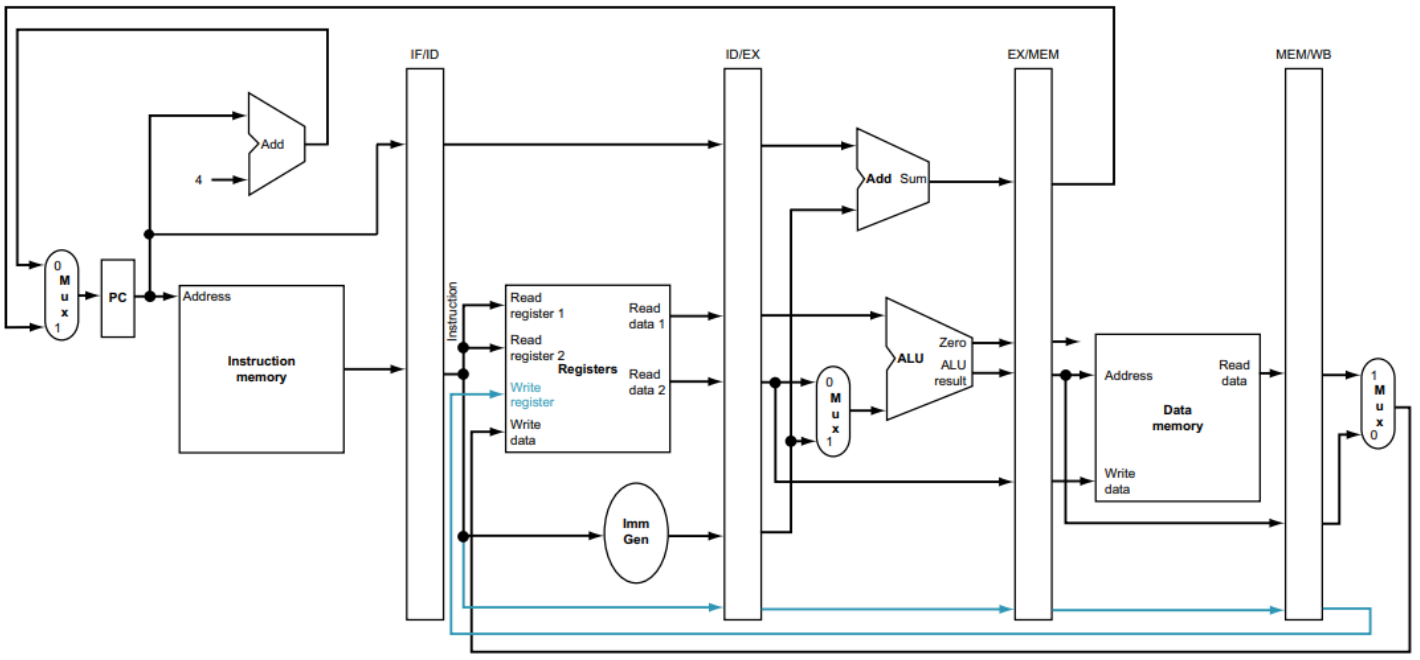
- **EX/MEM Register** (Between Execute and Memory Access):

  It stores ALU computation results and branch decision outputs. If there is a memory operation, the address is stored.

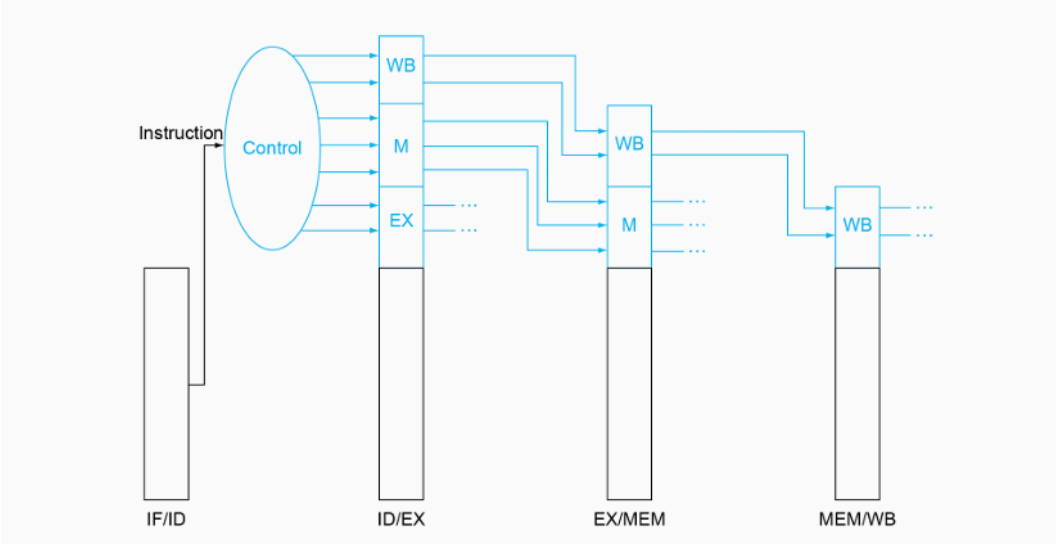
- **MEM/WB Register** (Between Memory Access and Write-Back):

  It stores the final result before updating the register file.

So the Datapath(without considering hazards and control signals) is:



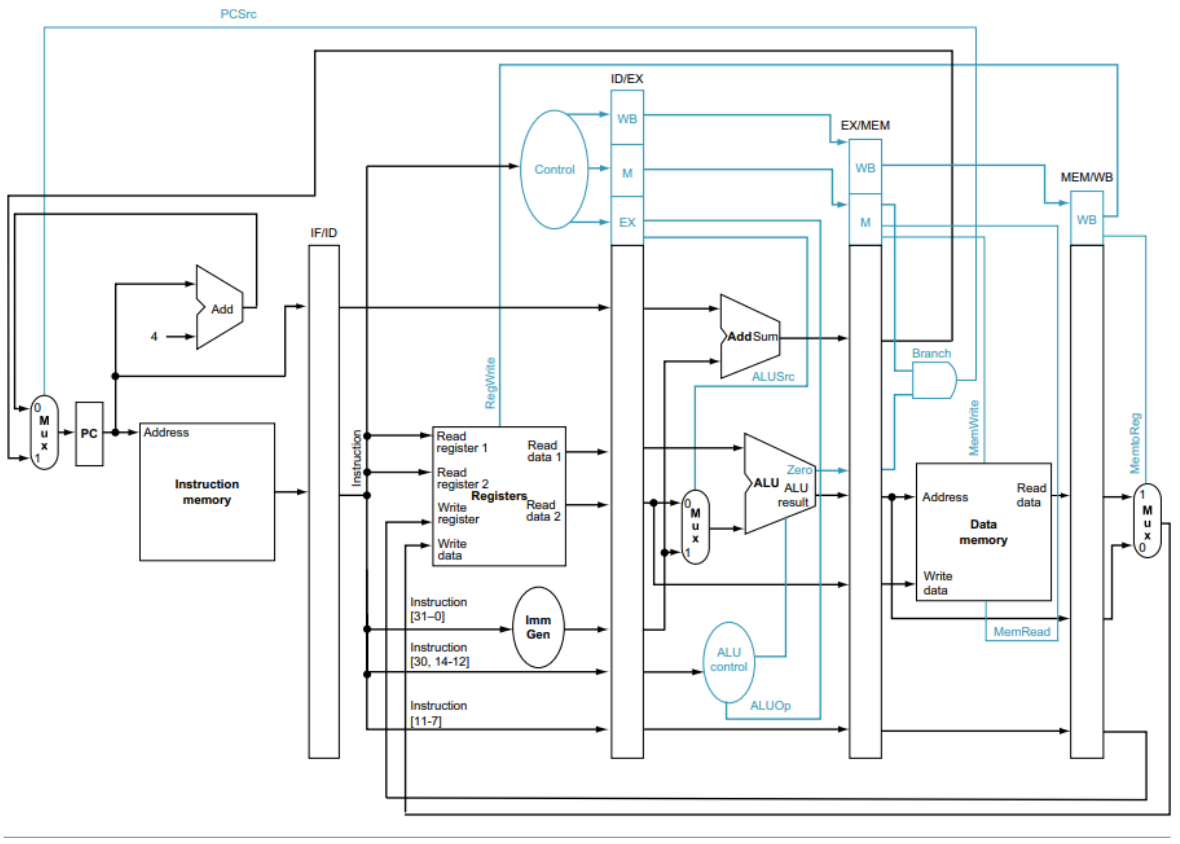
The control signals are given to the register in the following way:



The roles of these pipeline registers are:

1. They prevent data corruption by ensuring that each stage works with the correct instruction.
2. They allow multiple instructions to run simultaneously.

The Datapath with control signals is:



Working of the pipelined processor for the Instruction set:

```
002081b3  // add x3,x1,x2
00520333  // add x6,x4,x5
008384b3  // add x9,x7,x8
00b50633  // add x12,x10,x11
00e687b3  // add x15,x13,x14
```

Clock Cycle-1

```
================================================================================================================================================
Time: 5
IF Stage: PC_if = 0000000000000000 | instruction_if = 002081b3

ID Stage: PC_id = xxxxxxxxxxxxxxxx | instruction_id = xxxxxxxx

Decode Stage: ReadData1 = xxxxxxxxxxxxxxxx | ReadData2 = xxxxxxxxxxxxxxxx | imm =                    x
Decode Control Signals: Branch = x | MemRead = x | MemtoReg = x | ALUOp = xx | MemWrite = x | ALUSrc = x | RegWrite = x

Instruction=xxxxxxxx
Rs1= x, Rs2= x,Memread=x, idex_rd=xxxxx, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = xx | rs2 = xx | rd = xx | funct3 = x | funct7 = xx | imm =                   x
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = xx | rs2 = xx | rd = xx | funct3 = x | funct7 = xx | imm =                    x
IDEX Control Signals: Branch = x | MemRead = x | MemtoReg = x | ALUOp = xx | MemWrite = x | ALUSrc = x | RegWrite = x

EX Stage: PC_ex = xxxxxxxxxxxxxxxx

Execute Inputs: ALUOp = xx | funct7 = xx | funct3 = x | read_data1 = xxxxxxxxxxxxxxxx | pc = xxxxxxxxxxxxxxxx | imm =                    x

Execute Outputs: ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x | Branch PC = xxxxxxxxxxxxxxxx

EXMEM Inputs: Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | PC = xxxxxxxxxxxxxxxx | ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x | rd = xx
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = xxxxxxxxxxxxxxxx | rd = xx | Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x

MEM Stage: ReadData = 0000000000000000 | ALU Result = xxxxxxxxxxxxxxxx | PC = xxxxxxxxxxxxxxxx | rd = xx

MEMWB Inputs: PC = xxxxxxxxxxxxxxxx | rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = 00 | MemtoReg = 0 | RegWrite = 0 | ALU Result = 0000000000000000 | ReadData = 0000000000000000

WB Stage: Write Data = 0000000000000000 | RegWrite = 0 | rd = 00

================================================================================================================================================
```

Clock Cycle - 2

```
================================================================================================================================================
Time: 15
IF Stage: PC_if = 0000000000000004 | instruction_if = 00520333

ID Stage: PC_id = 0000000000000000 | instruction_id = 002081b3

Decode Stage: ReadData1 = 0000000000000001 | ReadData2 = 0000000000000001 | imm =                    0
Decode Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

Instruction=002081b3
Rs1= 1, Rs2= 2,Memread=x, idex_rd=xxxxx, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = 01 | rs2 = 02 | rd = 03 | funct3 = 0 | funct7 = 00 | imm =                   0
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = xx | rs2 = xx | rd = xx | funct3 = x | funct7 = xx | imm =                    x
IDEX Control Signals: Branch = x | MemRead = x | MemtoReg = x | ALUOp = xx | MemWrite = x | ALUSrc = x | RegWrite = x

EX Stage: PC_ex = xxxxxxxxxxxxxxxx

Execute Inputs: ALUOp = xx | funct7 = xx | funct3 = x | read_data1 = xxxxxxxxxxxxxxxx | pc = xxxxxxxxxxxxxxxx | imm =                    x

Execute Outputs: ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x | Branch PC = xxxxxxxxxxxxxxxx

EXMEM Inputs: Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | PC = xxxxxxxxxxxxxxxx | ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x | rd = xx
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = xxxxxxxxxxxxxxxx | rd = xx | Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x

MEM Stage: ReadData = 0000000000000000 | ALU Result = xxxxxxxxxxxxxxxx | PC = xxxxxxxxxxxxxxxx | rd = xx

MEMWB Inputs: PC = xxxxxxxxxxxxxxxx | rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | ReadData = 0000000000000000

WB Stage: Write Data = xxxxxxxxxxxxxxxx | RegWrite = x | rd = xx

================================================================================================================================================
```

Clock Cycle - 3

```
================================================================================================================================================
Time: 25
IF Stage: PC_if = 0000000000000008 | instruction_if = 008384b3

ID Stage: PC_id = 0000000000000004 | instruction_id = 00520333

Decode Stage: ReadData1 = 0000000000000004 | ReadData2 = 0000000000000005 | imm =                    0
Decode Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

Instruction=00520333
Rs1= 4, Rs2= 5,Memread=0, idex_rd=00011, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = 04 | rs2 = 05 | rd = 06 | funct3 = 0 | funct7 = 00 | imm =                   0
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = 01 | rs2 = 02 | rd = 03 | funct3 = 0 | funct7 = 00 | imm =                    0
IDEX Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

EX Stage: PC_ex = 0000000000000000

Execute Inputs: ALUOp = 10 | funct7 = 00 | funct3 = 0 | read_data1 = 0000000000000001 | pc = 0000000000000000 | imm =                    0

Execute Outputs: ALU Result = 0000000000000002 | Zero Flag = 0 | Branch PC = 0000000000000004

EXMEM Inputs: Branch = 0 | MemRead = 0 | MemtoReg = 0 | MemWrite = 0 | RegWrite = 1 | PC = 0000000000000000 | ALU Result = 0000000000000002 | Zero Flag = 0 | rd = 03
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = xxxxxxxxxxxxxxxx | rd = xx | Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x

MEM Stage: ReadData = 0000000000000000 | ALU Result = xxxxxxxxxxxxxxxx | PC = xxxxxxxxxxxxxxxx | rd = xx

MEMWB Inputs: PC = xxxxxxxxxxxxxxxx | rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | ReadData = 0000000000000000

WB Stage: Write Data = xxxxxxxxxxxxxxxx | RegWrite = x | rd = xx

================================================================================================================================================
```

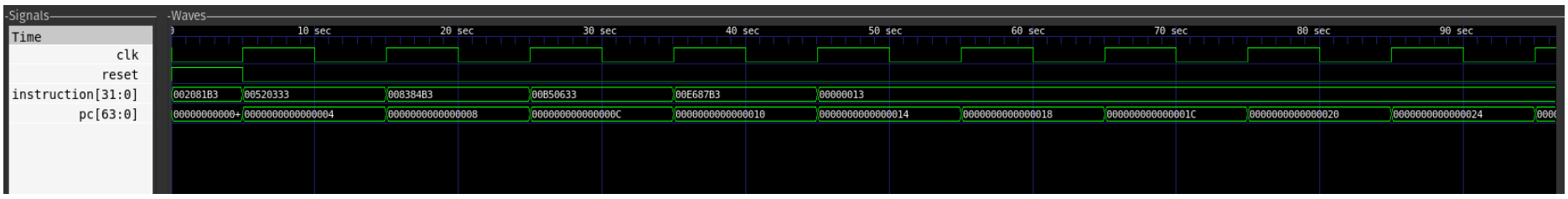
Clock Cycle - 4

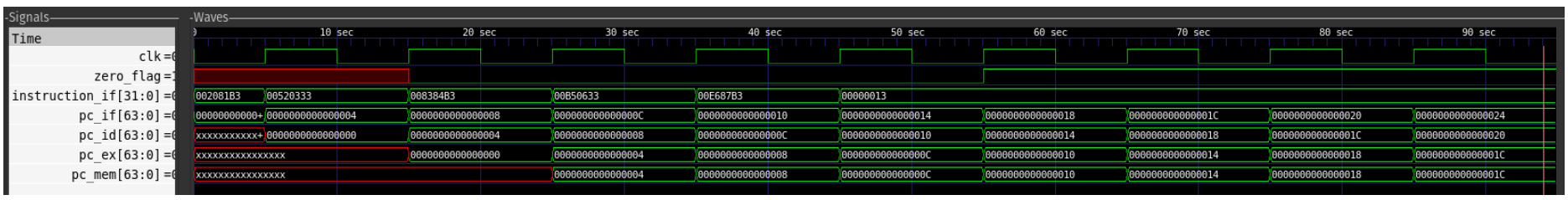
```
================================================================================================================================================
Time: 35
IF Stage: PC_if = 000000000000000c | instruction_if = 00b50633

ID Stage: PC_id = 0000000000000008 | instruction_id = 008384b3

Decode Stage: ReadData1 = 0000000000000007 | ReadData2 = 0000000000000008 | imm =                    0
Decode Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

Instruction=008384b3
Rs1= 7, Rs2= 8,Memread=0, idex_rd=00110, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = 07 | rs2 = 08 | rd = 09 | funct3 = 0 | funct7 = 00 | imm =                    0
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = 04 | rs2 = 05 | rd = 06 | funct3 = 0 | funct7 = 00 | imm =                    0
IDEX Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

EX Stage: PC_ex = 0000000000000004

Execute Inputs: ALUOp = 10 | funct7 = 00 | funct3 = 0 | read_data1 = 0000000000000004 | pc = 0000000000000004 | imm =                    0

Execute Outputs: ALU Result = 0000000000000009 | Zero Flag = 0 | Branch PC = 0000000000000008

EXMEM Inputs: Branch = 0 | MemRead = 0 | MemtoReg = 0 | MemWrite = 0 | RegWrite = 1 | PC = 0000000000000004 | ALU Result = 0000000000000009 | Zero Flag = 0 | rd = 06
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = 0000000000000004 | rd = 03 | Branch = 0 | MemRead = 0 | MemtoReg = 0 | MemWrite = 0 | RegWrite = 1 | ALU Result = 0000000000000002 | Zero Flag = 0

MEM Stage: ReadData = 0000000000000000 | ALU Result = 0000000000000002 | PC = 0000000000000004 | rd = 03

MEMWB Inputs: PC = 0000000000000004 | rd = 03 | MemtoReg = 0 | RegWrite = 1 | ALU Result = 0000000000000002 | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | ReadData = 0000000000000000

WB Stage: Write Data = xxxxxxxxxxxxxxxx | RegWrite = x | rd = xx

================================================================================================================================================
```

Clock Cycle - 5

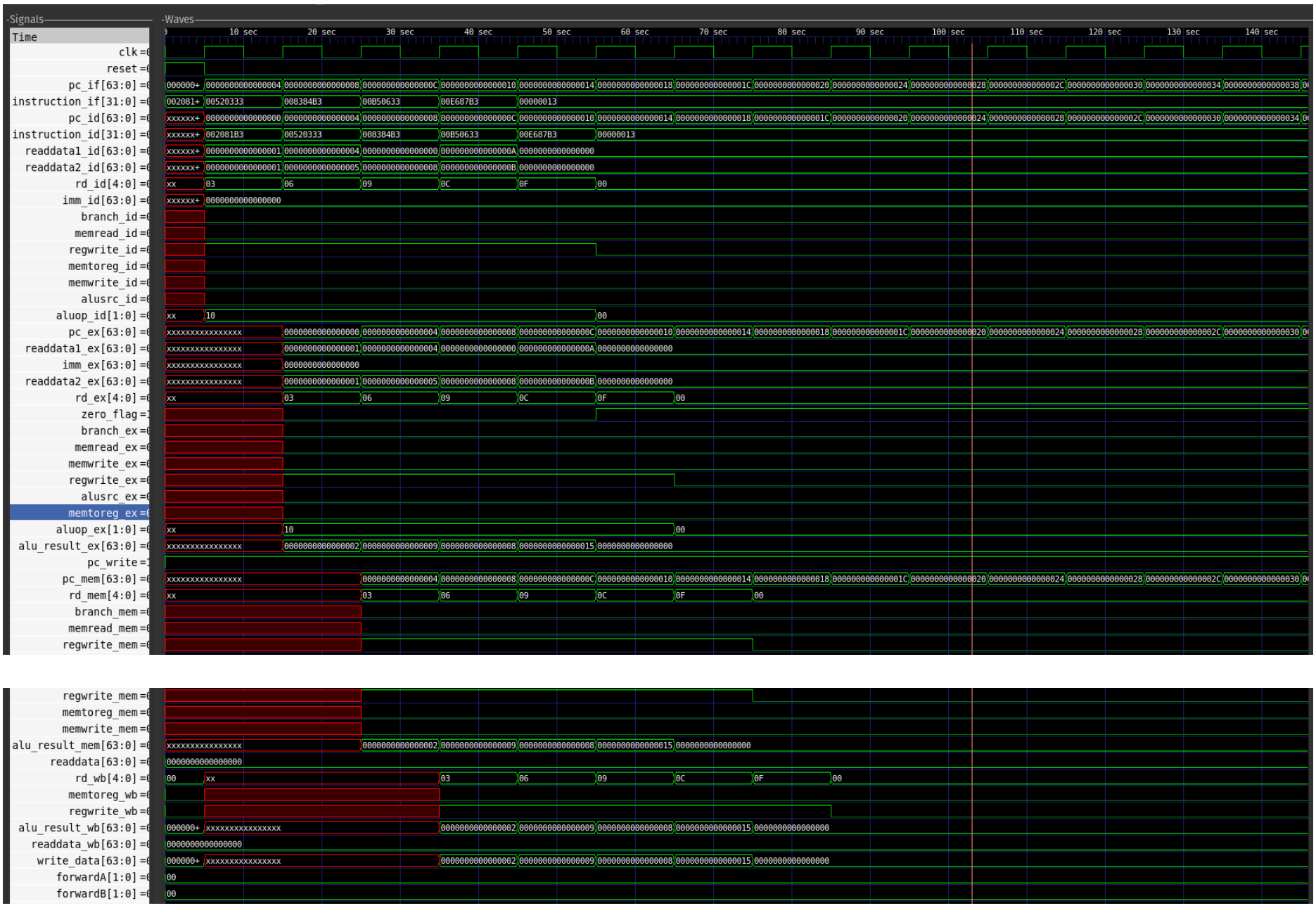
```
================================================================================================================================================
Time: 45
IF Stage: PC_if = 0000000000000010 | instruction_if = 00e687b3

ID Stage: PC_id = 000000000000000c | instruction_id = 00b50633

Decode Stage: ReadData1 = 000000000000000a | ReadData2 = 000000000000000b | imm =                    0
Decode Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

Instruction=00b50633
Rs1=10, Rs2=11,Memread=0, idex_rd=01001, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = 0a | rs2 = 0b | rd = 0c | funct3 = 0 | funct7 = 00 | imm =                    0
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = 07 | rs2 = 08 | rd = 09 | funct3 = 0 | funct7 = 00 | imm =                    0
IDEX Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

EX Stage: PC_ex = 0000000000000008

Execute Inputs: ALUOp = 10 | funct7 = 00 | funct3 = 0 | read_data1 = 0000000000000007 | pc = 0000000000000008 | imm =                    0

Execute Outputs: ALU Result = 000000000000000f | Zero Flag = 0 | Branch PC = 000000000000000c

EXMEM Inputs: Branch = 0 | MemRead = 0 | MemtoReg = 0 | MemWrite = 0 | RegWrite = 1 | PC = 0000000000000008 | ALU Result = 000000000000000f | Zero Flag = 0 | rd = 09
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = 0000000000000008 | rd = 06 | Branch = 0 | MemRead = 0 | MemtoReg = 0 | MemWrite = 0 | RegWrite = 1 | ALU Result = 0000000000000009 | Zero Flag = 0

MEM Stage: ReadData = 0000000000000000 | ALU Result = 0000000000000009 | PC = 0000000000000008 | rd = 06

MEMWB Inputs: PC = 0000000000000008 | rd = 06 | MemtoReg = 0 | RegWrite = 1 | ALU Result = 0000000000000009 | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = 03 | MemtoReg = 0 | RegWrite = 1 | ALU Result = 0000000000000002 | ReadData = 0000000000000000

WB Stage: Write Data = 0000000000000002 | RegWrite = 1 | rd = 03

================================================================================================================================================
```

Instruction and PC Update in a stage:





Complete GTK Wave:



Hazard Handling

1. Structural Hazards

a. They occur when two or more instructions in the pipeline require the same hardware resource at the same time, causing conflicts and stalls. This happens when the hardware does not support simultaneous access to shared components.
b. If the processor has only one memory unit, it cannot fetch an instruction and **load/store** data in the same cycle. For example, if an instruction fetch (IF stage) and a memory access (MEM stage) need memory access at the same time, a stall occurs. Example for structural Hazard can be:

```
ld x1, 0(x2)  // Load instruction (needs memory access)
add x3, x1, x4  // Arithmetic instruction (needs ALU)
sd x5, 4(x6)  // Store instruction (needs memory access)
```

Solution to the Structural Hazard:
Using separate instruction and data memory will solve structural hazard. So even if ld(load) and dd(store) both need memory access in the same cycle, there will not be any problem.

2. Data Hazards

A data Hazard occurs when an instruction depends on the result of a previous instruction that has not yet completed.

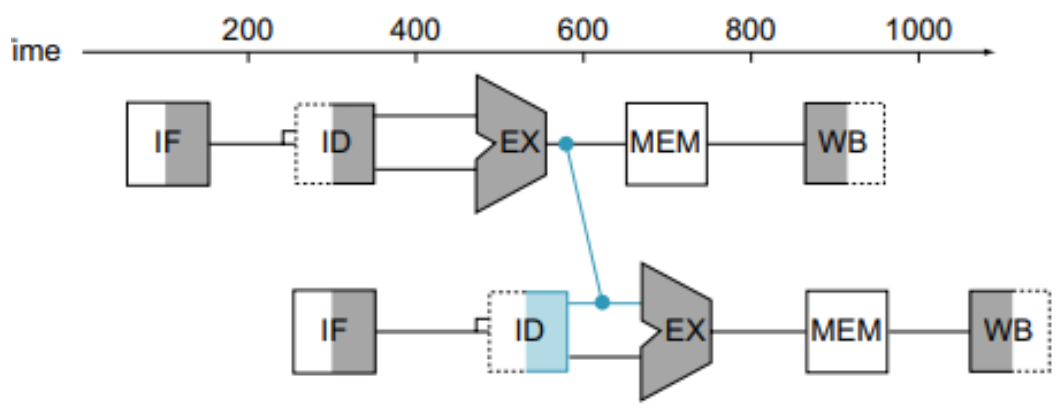
Since a pipelined processor executes multiple instructions simultaneously at different stages, a later instruction might need a value that an earlier instruction is still computing. For

example,

```
add x3, x1, x2  //x3 = x1 + x2
sub x4, x3, x5  //x4 = x3 - x5
```

Solutions to Data Hazards:
1. Pipelining Stalling i.e. inserting NOP statements: Stalling (or inserting pipeline bubbles) temporarily stops the pipeline until the required data is available. This prevents incorrect execution but causes performance loss as it wastes clock cycles.

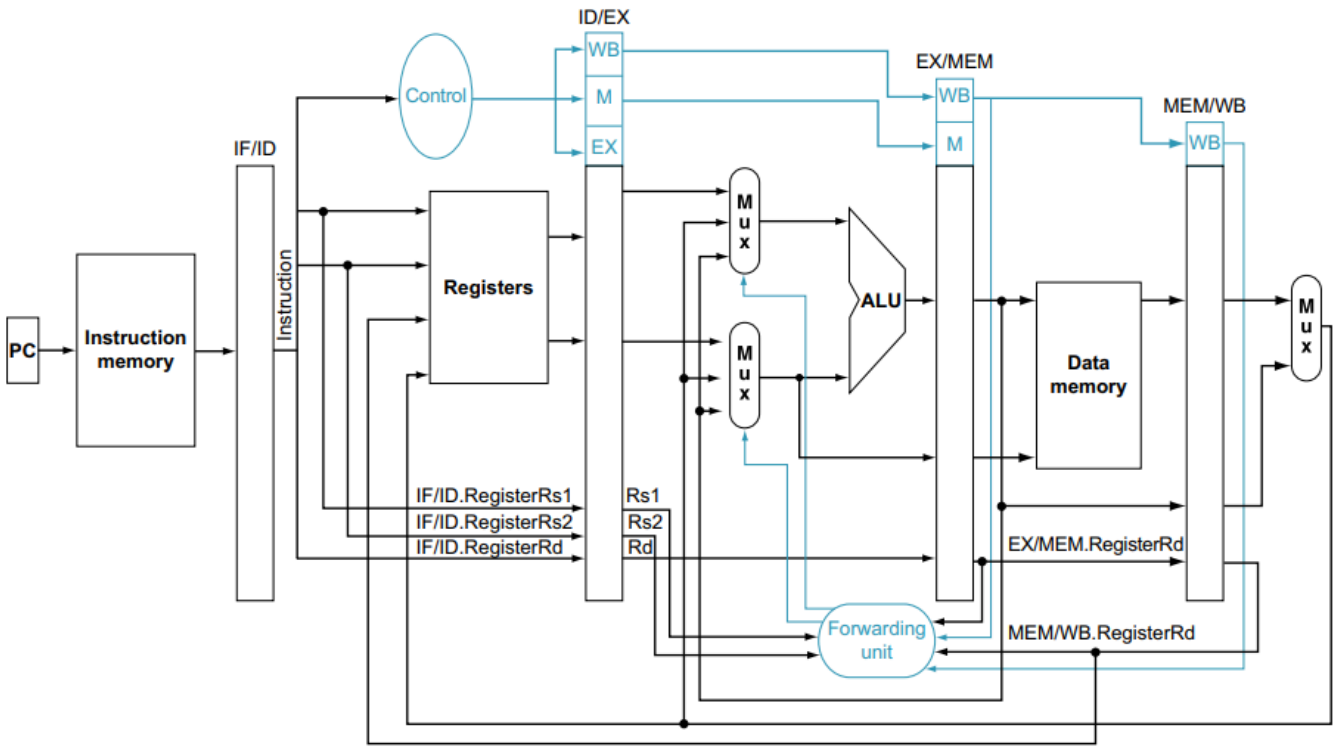
2. Data forwarding(Bypassing):Instead of stalling, bypassing is used to pass the required data directly from one pipeline stage to another avoiding unnecessary delays. For the above example forwarding eliminates stalls and improves performance. In Forwarding, instead of waiting for a value to be written back (WB stage) to the register file, the processor **forwards** the value directly from the EX/MEM or MEM/WB pipeline registers to the next instruction.
   For the above example, without forwarding, the sub instruction would stall for one cycle but with forwarding, the processor bypasses
   `x3` from the ALU output of `add` directly to the ALU input of `sub`.



The logic for the forwarding part:

Listing the hazard Conditions:

1a. EX/MEM.RegisterRd = ID/EX.RegisterRs1
1b. EX/MEM.RegisterRd = ID/EX.RegisterRs2
2a. MEM/WB.RegisterRd = ID/EX.RegisterRs1
2b. MEM/WB.RegisterRd = ID/EX.RegisterRs2
It is compulsory to note that EX/MEM.RegisterRd ≠ 0 and MEM/WB.RegisterRd ≠ 0
So in the Datapath, there should be a forwarding unit, which will check all the above hazard conditions and if hazard is detected it enables the mux to select the forwarded value instead of reading from the register file.



Data Hazard

```
002081b3  // add x3,x1,x2
 004182b3  // add x5,x3,x4
 006183b3  // add x7,x3,x6
```

Clock Cycle - 1

```
================================================================================================================================================
Time: 5
IF Stage: PC_if = 0000000000000000 | instruction_if = 002081b3

ID Stage: PC_id = xxxxxxxxxxxxxxxx | instruction_id = xxxxxxxx

Decode Stage: ReadData1 = xxxxxxxxxxxxxxxx | ReadData2 = xxxxxxxxxxxxxxxx | imm =                    x
Decode Control Signals: Branch = x | MemRead = x | MemtoReg = x | ALUOp = xx | MemWrite = x | ALUSrc = x | RegWrite = x

Instruction=xxxxxxxx
Rs1= x, Rs2= x,Memread=x, idex_rd=xxxxx, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = xx | rs2 = xx | rd = xx | funct3 = x | funct7 = xx | imm =                    x
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = xx | rs2 = xx | rd = xx | funct3 = x | funct7 = xx | imm =                    x
IDEX Control Signals: Branch = x | MemRead = x | MemtoReg = x | ALUOp = xx | MemWrite = x | ALUSrc = x | RegWrite = x

EX Stage: PC_ex = xxxxxxxxxxxxxxxx

Execute Inputs: ALUOp = xx | funct7 = xx | funct3 = x | read_data1 = xxxxxxxxxxxxxxxx | pc = xxxxxxxxxxxxxxxx | imm =                    x

Execute Outputs: ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x | Branch PC = xxxxxxxxxxxxxxxx

EXMEM Inputs: Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | PC = xxxxxxxxxxxxxxxx | ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x | rd = xx
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = xxxxxxxxxxxxxxxx | rd = xx | Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x

MEM Stage: ReadData = 0000000000000000 | ALU Result = xxxxxxxxxxxxxxxx | PC = xxxxxxxxxxxxxxxx | rd = xx

MEMWB Inputs: PC = xxxxxxxxxxxxxxxx | rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = 00 | MemtoReg = 0 | RegWrite = 0 | ALU Result = 0000000000000000 | ReadData = 0000000000000000

WB Stage: Write Data = 0000000000000000 | RegWrite = 0 | rd = 00

================================================================================================================================================
```

Clock Cycle - 2

```
================================================================================================================================================
Time: 15
IF Stage: PC_if = 0000000000000004 | instruction_if = 004182b3

ID Stage: PC_id = 0000000000000000 | instruction_id = 002081b3

Decode Stage: ReadData1 = 0000000000000001 | ReadData2 = 0000000000000001 | imm =                    0
Decode Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

Instruction=002081b3
Rs1= 1, Rs2= 2,Memread=x, idex_rd=xxxxx, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = 01 | rs2 = 02 | rd = 03 | funct3 = 0 | funct7 = 00 | imm =                    0
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = xx | rs2 = xx | rd = xx | funct3 = x | funct7 = xx | imm =                    x
IDEX Control Signals: Branch = x | MemRead = x | MemtoReg = x | ALUOp = xx | MemWrite = x | ALUSrc = x | RegWrite = x

EX Stage: PC_ex = xxxxxxxxxxxxxxx

Execute Inputs: ALUOp = xx | funct7 = xx | funct3 = x | read_data1 = xxxxxxxxxxxxxxx | pc = xxxxxxxxxxxxxxx | imm =                    x

Execute Outputs: ALU Result = xxxxxxxxxxxxxxx | Zero Flag = x | Branch PC = xxxxxxxxxxxxxxx

EXMEM Inputs: Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | PC = xxxxxxxxxxxxxxx | ALU Result = xxxxxxxxxxxxxxx | Zero Flag = x | rd = xx
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = xxxxxxxxxxxxxxx | rd = xx | Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxx | Zero Flag = x

MEM Stage: ReadData = 0000000000000000 | ALU Result = xxxxxxxxxxxxxxx | PC = xxxxxxxxxxxxxxx | rd = xx

MEMWB Inputs: PC = xxxxxxxxxxxxxxx | rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxx | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxx | ReadData = 0000000000000000

WB Stage: Write Data = xxxxxxxxxxxxxxx | RegWrite = x | rd = xx

================================================================================================================================================
```

Clock Cycle - 3

```
================================================================================================================================================
Time: 25
IF Stage: PC_if = 0000000000000008 | instruction_if = 006183b3

ID Stage: PC_id = 0000000000000004 | instruction_id = 004182b3

Decode Stage: ReadData1 = 0000000000000000 | ReadData2 = 0000000000000004 | imm =                    0
Decode Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

Instruction=004182b3
Rs1= 3, Rs2= 4,Memread=0, idex_rd=00011, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = 03 | rs2 = 04 | rd = 05 | funct3 = 0 | funct7 = 00 | imm =                    0
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = 01 | rs2 = 02 | rd = 03 | funct3 = 0 | funct7 = 00 | imm =                    0
IDEX Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

EX Stage: PC_ex = 0000000000000000

Execute Inputs: ALUOp = 10 | funct7 = 00 | funct3 = 0 | read_data1 = 0000000000000001 | pc = 0000000000000000 | imm =                    0

Execute Outputs: ALU Result = 0000000000000002 | Zero Flag = 0 | Branch PC = 0000000000000004

EXMEM Inputs: Branch = 0 | MemRead = 0 | MemtoReg = 0 | MemWrite = 0 | RegWrite = 1 | PC = 0000000000000000 | ALU Result = 0000000000000002 | Zero Flag = 0 | rd = 03
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = xxxxxxxxxxxxxxx | rd = xx | Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxx | Zero Flag = x

MEM Stage: ReadData = 0000000000000000 | ALU Result = xxxxxxxxxxxxxxx | PC = xxxxxxxxxxxxxxx | rd = xx

MEMWB Inputs: PC = xxxxxxxxxxxxxxx | rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxx | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxx | ReadData = 0000000000000000

WB Stage: Write Data = xxxxxxxxxxxxxxx | RegWrite = x | rd = xx

================================================================================================================================================
```

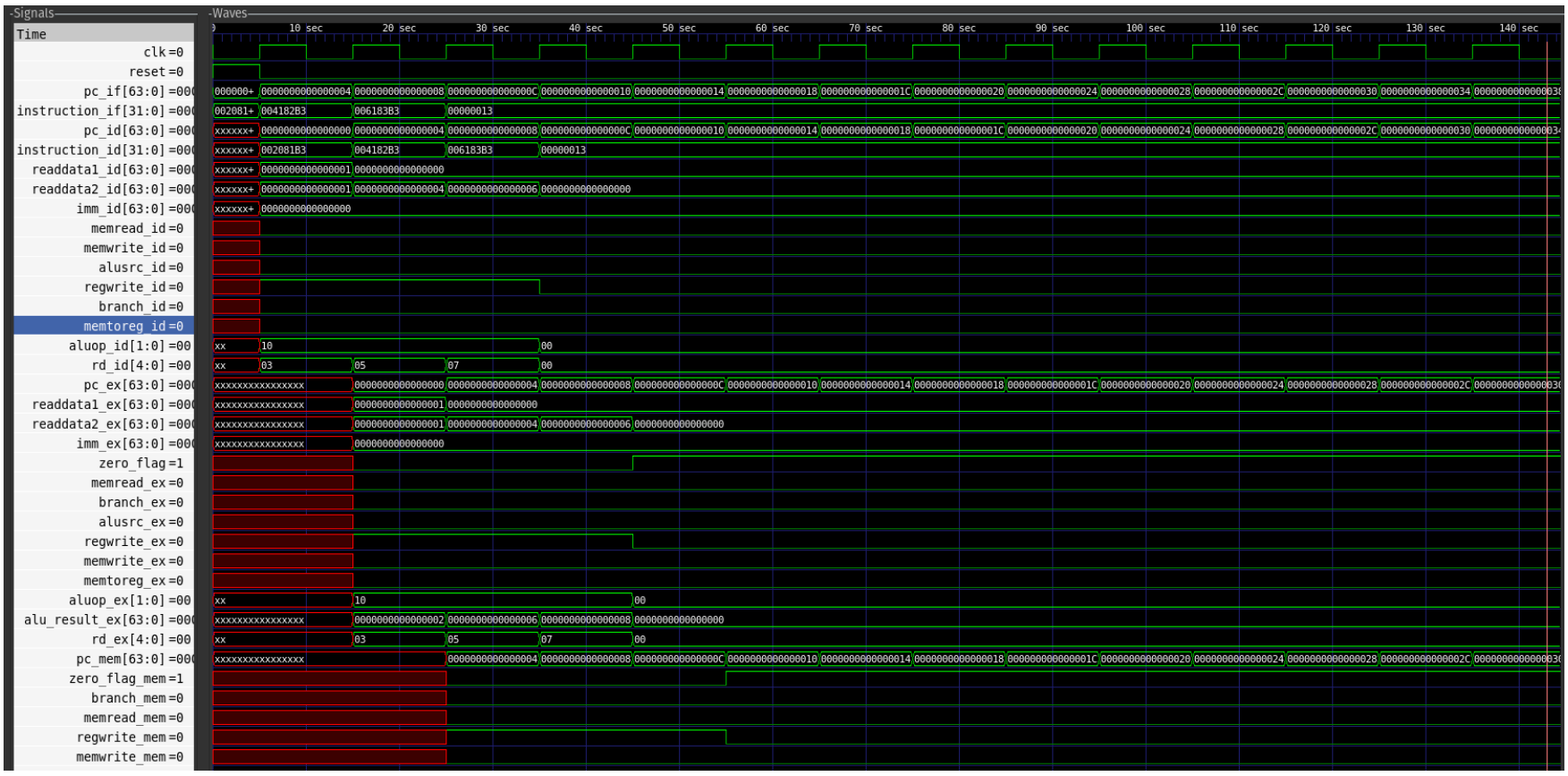
Clock Cycle - 4

```
================================================================================================================================================
Time: 35
IF Stage: PC_if = 000000000000000c | instruction_if = 00000021

ID Stage: PC_id = 0000000000000008 | instruction_id = 006183b3

Decode Stage: ReadData1 = 0000000000000000 | ReadData2 = 0000000000000006 | imm =                    0
Decode Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

Instruction=006183b3
Rs1= 3, Rs2= 6,Memread=0, idex_rd=00101, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = 03 | rs2 = 06 | rd = 07 | funct3 = 0 | funct7 = 00 | imm =                    0
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = 03 | rs2 = 04 | rd = 05 | funct3 = 0 | funct7 = 00 | imm =                    0
IDEX Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

EX Stage: PC_ex = 0000000000000004

Execute Inputs: ALUOp = 10 | funct7 = 00 | funct3 = 0 | read_data1 = 0000000000000000 | pc = 0000000000000004 | imm =                    0

Execute Outputs: ALU Result = 0000000000000006 | Zero Flag = 0 | Branch PC = 0000000000000008

EXMEM Inputs: Branch = 0 | MemRead = 0 | MemtoReg = 0 | MemWrite = 0 | RegWrite = 1 | PC = 0000000000000004 | ALU Result = 0000000000000006 | Zero Flag = 0 | rd = 05
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = 0000000000000004 | rd = 03 | Branch = 0 | MemRead = 0 | MemtoReg = 0 | MemWrite = 0 | RegWrite = 1 | ALU Result = 0000000000000002 | Zero Flag = 0

MEM Stage: ReadData = 0000000000000000 | ALU Result = 0000000000000002 | PC = 0000000000000004 | rd = 03

MEMWB Inputs: PC = 0000000000000004 | rd = 03 | MemtoReg = 0 | RegWrite = 1 | ALU Result = 0000000000000002 | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxx | ReadData = 0000000000000000

WB Stage: Write Data = xxxxxxxxxxxxxxx | RegWrite = x | rd = xx

================================================================================================================================================
```

Clock Cycle - 5

```
================================================================================================================================================
Time: 45
IF Stage: PC_if = 0000000000000010 | instruction_if = 00000013

ID Stage: PC_id = 000000000000000c | instruction_id = 00000021

Decode Stage: ReadData1 = 0000000000000000 | ReadData2 = 0000000000000000 | imm =                    0
Decode Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 00 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 0

Instruction=00000021
Rs1= 0, Rs2= 0,Memread=0, idex_rd=00111, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = 00 | rs2 = 00 | rd = 00 | funct3 = 0 | funct7 = 00 | imm =                    0
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = 03 | rs2 = 06 | rd = 07 | funct3 = 0 | funct7 = 00 | imm =                    0
IDEX Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

EX Stage: PC_ex = 0000000000000008

Execute Inputs: ALUOp = 10 | funct7 = 00 | funct3 = 0 | read_data1 = 0000000000000000 | pc = 0000000000000008 | imm =                    0

Execute Outputs: ALU Result = 0000000000000008 | Zero Flag = 0 | Branch PC = 000000000000000c

EXMEM Inputs: Branch = 0 | MemRead = 0 | MemtoReg = 0 | MemWrite = 0 | RegWrite = 1 | PC = 0000000000000008 | ALU Result = 0000000000000008 | Zero Flag = 0 | rd = 07
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = 0000000000000008 | rd = 05 | Branch = 0 | MemRead = 0 | MemtoReg = 0 | MemWrite = 0 | RegWrite = 1 | ALU Result = 0000000000000006 | Zero Flag = 0

MEM Stage: ReadData = 0000000000000000 | ALU Result = 0000000000000006 | PC = 0000000000000008 | rd = 05

MEMWB Inputs: PC = 0000000000000008 | rd = 05 | MemtoReg = 0 | RegWrite = 1 | ALU Result = 0000000000000006 | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = 03 | MemtoReg = 0 | RegWrite = 1 | ALU Result = 0000000000000002 | ReadData = 0000000000000000

WB Stage: Write Data = 0000000000000002 | RegWrite = 1 | rd = 03

================================================================================================================================================
```

Complete GTK Wave which handles Data Hazard:



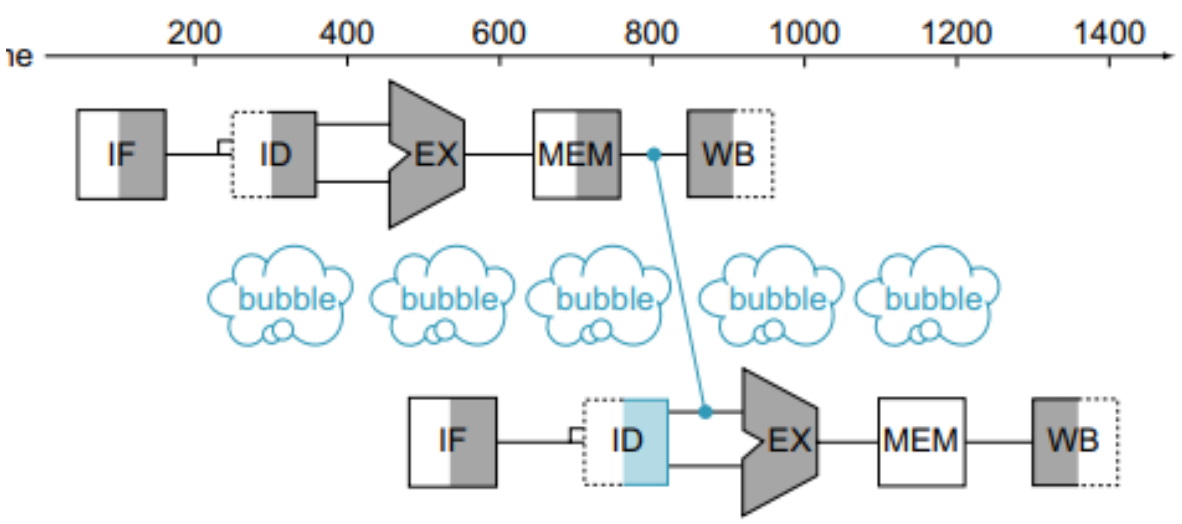
3. Load-Use Data Hazard

This hazard occurs when an instruction immediately depends on a value loaded from memory in the previous instruction. Since the data is not available until the MEM stage, the pipeline must **stall** the dependent instruction. For example:

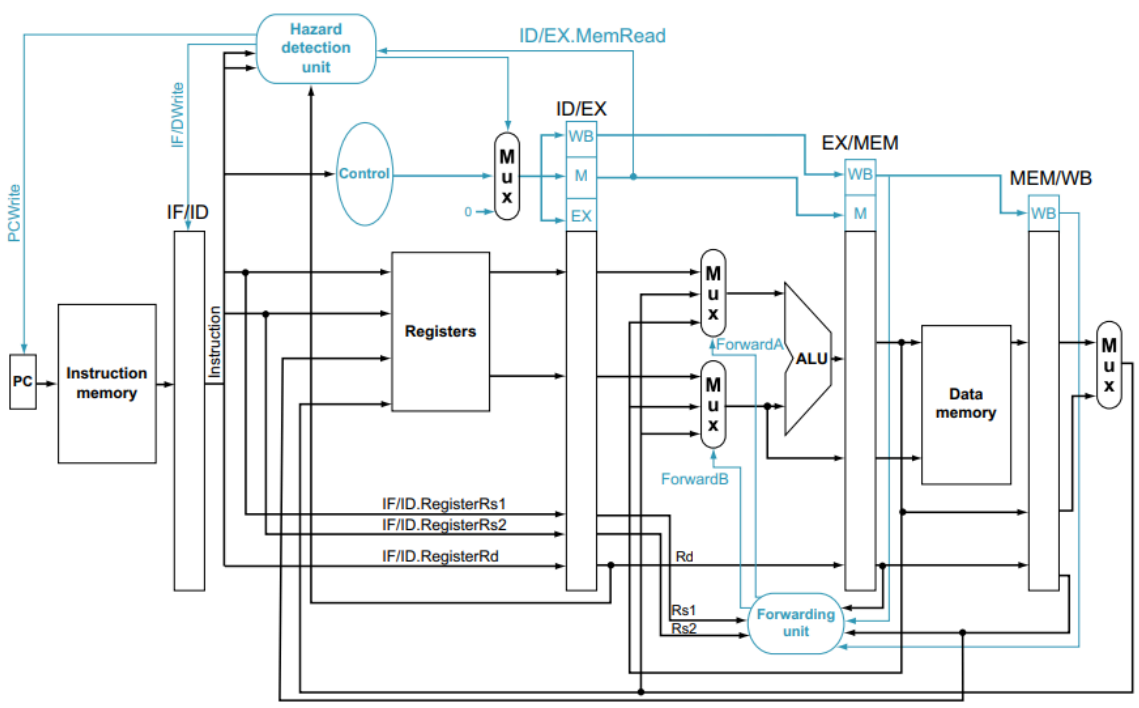
```
01043103  // ld x2, 16(x8)
00510233  // add x4, x2, x5
```

```
40610433  // sub x8, x2, x6
002204b3  // add x9, x4, x2
```

The `sub` instruction needs `x3` in the EX stage (cycle 4).However, `lw` writes `x3` in the MEM stage (cycle 4), so the value is not available yet. Here Forwarding does not work because memory access is not completed in time. So, the pipeline must stall the `sub` instruction for one cycle. After stalling the instruction, forwarding should take place.



Another part of the Datapath is the Hazard Detection unit. This unit controls the writing of the PC and IF/ID registers plus the multiplexor that chooses between the real control values and all 0s. The hazard detection unit stalls and deasserts the control fields if the load-use hazard test above is true.
So, the Datapath now is:



Example:

```
01043103  // ld x2, 16(x8)
00510233  // add x4, x2, x5
40610433  // sub x8, x2, x6
002204b3  // add x9, x4, x2
```

Clock Cycle - 1

```
================================================================================================================================================
Time: 5
IF Stage: PC_if = 0000000000000000 | instruction_if = 01043103

ID Stage: PC_id = xxxxxxxxxxxxxxxx | instruction_id = xxxxxxxx

Decode Stage: ReadData1 = xxxxxxxxxxxxxxxx | ReadData2 = xxxxxxxxxxxxxxxx | imm =                    x
Decode Control Signals: Branch = x | MemRead = x | MemtoReg = x | ALUOp = xx | MemWrite = x | ALUSrc = x | RegWrite = x

Instruction=xxxxxxxx
Rs1= x, Rs2= x,Memread=x, idex_rd=xxxxx, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = xx | rs2 = xx | rd = xx | funct3 = x | funct7 = xx | imm =                    x
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = xx | rs2 = xx | rd = xx | funct3 = x | funct7 = xx | imm =                     x
IDEX Control Signals: Branch = x | MemRead = x | MemtoReg = x | ALUOp = xx | MemWrite = x | ALUSrc = x | RegWrite = x

EX Stage: PC_ex = xxxxxxxxxxxxxxxx

Execute Inputs: ALUOp = xx | funct7 = xx | funct3 = x | read_data1 = xxxxxxxxxxxxxxxx | pc = xxxxxxxxxxxxxxxx | imm =                    x

Execute Outputs: ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x | Branch PC = xxxxxxxxxxxxxxxx

EXMEM Inputs: Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | PC = xxxxxxxxxxxxxxxx | ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x | rd = xx
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = xxxxxxxxxxxxxxxx | rd = xx | Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x

MEM Stage: ReadData = 0000000000000000 | ALU Result = xxxxxxxxxxxxxxxx | PC = xxxxxxxxxxxxxxxx | rd = xx

MEMWB Inputs: PC = xxxxxxxxxxxxxxxx | rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = 00 | MemtoReg = 0 | RegWrite = 0 | ALU Result = 0000000000000000 | ReadData = 0000000000000000

WB Stage: Write Data = 0000000000000000 | RegWrite = 0 | rd = 00

================================================================================================================================================
```

Clock Cycle - 2

```
================================================================================================================================================
Time: 15
IF Stage: PC_if = 0000000000000004 | instruction_if = 00510233

ID Stage: PC_id = 0000000000000000 | instruction_id = 01043103

Decode Stage: ReadData1 = 0000000000000008 | ReadData2 = 0000000000000000 | imm =                   16
Decode Control Signals: Branch = 0 | MemRead = 1 | MemtoReg = 1 | ALUOp = 00 | MemWrite = 0 | ALUSrc = 1 | RegWrite = 1

Instruction=01043103
Rs1= 8, Rs2=16,Memread=x, idex_rd=xxxxx, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = 08 | rs2 = 10 | rd = 02 | funct3 = 3 | funct7 = 00 | imm =                  16
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = xx | rs2 = xx | rd = xx | funct3 = x | funct7 = xx | imm =                    x
IDEX Control Signals: Branch = x | MemRead = x | MemtoReg = x | ALUOp = xx | MemWrite = x | ALUSrc = x | RegWrite = x

EX Stage: PC_ex = xxxxxxxxxxxxxxxx

Execute Inputs: ALUOp = xx | funct7 = xx | funct3 = x | read_data1 = xxxxxxxxxxxxxxxx | pc = xxxxxxxxxxxxxxxx | imm =                    x

Execute Outputs: ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x | Branch PC = xxxxxxxxxxxxxxxx

EXMEM Inputs: Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | PC = xxxxxxxxxxxxxxxx | ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x | rd = xx
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = xxxxxxxxxxxxxxxx | rd = xx | Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x

MEM Stage: ReadData = 0000000000000000 | ALU Result = xxxxxxxxxxxxxxxx | PC = xxxxxxxxxxxxxxxx | rd = xx

MEMWB Inputs: PC = xxxxxxxxxxxxxxxx | rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | ReadData = 0000000000000000

WB Stage: Write Data = xxxxxxxxxxxxxxxx | RegWrite = x | rd = xx

================================================================================================================================================
```

Clock Cycle - 3

```
================================================================================================================================================
Time: 25
IF Stage: PC_if = 0000000000000008 | instruction_if = 40610433

ID Stage: PC_id = 0000000000000004 | instruction_id = 00510233

Decode Stage: ReadData1 = 0000000000000001 | ReadData2 = 0000000000000005 | imm =                    0
Decode Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 00 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 0

Instruction=00510233
Rs1= 2, Rs2= 5,Memread=1, idex_rd=00010, pc write: 0, ifid write=0, bubble=1

IDEX Inputs: rs1 = 02 | rs2 = 05 | rd = 04 | funct3 = 0 | funct7 = 00 | imm =                    0
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = 08 | rs2 = 10 | rd = 02 | funct3 = 3 | funct7 = 00 | imm =                   16
IDEX Control Signals: Branch = 0 | MemRead = 1 | MemtoReg = 1 | ALUOp = 00 | MemWrite = 0 | ALUSrc = 1 | RegWrite = 1

EX Stage: PC_ex = 0000000000000000

Execute Inputs: ALUOp = 00 | funct7 = 00 | funct3 = 3 | read_data1 = 0000000000000008 | pc = 0000000000000000 | imm =                   16

Execute Outputs: ALU Result = 0000000000000018 | Zero Flag = 0 | Branch PC = 0000000000000004

EXMEM Inputs: Branch = 0 | MemRead = 1 | MemtoReg = 1 | MemWrite = 0 | RegWrite = 1 | PC = 0000000000000000 | ALU Result = 0000000000000018 | Zero Flag = 0 | rd = 02
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = xxxxxxxxxxxxxxxx | rd = xx | Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x

MEM Stage: ReadData = 0000000000000000 | ALU Result = xxxxxxxxxxxxxxxx | PC = xxxxxxxxxxxxxxxx | rd = xx

MEMWB Inputs: PC = xxxxxxxxxxxxxxxx | rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | ReadData = 0000000000000000

WB Stage: Write Data = xxxxxxxxxxxxxxxx | RegWrite = x | rd = xx

================================================================================================================================================
```

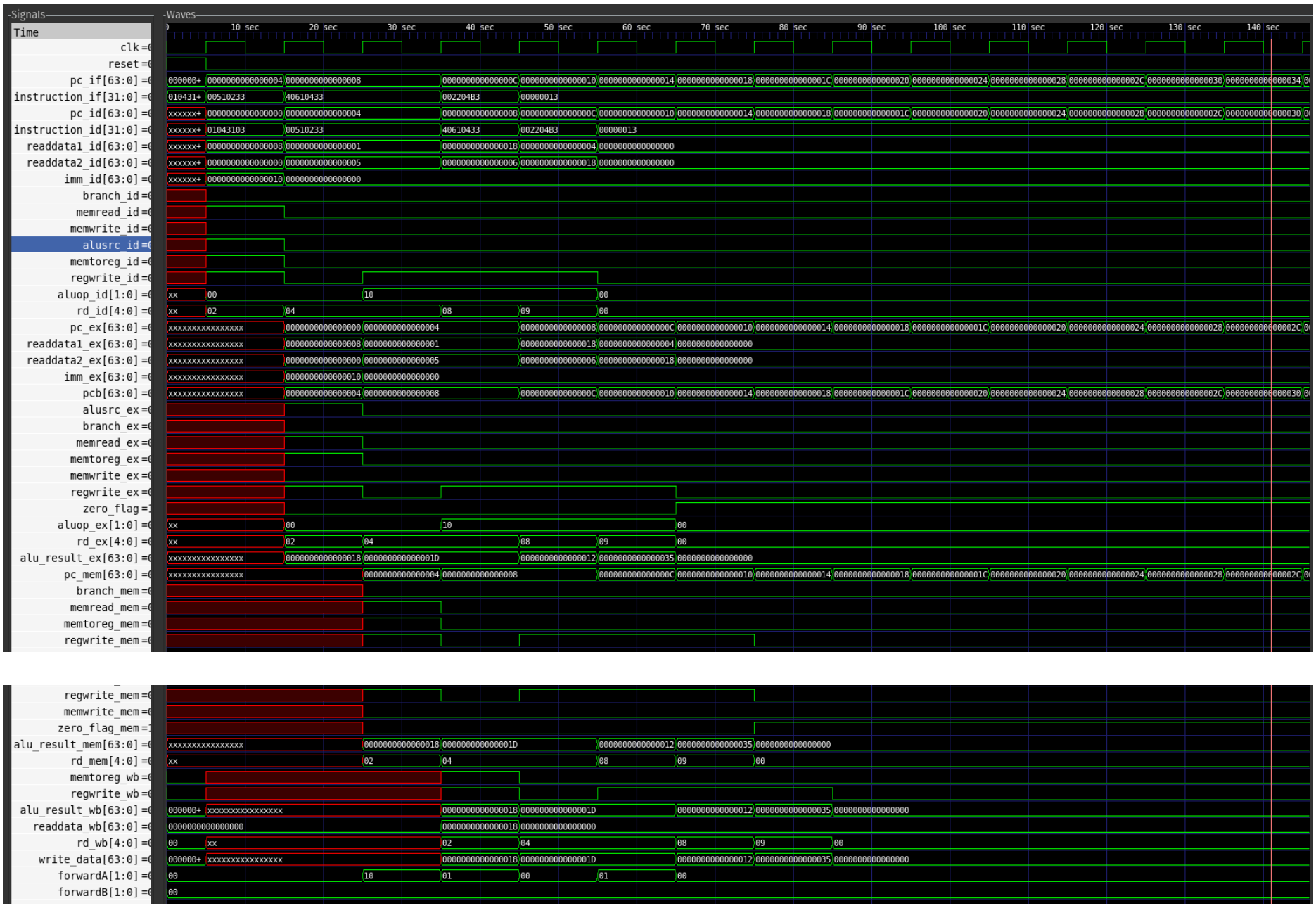
Clock Cycle - 4

```
================================================================================================================================================
Time: 35
IF Stage: PC_if = 0000000000000008 | instruction_if = 40610433

ID Stage: PC_id = 0000000000000004 | instruction_id = 00510233

Decode Stage: ReadData1 = 0000000000000001 | ReadData2 = 0000000000000005 | imm =                    0
Decode Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

Instruction=00510233
Rs1= 2, Rs2= 5,Memread=0, idex_rd=00100, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = 02 | rs2 = 05 | rd = 04 | funct3 = 0 | funct7 = 00 | imm =                   0
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = 02 | rs2 = 05 | rd = 04 | funct3 = 0 | funct7 = 00 | imm =                   0
IDEX Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 00 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 0

EX Stage: PC_ex = 0000000000000004

Execute Inputs: ALUOp = 00 | funct7 = 00 | funct3 = 0 | read_data1 = 0000000000000001 | pc = 0000000000000004 | imm =                    0

Execute Outputs: ALU Result = 000000000000001d | Zero Flag = 0 | Branch PC = 0000000000000008

EXMEM Inputs: Branch = 0 | MemRead = 0 | MemtoReg = 0 | MemWrite = 0 | RegWrite = 0 | PC = 0000000000000004 | ALU Result = 000000000000001d | Zero Flag = 0 | rd = 04
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = 0000000000000004 | rd = 02 | Branch = 0 | MemRead = 1 | MemtoReg = 1 | MemWrite = 0 | RegWrite = 1 | ALU Result = 0000000000000018 | Zero Flag = 0

MEM Stage: ReadData = 0000000000000018 | ALU Result = 0000000000000018 | PC = 0000000000000004 | rd = 02

MEMWB Inputs: PC = 0000000000000004 | rd = 02 | MemtoReg = 1 | RegWrite = 1 | ALU Result = 0000000000000018 | ReadData = 0000000000000018
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | ReadData = 0000000000000000

WB Stage: Write Data = xxxxxxxxxxxxxxxx | RegWrite = x | rd = xx

================================================================================================================================================
```

Clock Cycle - 5

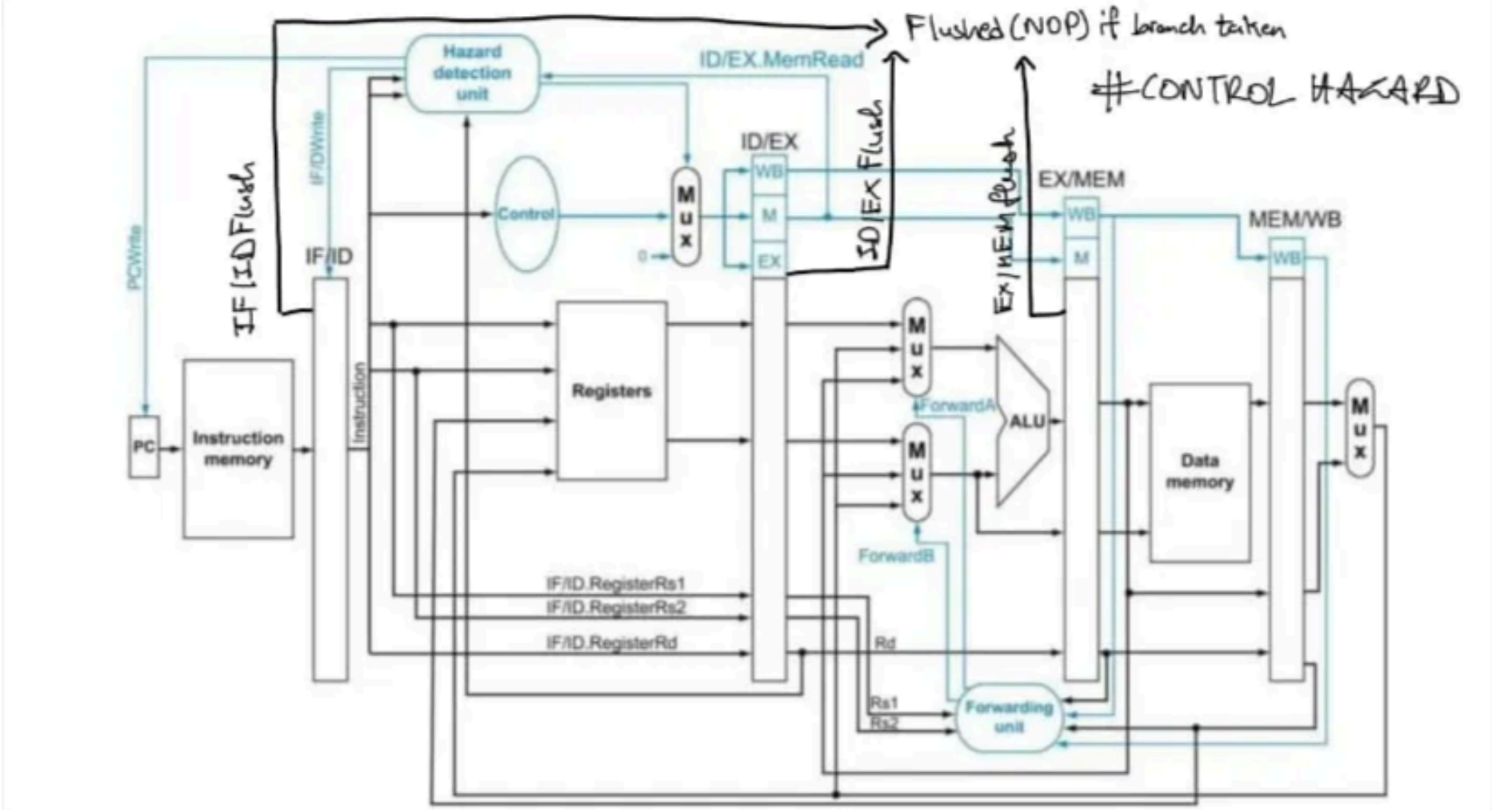
```
================================================================================================================================================
Time: 45
IF Stage: PC_if = 000000000000000c | instruction_if = 002204b3

ID Stage: PC_id = 0000000000000008 | instruction_id = 40610433

Decode Stage: ReadData1 = 0000000000000018 | ReadData2 = 0000000000000006 | imm =                    0
Decode Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

Instruction=40610433
Rs1= 2, Rs2= 6,Memread=0, idex_rd=00100, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = 02 | rs2 = 06 | rd = 08 | funct3 = 0 | funct7 = 20 | imm =                   0
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = 02 | rs2 = 05 | rd = 04 | funct3 = 0 | funct7 = 00 | imm =                   0
IDEX Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 10 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 1

EX Stage: PC_ex = 0000000000000004

Execute Inputs: ALUOp = 10 | funct7 = 00 | funct3 = 0 | read_data1 = 0000000000000001 | pc = 0000000000000004 | imm =                    0

Execute Outputs: ALU Result = 000000000000001d | Zero Flag = 0 | Branch PC = 0000000000000008

EXMEM Inputs: Branch = 0 | MemRead = 0 | MemtoReg = 0 | MemWrite = 0 | RegWrite = 1 | PC = 0000000000000004 | ALU Result = 000000000000001d | Zero Flag = 0 | rd = 04
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = 0000000000000008 | rd = 04 | Branch = 0 | MemRead = 0 | MemtoReg = 0 | MemWrite = 0 | RegWrite = 0 | ALU Result = 000000000000001d | Zero Flag = 0

MEM Stage: ReadData = 0000000000000000 | ALU Result = 000000000000001d | PC = 0000000000000008 | rd = 04

MEMWB Inputs: PC = 0000000000000008 | rd = 04 | MemtoReg = 0 | RegWrite = 0 | ALU Result = 000000000000001d | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = 02 | MemtoReg = 1 | RegWrite = 1 | ALU Result = 0000000000000018 | ReadData = 0000000000000018

WB Stage: Write Data = 0000000000000018 | RegWrite = 1 | rd = 02

================================================================================================================================================
```

Complete GTK Wave which handles Load Use Data Hazard:



4. Control Hazards

A control hazard (or branch hazard) occurs when the pipeline does not know which instruction to fetch next due to a control flow change (e.g., branch). Since instruction fetching happens in every cycle, any uncertainty in the program flow can lead to wasted cycles or incorrect execution. Here we are checking the control hazards which occur when executing conditional branches i.e. beq.



Example for the control hazard is:

```
00208463  // beq x1, x2, 8
005201B3  // add x3, x4, x5
01043103  // ld x2, 16(x8)
00510233  // add x4, x2, x5
40610433  // sub x8, x2, x6
002204B3  // add x9, x4, x2
```

In the above example the processor does not know whether to execute `add x3, x4, x5` or fetch from the branch at 8 until the `beq` instruction completes the EX stage.
Solution for the control hazard is :

Static branch prediction: The processor always assumes that the branch is not taken. If the branch is taken, the incorrectly fetched instructions are flushed (discarded).

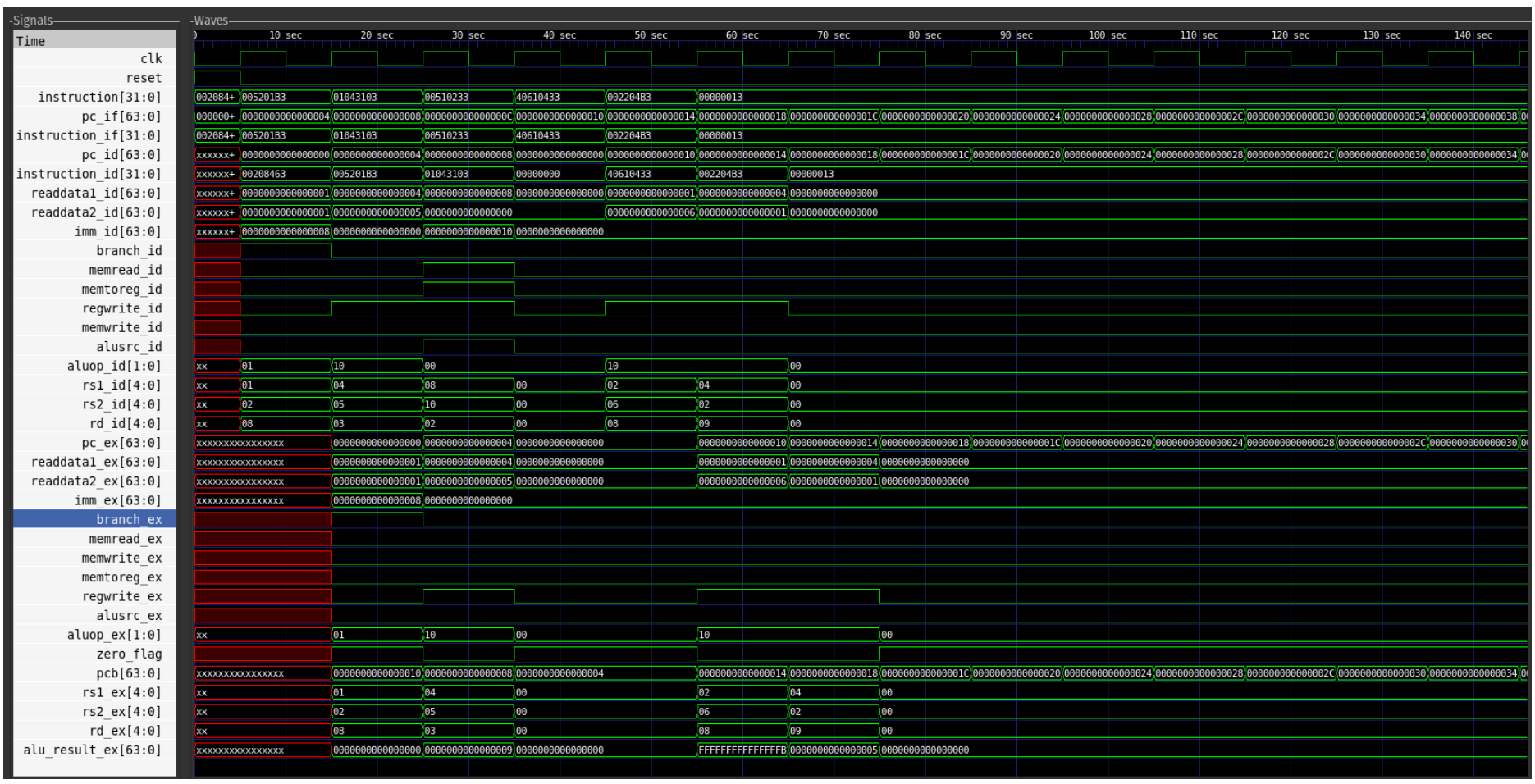
```
================================================================================================================================================
Time: 5
IF Stage: PC_if = 0000000000000000 | instruction_if = 00208463

ID Stage: PC_id = xxxxxxxxxxxxxxxx | instruction_id = xxxxxxxx

Decode Stage: ReadData1 = xxxxxxxxxxxxxxxx | ReadData2 = xxxxxxxxxxxxxxxx | imm =                    x
Decode Control Signals: Branch = x | MemRead = x | MemtoReg = x | ALUOp = xx | MemWrite = x | ALUSrc = x | RegWrite = x

Instruction=xxxxxxxx
Rs1= x, Rs2= x,Memread=x, idex_rd=xxxxx, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = xx | rs2 = xx | rd = xx | funct3 = x | funct7 = xx | imm =                    x
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = xx | rs2 = xx | rd = xx | funct3 = x | funct7 = xx | imm =                    x
IDEX Control Signals: Branch = x | MemRead = x | MemtoReg = x | ALUOp = xx | MemWrite = x | ALUSrc = x | RegWrite = x

EX Stage: PC_ex = xxxxxxxxxxxxxxxx

Execute Inputs: ALUOp = xx | funct7 = xx | funct3 = x | read_data1 = xxxxxxxxxxxxxxxx | pc = xxxxxxxxxxxxxxxx | imm =                    x

Execute Outputs: ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x | Branch PC = xxxxxxxxxxxxxxxx

EXMEM Inputs: Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | PC = xxxxxxxxxxxxxxxx | ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x | rd = xx
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = xxxxxxxxxxxxxxxx | rd = xx | Branch = x | MemRead = x | MemtoReg = x | MemWrite = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | Zero Flag = x

MEM Stage: ReadData = 0000000000000000 | ALU Result = xxxxxxxxxxxxxxxx | PC = xxxxxxxxxxxxxxxx | rd = xx

MEMWB Inputs: PC = xxxxxxxxxxxxxxxx | rd = xx | MemtoReg = x | RegWrite = x | ALU Result = xxxxxxxxxxxxxxxx | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = 00 | MemtoReg = 0 | RegWrite = 0 | ALU Result = 0000000000000000 | ReadData = 0000000000000000

WB Stage: Write Data = 0000000000000000 | RegWrite = 0 | rd = 00

================================================================================================================================================
```

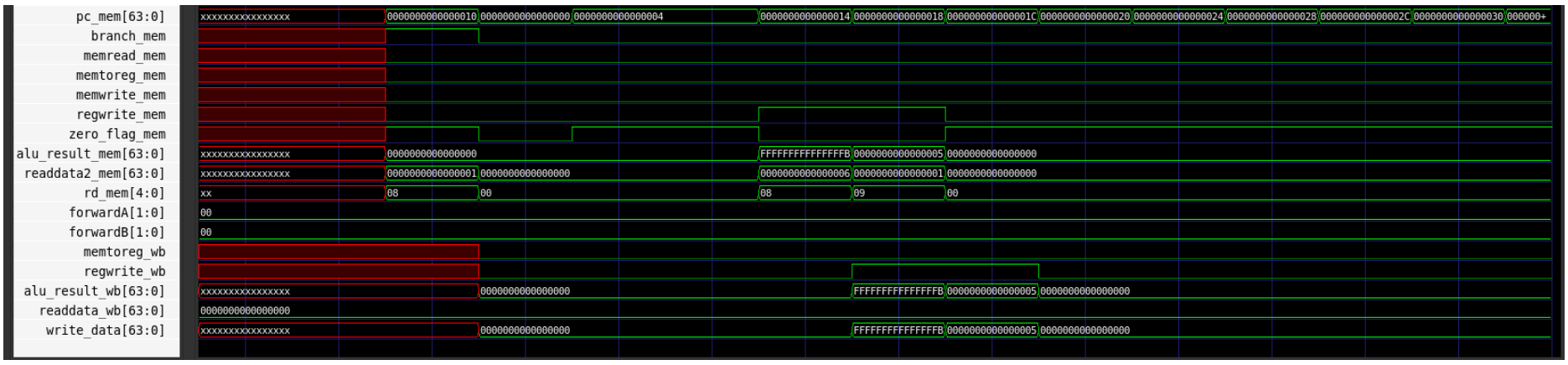
```
================================================================================================================================================
Time: 45
IF Stage: PC_if = 0000000000000010 | instruction_if = 40610433

ID Stage: PC_id = 0000000000000000 | instruction_id = 00000000

Decode Stage: ReadData1 = 0000000000000000 | ReadData2 = 0000000000000000 | imm =                    0
Decode Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 00 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 0

Instruction=00000000
Rs1= 0, Rs2= 0,Memread=0, idex_rd=00000, pc write: 1, ifid write=1, bubble=0

IDEX Inputs: rs1 = 00 | rs2 = 00 | rd = 00 | funct3 = 0 | funct7 = 00 | imm =                   0
------------------------------------------------------------------------------------------------------------------------------------------------
IDEX Outputs: rs1 = 00 | rs2 = 00 | rd = 00 | funct3 = 0 | funct7 = 00 | imm =                    0
IDEX Control Signals: Branch = 0 | MemRead = 0 | MemtoReg = 0 | ALUOp = 00 | MemWrite = 0 | ALUSrc = 0 | RegWrite = 0

EX Stage: PC_ex = 0000000000000000

Execute Inputs: ALUOp = 00 | funct7 = 00 | funct3 = 0 | read_data1 = 0000000000000000 | pc = 0000000000000000 | imm =                    0

Execute Outputs: ALU Result = 0000000000000000 | Zero Flag = 1 | Branch PC = 0000000000000004

EXMEM Inputs: Branch = 0 | MemRead = 0 | MemtoReg = 0 | MemWrite = 0 | RegWrite = 0 | PC = 0000000000000000 | ALU Result = 0000000000000000 | Zero Flag = 1 | rd = 00
------------------------------------------------------------------------------------------------------------------------------------------------
EXMEM Outputs: PC = 0000000000000000 | rd = 00 | Branch = 0 | MemRead = 0 | MemtoReg = 0 | MemWrite = 0 | RegWrite = 0 | ALU Result = 0000000000000000 | Zero Flag = 0

MEM Stage: ReadData = 0000000000000000 | ALU Result = 0000000000000000 | PC = 0000000000000000 | rd = 00

MEMWB Inputs: PC = 0000000000000000 | rd = 00 | MemtoReg = 0 | RegWrite = 0 | ALU Result = 0000000000000000 | ReadData = 0000000000000000
------------------------------------------------------------------------------------------------------------------------------------------------
MEMWB Outputs: rd = 08 | MemtoReg = 0 | RegWrite = 0 | ALU Result = 0000000000000000 | ReadData = 0000000000000000

WB Stage: Write Data = 0000000000000000 | RegWrite = 0 | rd = 08

================================================================================================================================================
```

The processor detects the branch instruction (and the zero flag) and jumps to the address 16 which is twice the offset of the first instruction. As the branch is taken, all the control signals in ID/EX register and the previous instructions get flushed i.e. they become zero. So in this way the update of PC and IF/ID register is prevented. The present instruction is decoded again and the following instruction is fetched again.

GTK Wave output for this Hazard Handling:

```
00208463  // beq x1, x2, 8
005201B3  // add x3, x4, x5
01043103  // ld x2, 16(x8)
00510233  // add x4, x2, x5
40610433  // sub x8, x2, x6
002204B3  // add x9, x4, x2
```





Handling Different Combinations of Hazards:

Lode Use Data Hazard + Branch Hazard

This hazard is handled by **stalling**, where the processor detects that `beq x11, x10, 12` depends on the result of `lw x10, 4(x15)`, which has not yet completed. Since `x10` is updated only in the **write-back (WB) stage** of `lw`, the pipeline control unit inserts **two stall cycles** before executing `beq`. These stalls act as **pipeline bubbles**, preventing `beq` from using an incorrect value of `x10`. After the stalls, `beq` can proceed safely with the correct data, ensuring proper execution while maintaining program correctness.

```
0047A503  // Lw x10,4(x15)
00a58663  // Beq x11,x10,12
00C68633  // Add x12,x13,x12
00B60533  // Add x10,x12,x11
```

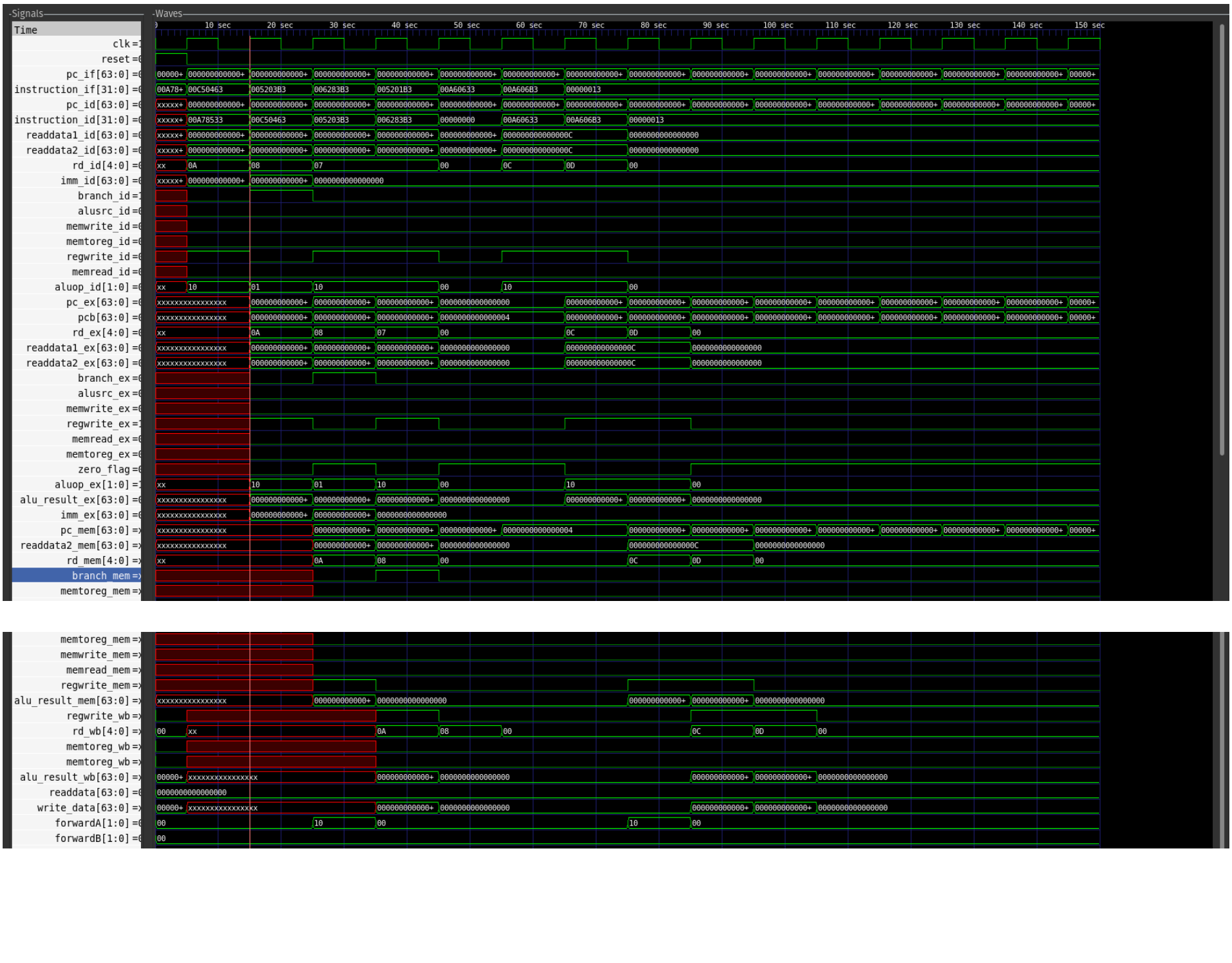
The GTKWave (for the above case of control hazard)

Forwarding + Branch Hazard

```
00A78533  // Add x10,x15,x10
00C50463  // Beq x10,x12,8
005203b3  // add x7, x4,x5
```

```
006283b3  // add x7, x5,x6
005201b3  // add x3, x4, x5
00a60633  // add x12,x12,x10
00a606b3  // add x13,x12,x10
```

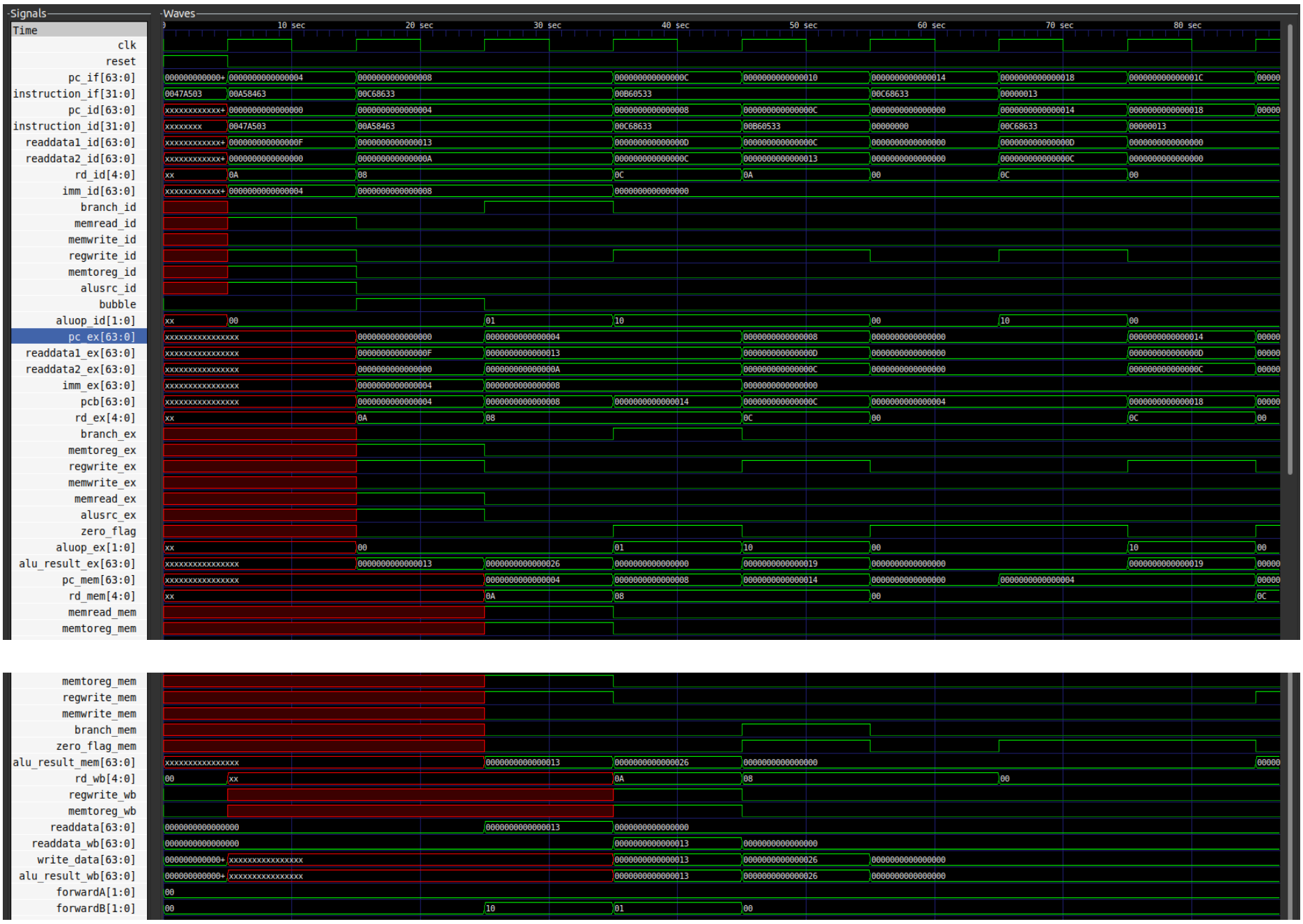
This hazard is handled using **forwarding**, where the result of `add x10, x15, x10` is sent directly from the **EX/MEM stage** to the **EX stage** of `beq x10, x12, 8`, avoiding the need to wait for write-back. Similarly, for `add x12, x12, x10` and `add x13, x12, x10`, the values of `x10` and `x12` are forwarded from earlier pipeline stages to ensure they use the correct updated values without stalling. This technique improves efficiency by reducing pipeline delays while maintaining correct execution.



Lode Use Data Hazard + Branch Hazard

```
0047A503  // Lw x10,4(x15)
00A58463  // beq x11, x10, 8
00C68633  // Add x12,x13,x12
00B60533  // add x10,x12,x11
00B60533  // add x10,x12,x11
00C68633  // Add x12,x13,x12
```

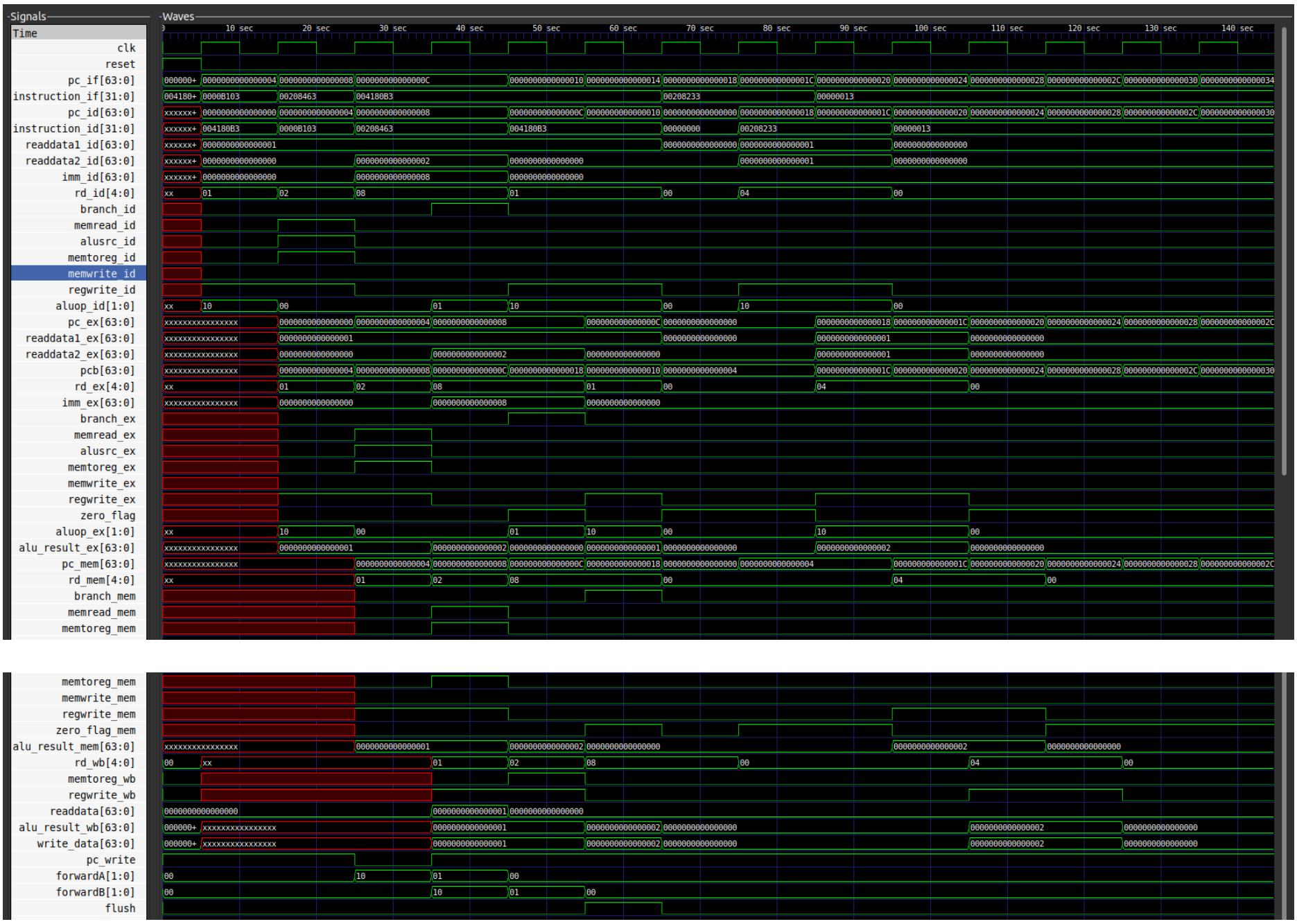
This hazard is handled by **stalling**, as `beq x11, x10, 8` depends on `lw x10, 4(x15)`, which completes in the **WB stage**. Since `x10` is not available when `beq` reaches the **ID stage**, the processor inserts **two stall cycles**, preventing incorrect execution. These stalls allow `lw` to complete its memory access and write-back, ensuring `beq` compares the correct value of `x10`. Once the stalls resolve the hazard, execution continues normally with subsequent `add` instructions.



Forwarding + Load Use Data Hazard + Branch Hazard

```
004180b3 //add x1,x3,x4
0000b103 //ld x2, 0(x1)
00208463 //beq x1,x2,8
004180b3 //add x1,x3,x4
004180b3 //add x1,x3,x4
004180b3 //add x1,x3,x4
00208233 //add x4,x1,x2
00208233 //add x4,x1,x2
```

In this RISC-V instruction sequence, **data forwarding** helps resolve dependencies between `add` and `ld`. The instruction `add x1, x3, x4` computes a result and stores it in `x1`, which is immediately used by `ld x2, 0(x1)`. Instead of waiting for `x1` to be written back, forwarding allows `ld` to receive `x1` directly from the execute stage, avoiding a stall. However, a **load-use data hazard** arises between `ld x2, 0(x1)` and `beq x1, x2, 8` since `beq` needs `x2` before it is available. Since `ld` completes in the memory stage and `beq` needs `x2` in decode, a stall is required unless forwarding is available. Additionally, `beq` introduces a **branch hazard** because the decision to branch is only resolved in the execute stage. Without branch prediction, the processor must stall until the branch outcome is determined, potentially flushing incorrect instructions if the prediction is wrong.



Contributions:

1. **Sanjana Sheela**
   - Sequential - main.v, processor_tb.v, Writeback and Mux
   - Combining Pipeline registers to the Sequential Implementation
   - Branch Hazard

2. **Snigdha Stp**
   - Sequential - Fetch, Execute and ALU Modules
   - Pipeline Registers
   - Load Use Data Hazard
   - Branch Hazard

3. **Khyathi Sri**
   - Sequential - Register File, Decode and Memory Modules
   - Testing and Debugging of both implementations
   - Forwarding (Data hazard)
   - Load Use Data Hazard

→ Report has been equally contributed by all.