BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

DHAKA, BANGLADESH

4-bit MIPS

Course No: CSE306

Couse Title: Computer Architecture Sessional

Session: July,2022

Submitted By:

1905042- Rakibul Hasan Rafi

1905046- Niaz Rahman

1905047- Rakib Abdullah

1905048- Md. Al-Amin Sany

1905052- Bijoy Ahmed Saiem

Introduction:

MIPS is a RISC ISA. The full form of MIPS is **M**icroprocessor without **I**nterlocked **P**ipeline **S**tages whereas the full form of RISC is **R**educed **I**nstruction **S**et **A**rchitecture. ISA means **I**nstruction **S**et **A**rchitecture. The instructions of MIPS are fixed and rigid which ensures regularity.

According to MIPS instruction properties, the size of a register is 32 bits and there are 32 registers. There are different types of instruction format for different types of instruction.

R-Format:

op	rs	rt	rd	shamt	func
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Here, op = opcode to denote operation type and format
rs = first source register
rt = second source register
rd = destination register
shamt = shift amount
funct = function code to denote specific variant of an operation

I-Format:

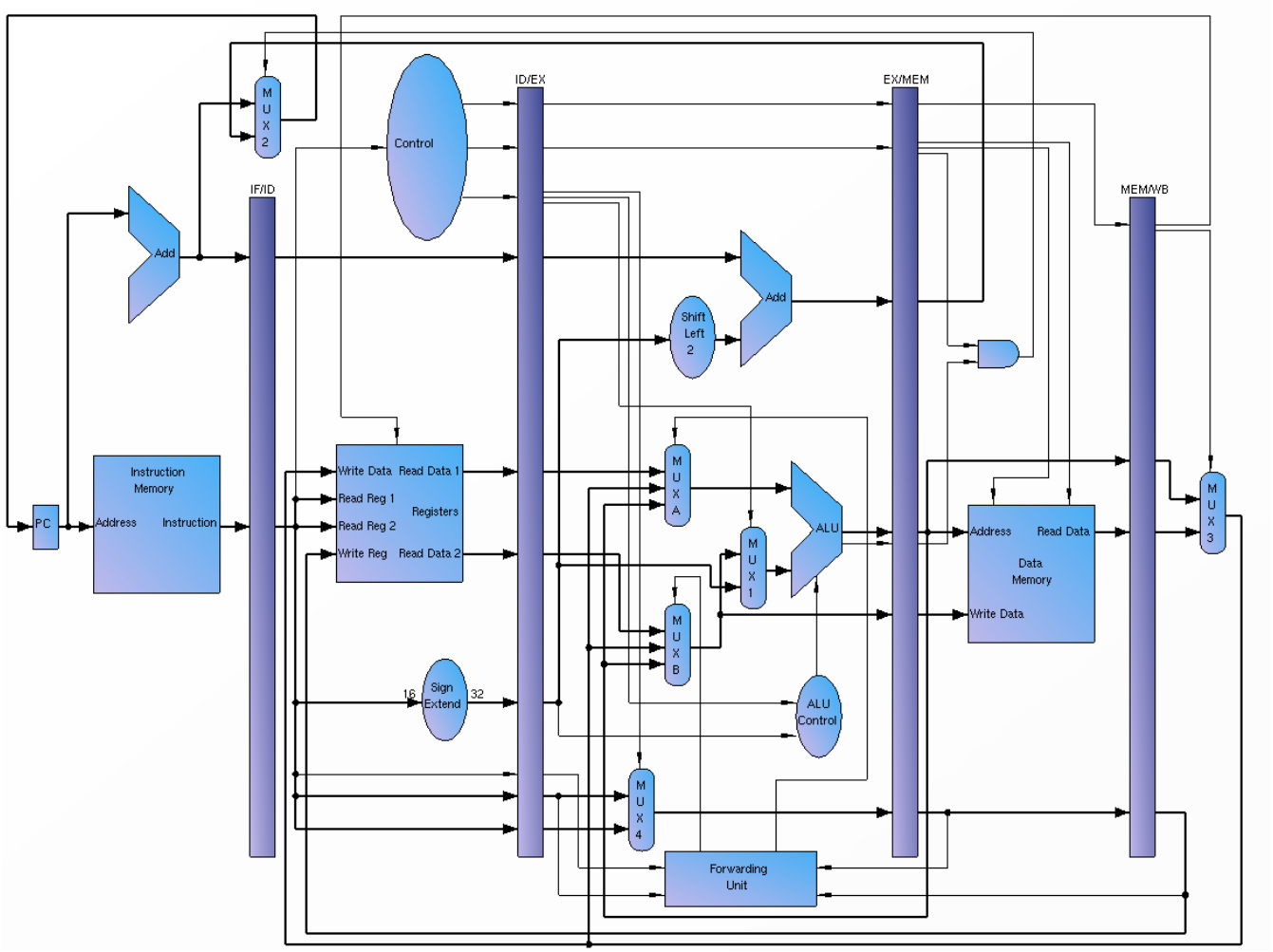
op	rs	rt	constant or address
6 bits	5 bits	5 bits	16 bits

Here, op = opcode to denote operation type and format
rs = first source register
rt = destination register
constant or address = the constant or address to store/load the value to/from

J-Format:

op	jumping address
6 bits	26 bits

Here, op = opcode to denote operation type and format
Jumping address= address of target



It is the datapath of MIPS. Though in real, MIPS use pipeline, we are ignoring pipelining for simplicity.

A datapath is built with elements that process data and address. The instructions are fed through a datapath to perform different instructions. A datapath also depicts the flow of data through different components of a computer. The control unit is microprogrammed and the control signals associated with the operations are stored in memory. All instruction need 1 clock cycle to be executed in our case.

The main components of MIPS:

ALU, Register File, Instruction Memory, Data Memory and a Control Unit.

Problem Specification:

Instruction Set:

Instruction ID	Instruction Type	Instruction
H	Logic	ori
O	Control	bneq
G	Logic	or
K	Logic	nor
D	Arithmetic	subi
J	Logic	srl
I	Logic	sll
N	Control	beq
L	Memory	lw
B	Arithmetic	addi
F	Logic	andi
A	Arithmetic	add
M	Memory	sw
C	Arithmetic	sub
P	Control	j
E	Logic	and

Instruction Format:

R-type:

Opcode	Src Reg 1	Src Reg 2	Dst Reg
4 bits	4 bits	4 bits	4 bits

S-type:

Opcode	Src Reg 1	Dst Reg	Shamt

I-type:

Opcode	Src Reg 1	Src reg 2/ Dst Reg	Addr./Immdt
4 bits	4 bits	4 bits	4 bits

J-type:

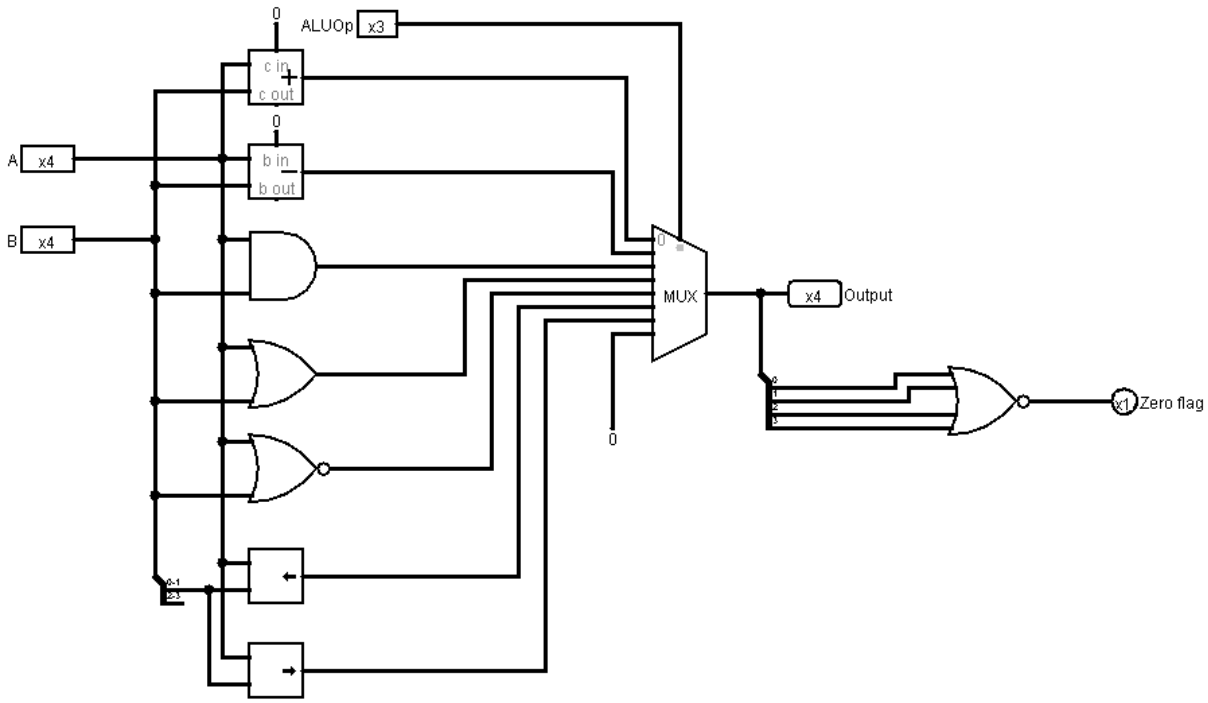
Opcode	Target Jump Address	0
4 bits	8 bits	4 bits

Circuit Diagram:

1.ALU:

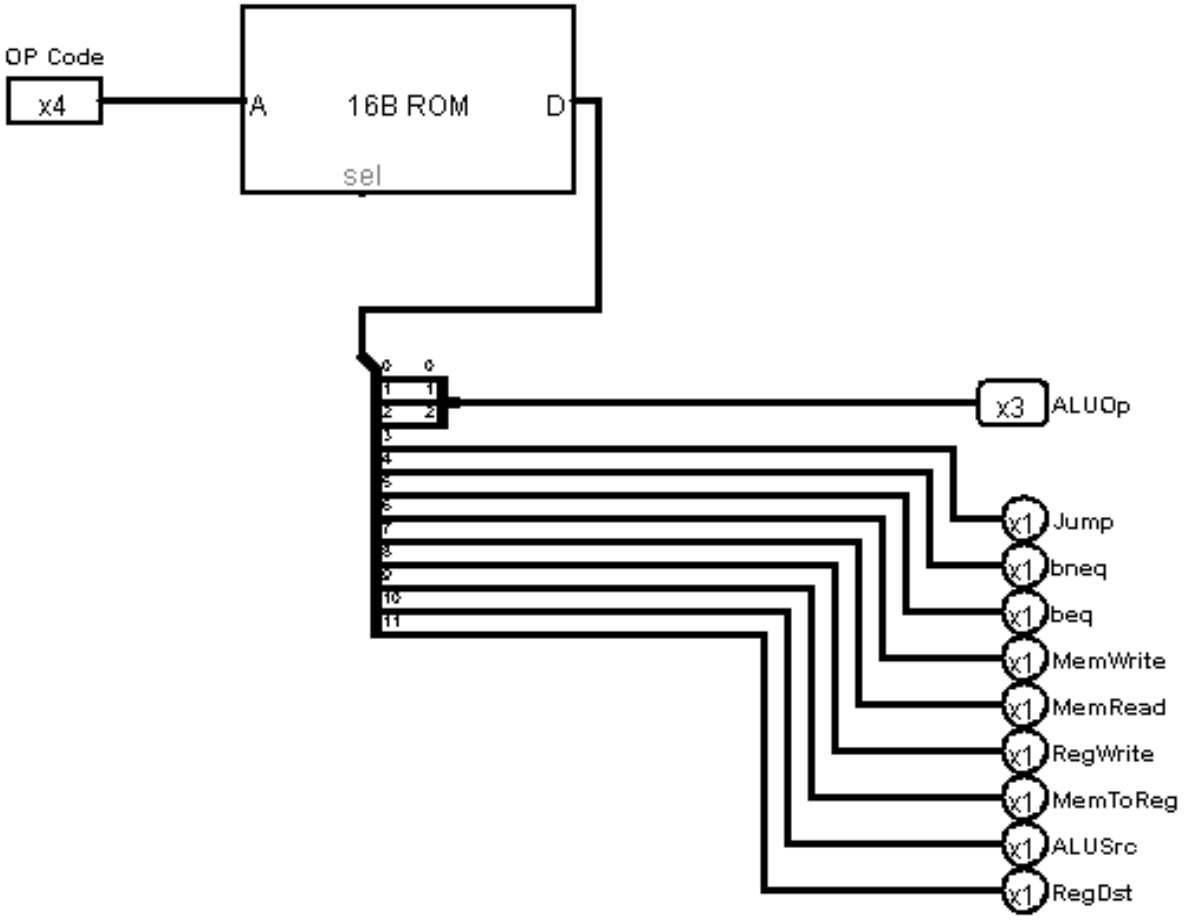
This circuit compute the arithmetic and logical part of the main circuit. There are seven operation-

- add
- sub
- or
- and
- nor
- sll
- srl



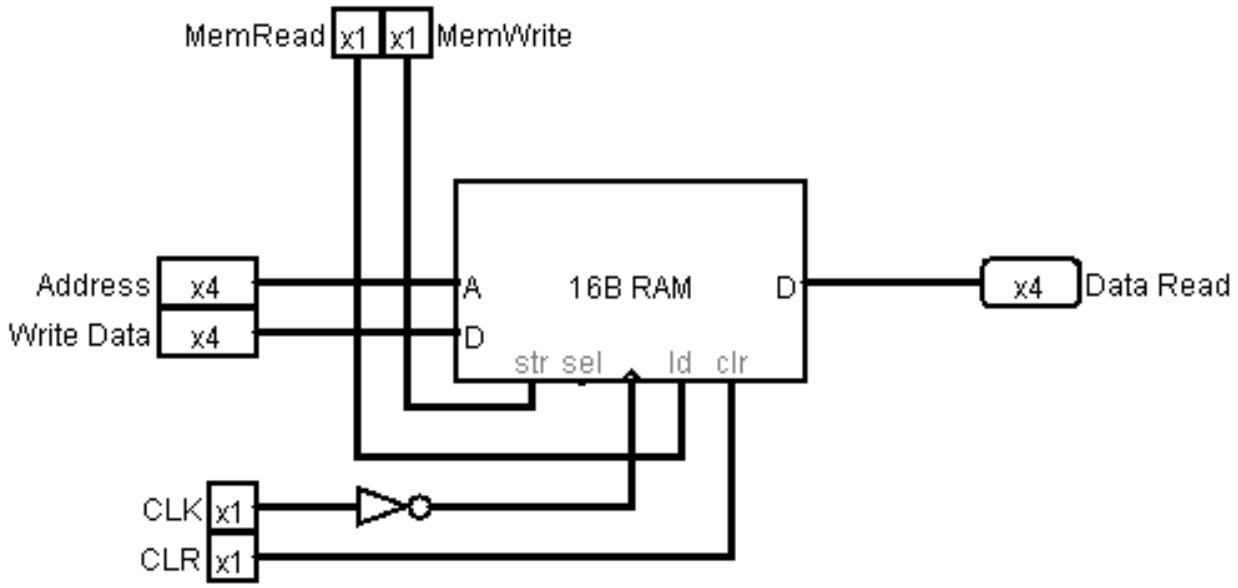
2.Control:

This part uses a ROM and controls the decision in main circuit. It also gives ALUop to determine which ALU operation is going to be done.



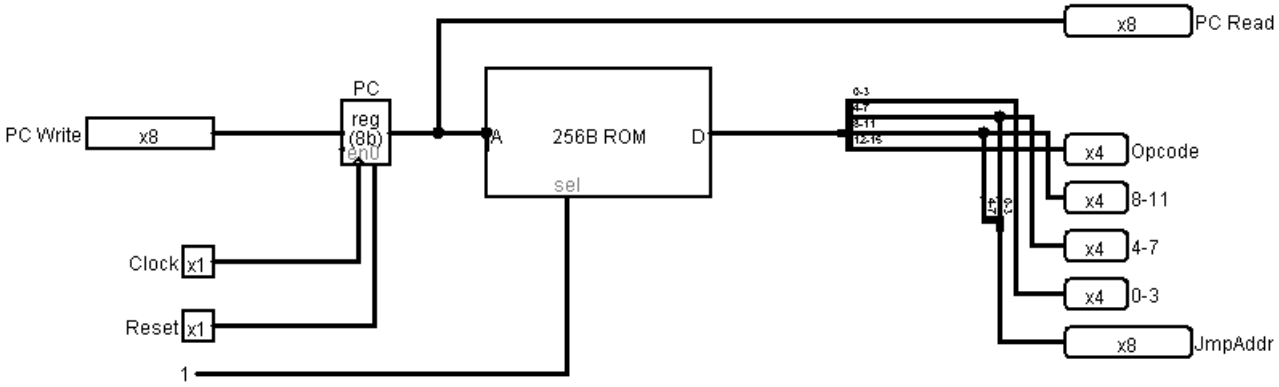
3.Data Memory:

It is the part where we store and load data. It plays a crucial role in store word and load word operation.



4.Instruction:

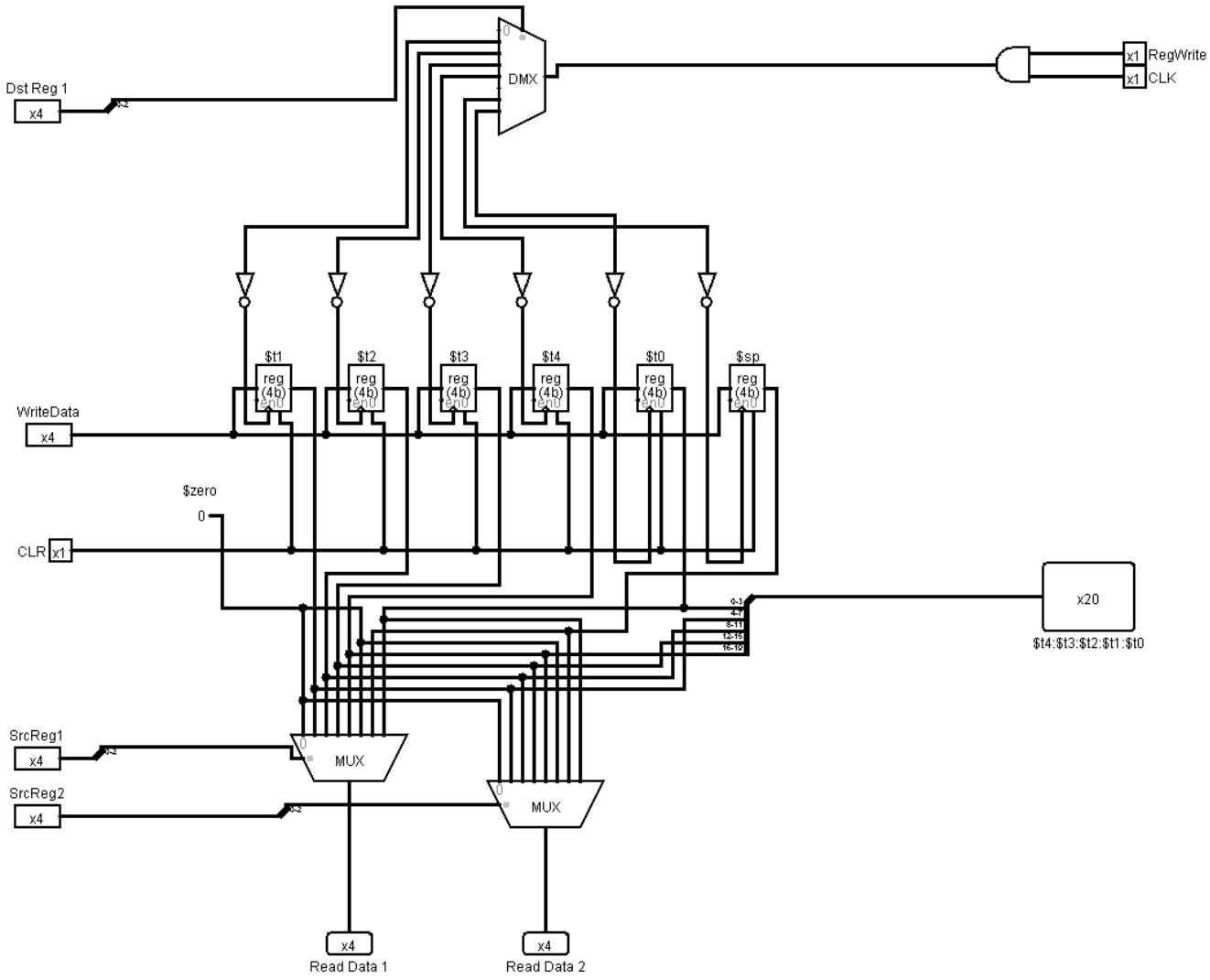
It stores the instruction in hex code which we got from our assembler program. The main circuit fetch instruction from this block.



5.Registers:

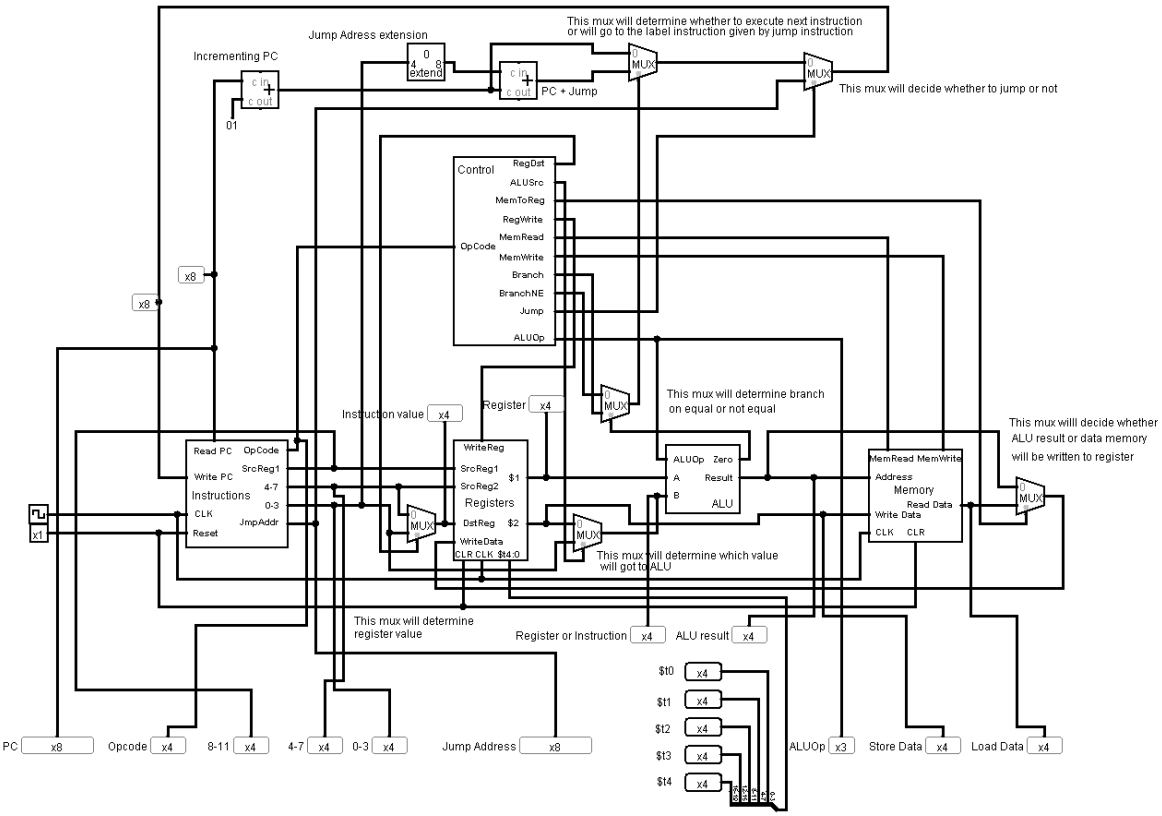
This is the register file. It uses flipflops to store the register value. There are six registers in our circuit. They are-

- $t0
- $t1
- $t2
- $t3
- $t3
- $t4
- $zero

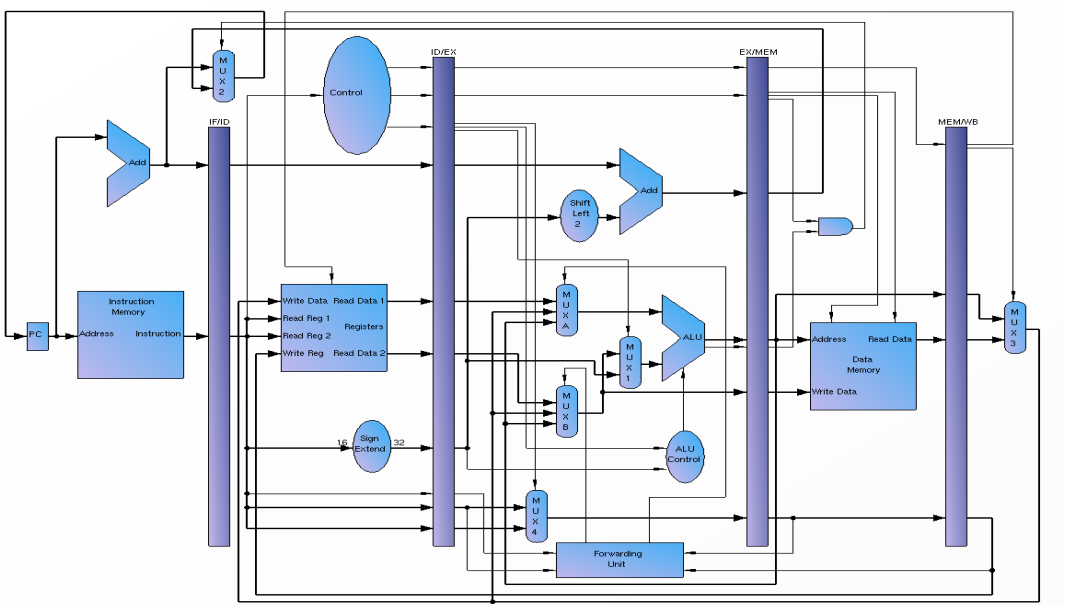


Main Circuit:

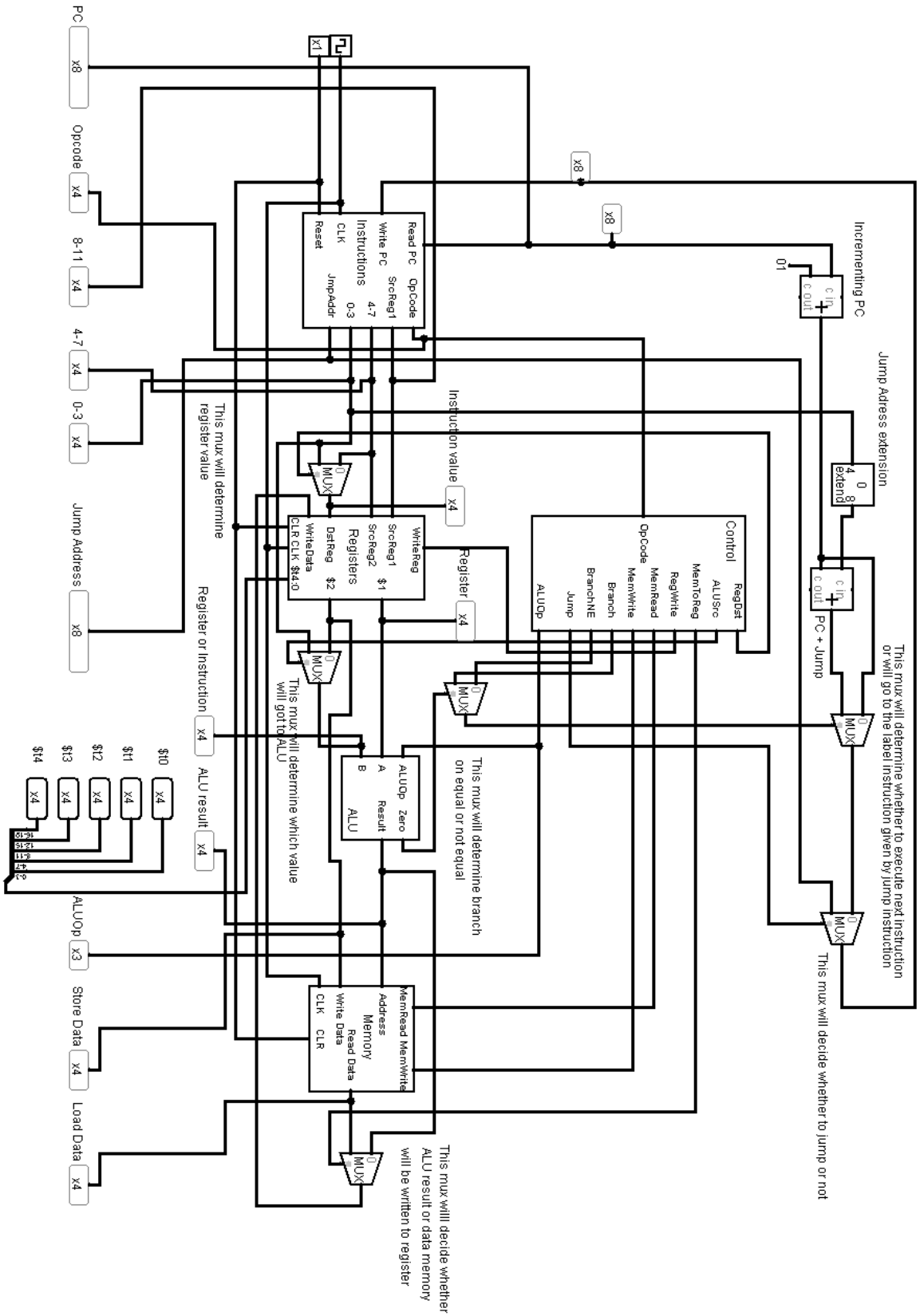
This is the main circuit of our design.



It is similar to the diagram mentioned below-



Main Circuit (Landscape View):



Program Execution Instruction:

For Softaware Simulation:

Machine Code Generation: Write an assembly code in a text file with .txt extension. Run the assembler program to generate an output file. There will be three output file -output.hex, output.bin, rom.hex.

Burn the ROM: Open the .circ file in Logisim. Go to the Instruction block and edit the contents of ROM by output.hex file. Go to the Control block and burn the ROM by rom.hex file. The control ROM can be burned only once but if we change instruction we have to burn instruction ROM every time.

Simulation: Now, go to the main block in .circ file and give clock to see the output. Every clock will execute one instruction.

For Hardware Simulation:

Hex File Generation: Run all the C program in Microchip Studio and generate hex file for every program. There are five C program, so the number of hex file is also five.

Burning ATMEGA: There are five ATMEGA32. We have already generated all the hex files. Now, burn the ATMEGA32 with appropriate hex file.

Running Hardware: Now, provide clock pulse to start execution. Every clock pulse will execute one instruction. There are four LEDs to see register output and five input to configure which register we want see. So, configure the input to see the particular register value.

Software Simulation for Hardware: There is a software called Proteus. We can do exactly same thing in proteus as we do in hardware part.

Special Feature:

We have considered stack memory. This will be used for push and pop instructions.

push $t0 instruction is replaced by the following instruction:

subi $sp,$sp,1

sw $t0,0($sp)

pop $t0 instruction is replaced by the following instruction:

addi $sp,$sp,1

lw $t0,0($sp)

IC Count:

Components	Details	Count
IC7404	Logical NOT Gate	1
IC7483	4-bit Adder	4
IC74157	Quad 2x1 MUX	9
IC74273	8-Bit D Flip Flop	1
ATMEGA32	Microcontroller	5

Software Used:

1.Logisim V2.7.1

2.Proteus

3.Microchip Studio

Discussion:

We use 1 clock cycle per instructions (CPI), the memory needs to be read at rising edge of the clock pulse and the registers need to be written to at the falling edge. Again, for shift if the shift amount exceeds 3 we have taken to modulo 4 because the shift amount cannot exceed the operand bit size in shifter IC.

Due to the shortage of IC74181, we use ATMEGA32. It also supports shift operation which are not supported in IC74181. Again due to shortage of EEPROM, the control unit, memory file and instruction file is implemented using ATMEGA32. The clock cycle of ATMEGA is fast enough that operations appear instantaneous lie the combinatorial circuit it replaces.

Two additional adders are used for PC increment and branching using IC7483 and IC74273 is used for PC. However, for branching operation the offset is sign extended. This allows branching 8 instructions forward and 7 instructions backward.

The original MIPS does not have S type format. But we implemented it. And other formats were also modified. In our modified version there is no bits reserved for shamt or func. As per problem specification, our address bit is 8 bits long. A total of 256 instructions can be loaded at once and the memory file has 256*4=1024 bits total capacity.

For simulation, we have to design two circuit. One is in Logisim using ROM, RAM, Flip Flop, ICs etc. Another is in proteus using ATMEGA, Flip Flop, ICs etc. This was very time consuming.

This assignment requires lots of hard work, knowledge and time. We Gave our best to complete it. Though we are not perfect, but we successfully completed it.