# CSE 306

# Assignment 3 : MIPS

Hafijul Hoque Chowdhury : 1905013
Tanhiat Fatema Afnan : 1905014
Syeda Rifah Tasfia : 1905019
Rakib Ahsan : 1905024
Wasif Hamid : 1905026

**Bangladesh University of Engineering and Technology**

**27 February 2023**

# 1 Introduction

A processor is the logic circuitry that responds to and processes the basic instructions that drive a computer. The processor is seen as the main and most crucial integrated circuitry (IC) chip in a computer, as it is responsible for interpreting most of computers commands. Although there are other processors in a computer(most notably GPU), it is used interchangeably with CPU (Central Processing Unit). CPUs will perform most basic arithmetic, logic and I/O operations, as well as allocate commands for other chips and components running in a computer.

Principal components of a CPU include a Instruction Memory Address which supplies the instructions and points to the next instruction, Control Unit that orchestrates the fetching (from memory) and execution of instructions by directing the coordinated operations of the ALU, registers and other components, ALU (Arithmetic and Logical Unit) that performs arithmetic and logic operations, processor registers that supply operands to the ALU and store the results of ALU operations, and finally memory unit that stores data and supplies it when needed.

There are many types of processor design architechture and implementations. MIPS (Microprocessor without Interlocked Pipelined Stages) is a family of reduced instruction set computer (RISC) instruction set architectures (ISA) developed by MIPS Computer Systems, now MIPS Technologies, based in the United States. A processor built using MIPS ISA is called a MIPS processor.

In this assignment we have implemented a simple 4 bit processor that implements the MIPS ISA. Each instruction will take 1 clock cycle to be executed. The processor is composed of five components.

1. **Program Counter** : The program counter (PC) is a 8-bit Register which activates at the falling edge of the clock signal. After every clock it adds 1 to its previous address. The value it stores is used as the Instruction Memory Address.

2. **Register File** : Register File is a bank of six Registers. They are denoted as $zero, $t0, $t1, $t2, $t3, $t4, $t5. $zero register stores 00H. Other Registers are used as General Purpose Registers.

3. **ALU** : The ALU does all the arithmetic and logical calculations. It is controlled by 3-bit ALUop code which sets the type of calculation it performs. It performs calculations with two 4-bit binary numbers.

4. **Control Unit** : Control Unit : The Control Unit decodes the instruction by giving the selection input to all the MUXs, Register File, Data Memory and ALU.

5. **Data Memory** : The Data Memory stores works as main memory. It has 256 bytes storage capacity. It stores Data as 8-bit value.

# 2 Instruction Set

## 2.1 MIPS Instruction Format

Our MIPS Instructions will be 16-bits long with the following four formats.

- 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 2 | Shamt |
|---|---|---|---|
| 4-bits | 4-bits | 4-bits | 4-bits |

- 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 |

## 2.2 Instruction Set with Instruction ID

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

## 2.3 Instruction Set with Op-Code

| Instruction ID | Instruction Type | Format | Instruction |
|---|---|---|---|
| 0000 | Control-Conditional | I | bneq |
| 0001 | Logic | R | sll |
| 0010 | Logic | R | nor |
| 0011 | Arithmetic | R | add |
| 0100 | Memory | I | lw |
| 0101 | Control-Conditional | J | j |
| 0110 | Logic | R | and |
| 0111 | Logic | I | andi |
| 1000 | Logic | R | or |
| 1001 | Logic | R | srl |
| 1010 | Arithmetic | R | sub |
| 1011 | Memory | I | sw |
| 1100 | Logic | I | ori |
| 1101 | Control-Conditional | I | beq |
| 1110 | Arithmetic | I | subi |
| 1111 | Arithmetic | I | addi |

# 3 High Level Block Diagram

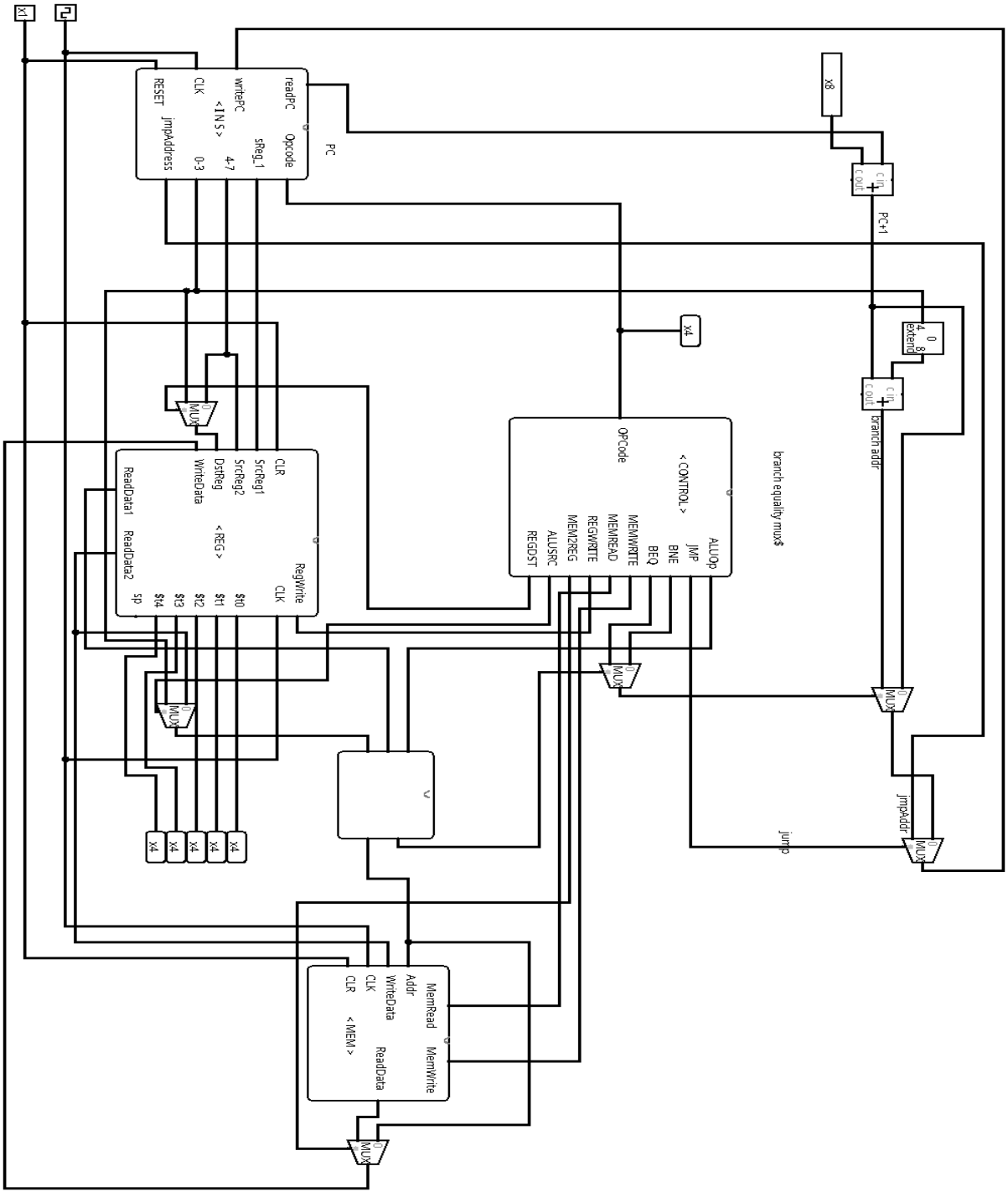


Figure 1: High Level Block Diagram of 4-Bit MIPS

# 4 Block Diagram of Components

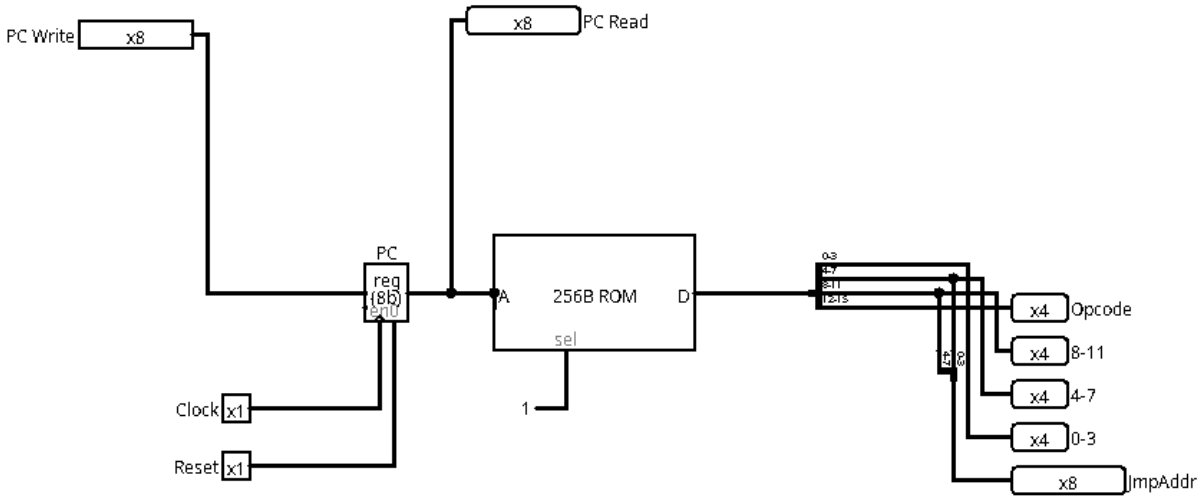


Figure 2: Instruction Memory

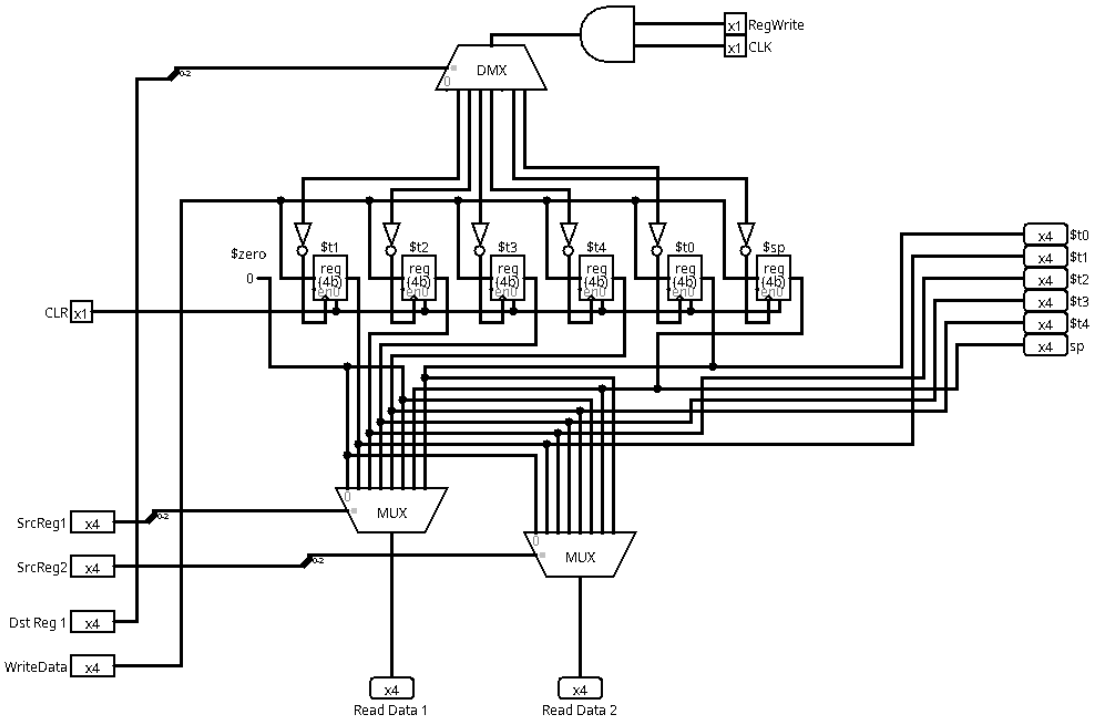


Figure 3: Register File

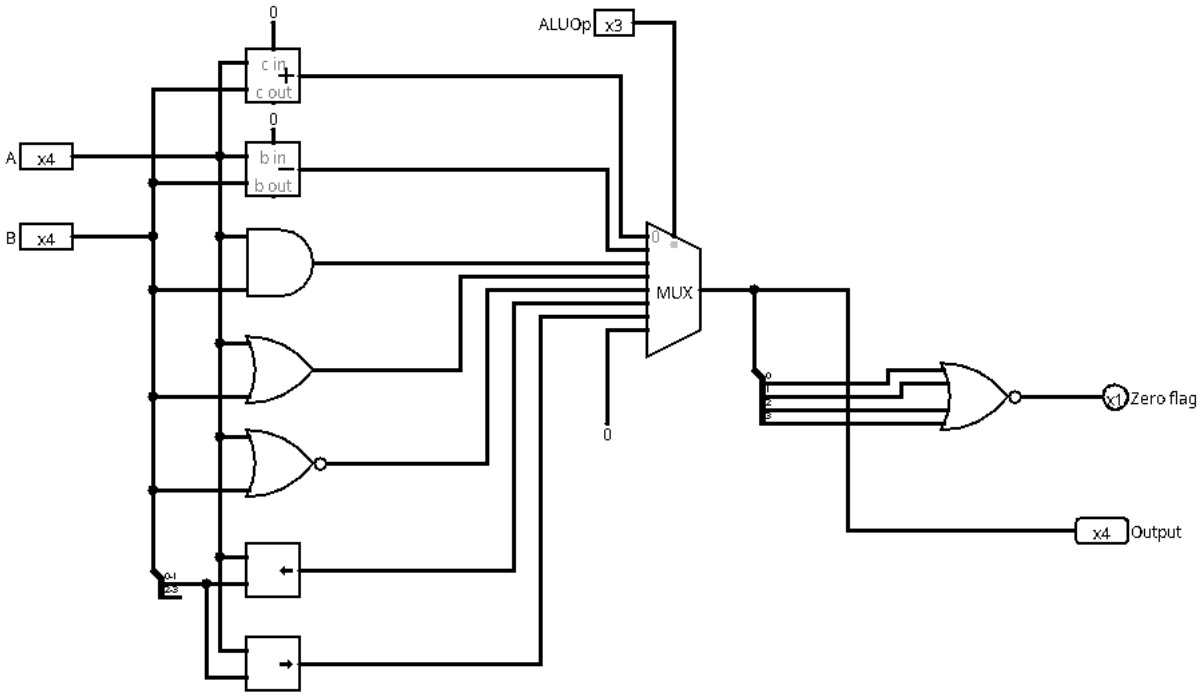


Figure 4: Arithmetic and Logical Unit

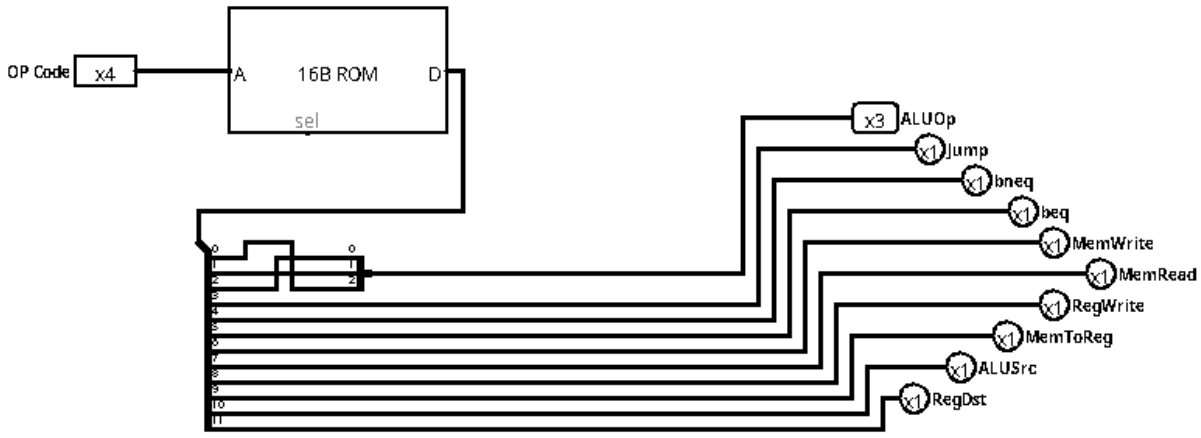


Figure 5: Control Unit

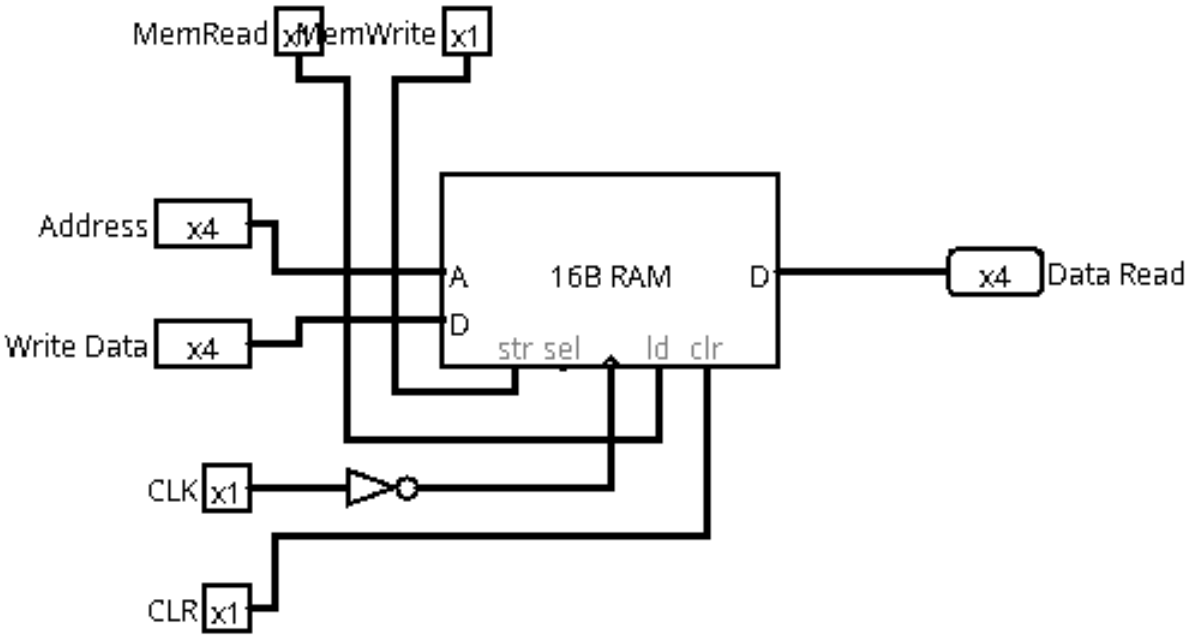


Figure 6: Data Memory

# 5 Number of ICs Used

| Gate<br>MUX (2-to-1) | Gate Count<br>29 | IC<br>74157 | IC count<br>8 |
|---|---|---|---|
| Adder (4-bit) | 4 | 7483 | 4 |
| D flip-flop | 4 | 74273 | 1 |
| Microprocessor | 5 | ATmega32 | 1 |

# 6 Simulation

Simulation is done using Logisim version 2.7.1

# 7 Discussion

In the experiment, we have recreated both the software and hardware implementation of a MIPS processor. The software implementation was completed without any large setbacks but in the hardware section of the implementation we have faced many obstacles time and time again. We have used Atmega32s in place of control unit, ALU, Register and Data memory. We had to spend a lot of time to

determine which ports of the Atmega 32 to use for which case. We faced some problems using the PORT C of the Atmega 32s as well. For the PC counter originally we had tried using binary up counter with mixed results. Afterwards, it was replaced by D flipflops which was working correctly at first but unexpectedly broke down after some time. Eventually we had to drop jump and branch instructions from our implementation. We also faced some problems regarding power distribution in the circuit. Additionally, our breadboards had some problems where the Atmega's were not being set up properly while some ports was shorted which fried some of our 2x1 MUXs. The connections were given very carefully, particularly the jumper wires were checked before using them in the circuit. Ultimately it has been a learning experience and have been a very satisfying experiment.