



CSE 306

Computer Architecture Sessional

Experiment 03

Assignment on 4-bit MIPS Design, Simulation, and Implementation

Group 03

1805067 - Muhammad Ehsanul Kader

1805068 - Showvik Biswas

1805077 - Kazi Ababil Azam

1805082 - Anup Bhowmik

1805087 - Fardin Anam Aungon

Date of Submission

19 - 08 - 2022

Introduction:

A processor (CPU) is the logic circuitry that responds to and processes the basic instructions that drive a computer. There are several types of processors available. The processor that uses MIPS (Microprocessor without Interlocked Pipelined Stages) is a family of reduced instruction set computer (RISC) instruction set architectures (ISA) is known as a MIPS processor.

A processor is capable of performing a handful of operations such as

1. Arithmetic operations
2. Logic operations
3. I/O operations

Arithmetic logic unit (ALU) is the fundamental component of a processor that is required to perform almost any instruction i.e., program count, any arithmetic and logical operations, address count, finding the jumping address, calculating memory offset, and many more. The other necessary components of a processor are: address and data buses, register files, instruction memory, data memory, etc.

In our given assignment, we have implemented a 4-bit processor(having 4-bit ALU) that takes a modified and reduced version of the MIPS instruction set. Each instruction takes one clock cycle and is executed on the falling edge of the clock. We have designed the following necessary components in order to implement the processor:

1. 4-bit ALU

Responsible for all the arithmetic and logical operations along with all the addition and subtraction operations required for finding load or store addresses, unconditional jump or branching instructions.

2. Program Counter

Program Counter (PC) is an 8-bit register that keeps track of the current instruction. After each instruction, PC is incremented by 1 (the ROM we have used is word (16-bit) addressable, adding 1 takes us to the next instruction written in the ROM). The stored value indicates the Instruction memory address

3. Instruction memory

Stores the actual hex code of the instruction that is converted to binary to run the processor

4. Data memory

Stores 4-bit data and works as the main memory.

5. Register file

We have used 6 registers namely \$zero, \$t0, \$t1, \$t2, \$t3, \$t4. \$zero register stores value 0H. The others are general purpose registers.

6. Control unit

Decodes the instruction by giving the selection input to all the MUXs, Register File, Data Memory, and ALU.

Instruction Set:

Our group has the following sequence

CHOBNGMJEF LAIDPK

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

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	Shamt
4-bits	4-bits	4-bits	4-bits

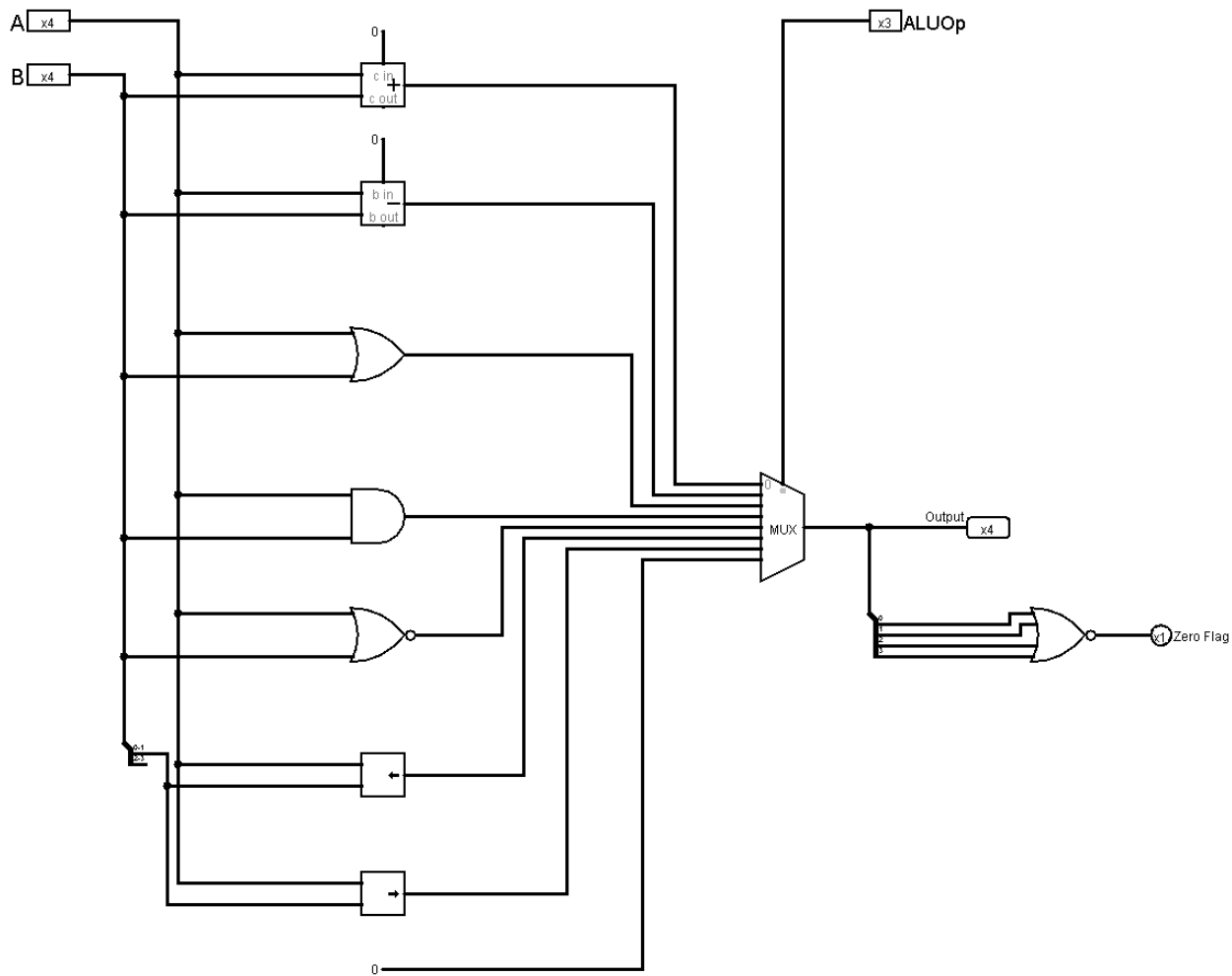
I-type

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

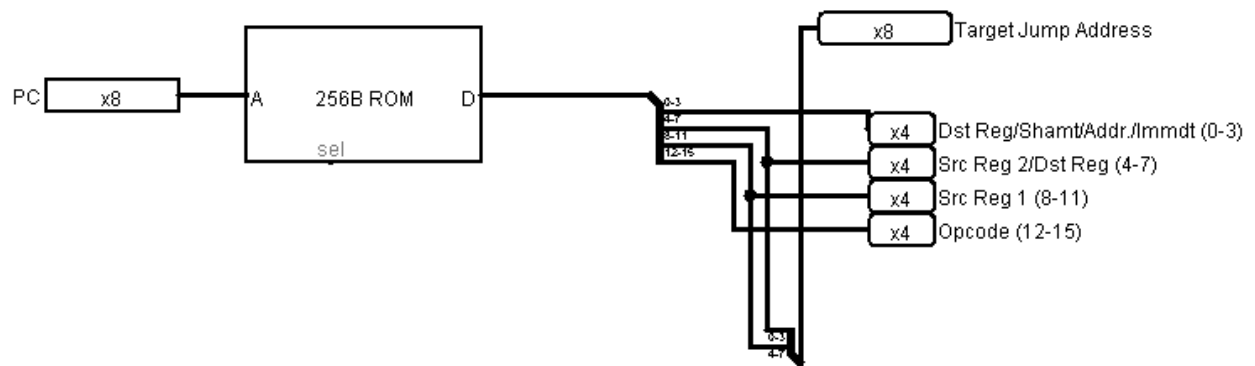
J-type

Opcode	Target Jump Address	0
4-bits	8-bits	4-bits

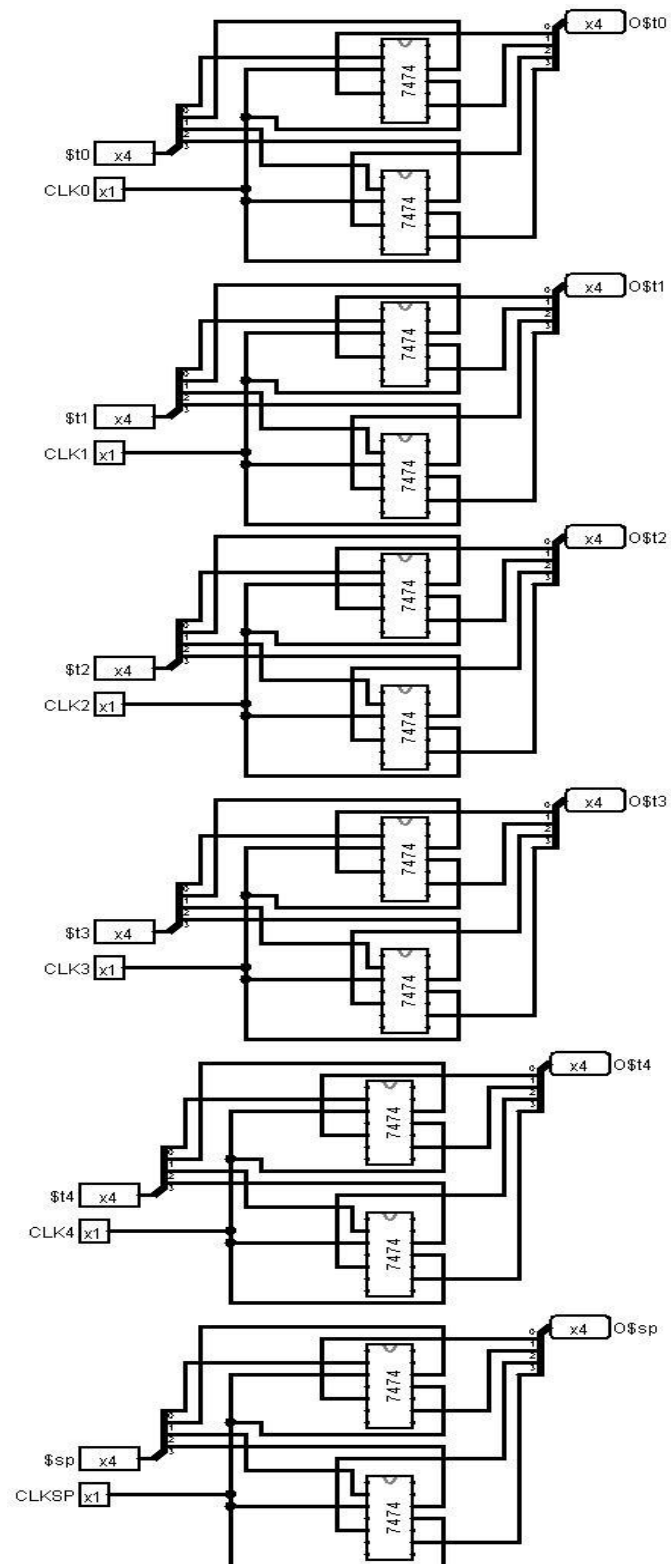
Circuit Diagram



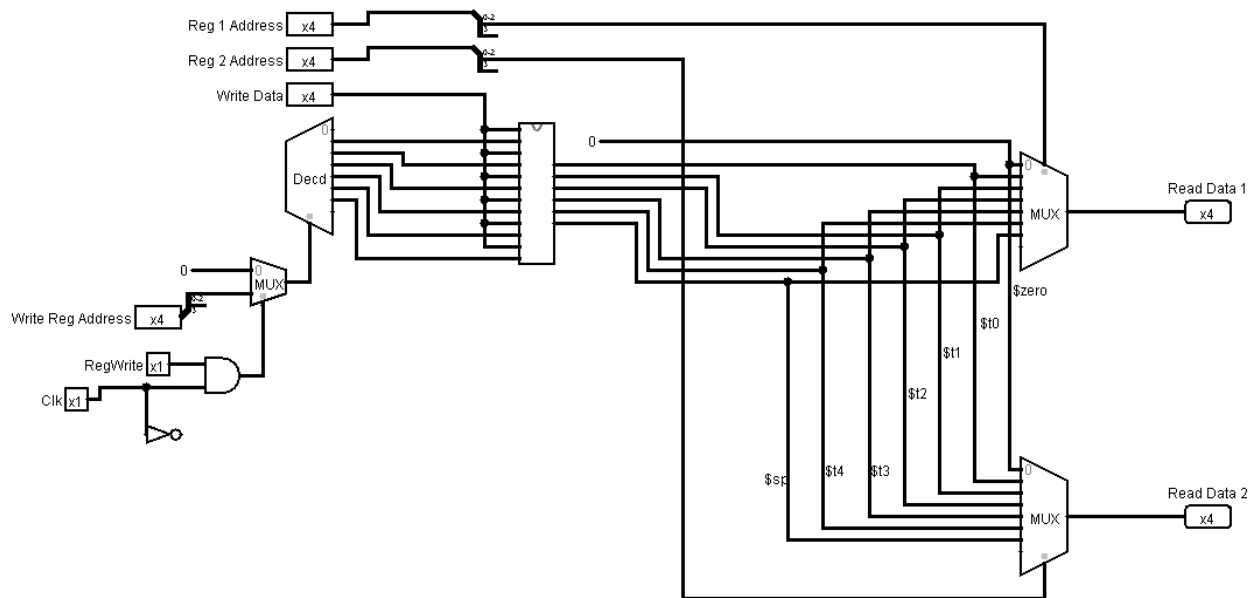
4 bit ALU



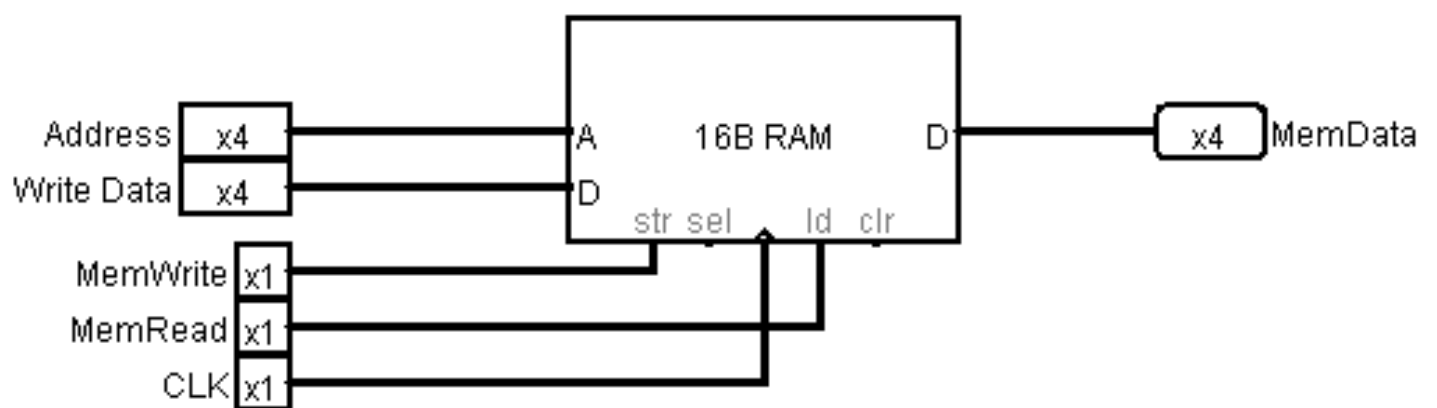
Instruction Memory



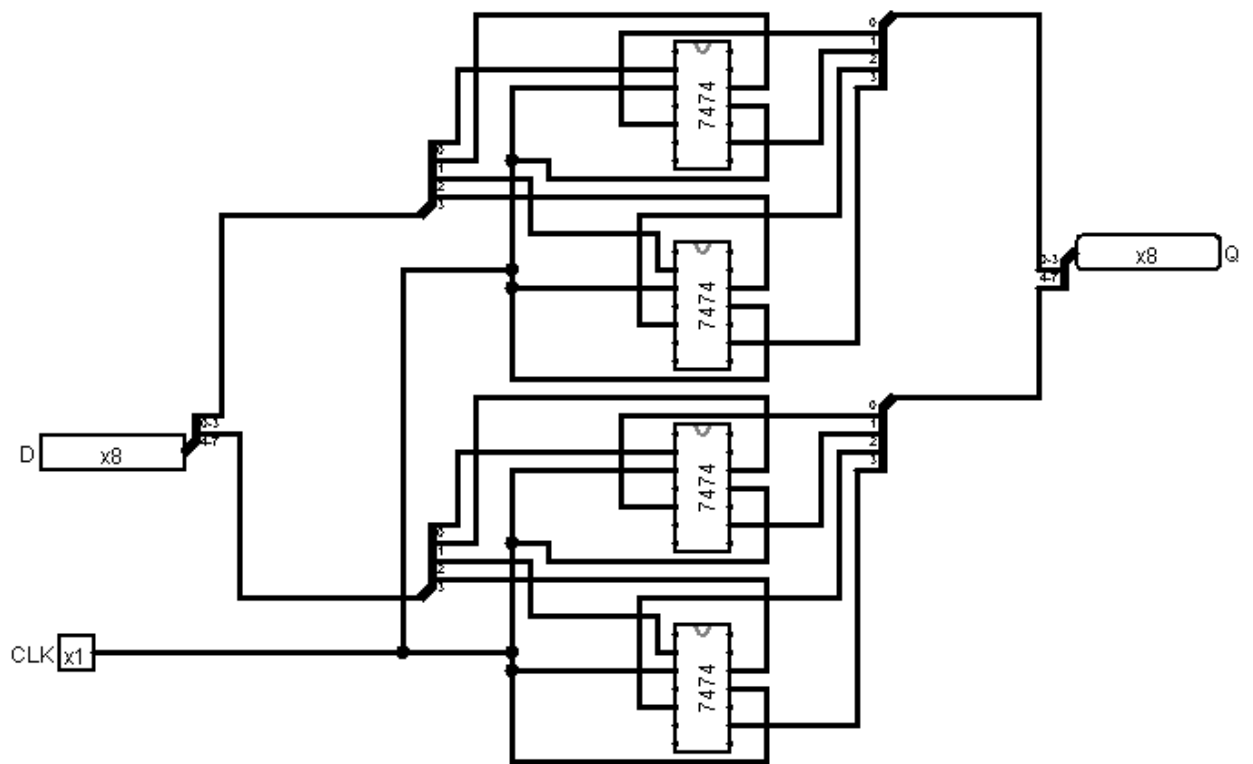
Registers



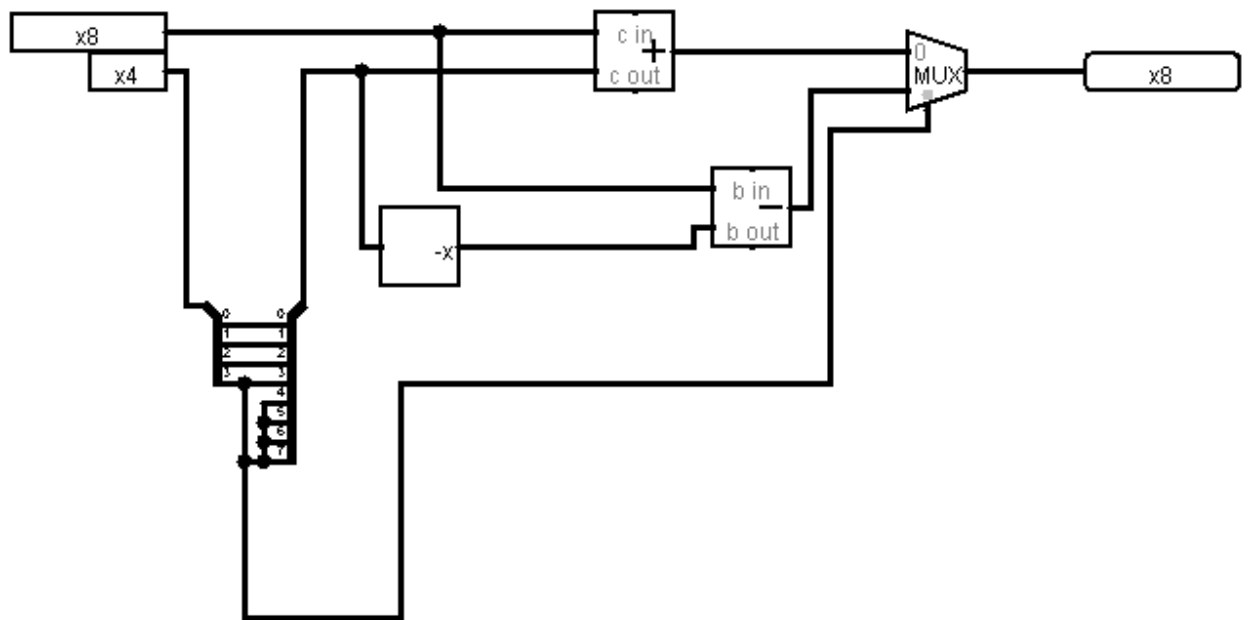
Register file



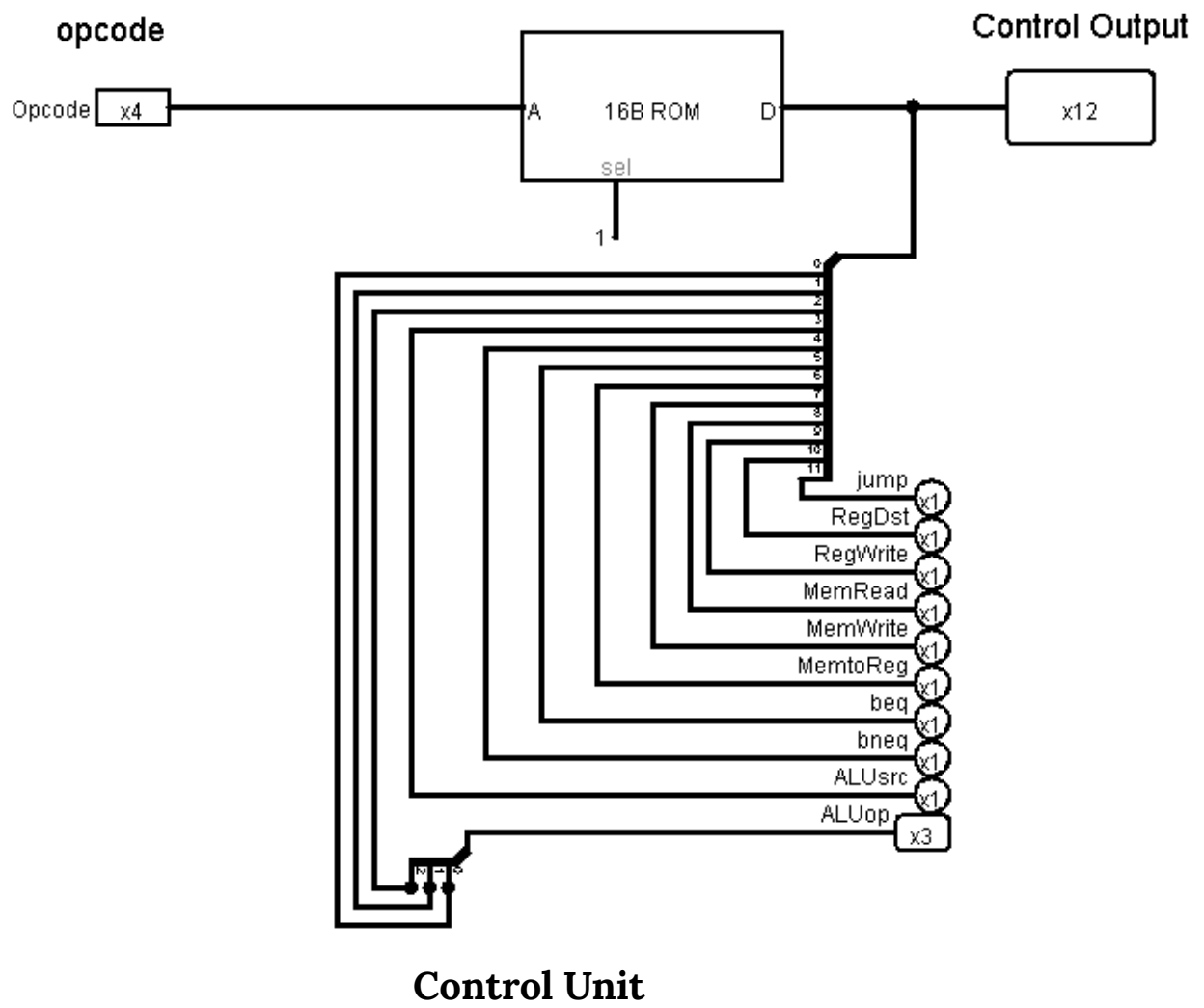
Data Memory



8 bit D Flip-Flop



8 bit adder/subtractor





4 bit MIPS Computer

How to write and execute a program in this machine:

We are using a ROM to write program that is fed to the machine. A C program takes the instructions in plain text and generates our machine-readable hex code.

ICs used with their count

IC/ Gate	IC Count
7474	17
7408	1
ROM	2
RAM	1
Multiplexer	8

Simulation Software

Logisim 2.7.1

Discussion

We have invested quite some time to make the design as efficient as possible. We tried to minimize the number of ICs. We have tested the given instructions as well as the corner cases. The experiment helped us better understand the overall MIPS instruction set and the datapath as a whole.