CSE-306: Offline-3 Report

Section: B1

Group: 01

Group Members:

Noshin Nawal – 1805061

Ayesha Binte Mostofa – 1805062

Razin Reaz Abedin – 1805074

Maneesha Rani Saha – 1805076

Sumaiya Sultana - 1806079

Nusrat Nur Labiba – 1805081

Submission Date: 17 August 2022

Introduction:

A processor is an integrated electronic circuit that performs arithmetical, logical, input/output (I/O) and other instructions on some external data source that run a computer. Main components of a processor are Arithmetic Logic Unit (ALU), Control Unit and Datapath.

In this assignment, we have designed a 4-bit processor that implements the MIPS (Microprocessor without Interlocked Pipelined Stages) instruction set. We have implemented the MIPS processor using six main components:

- **Program Counter:** The program counter (PC) of the processor is an 8-bit register which keeps track of the instruction count of our program and the address of current instruction. The PC increments with at the falling edge of each clock pulse.
- **Register File:** There are six temporary registers in the register file, namely: $zero, $t0, $t1, $t2, $t3, $t4. $zero register is a special register only used to store the value 0. The other five registers are used for general purpose.
- **Control Unit:** The control unit takes opcode as input from the instruction and generates 13 control bits which provides selection bits to MUXs, register file, ALU and data memory.
- **ALU:** The algorithmic logic unit performs different arithmetic and logical operations depending on the ALU-opcode generated by the control unit.
- **Instruction Memory:** The instruction memory stores all instructions of our program in hex value. Depending on the 8-bit address of PC, it executes all instructions one by one. Instructions of our processor are 16-bit long and are of 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

- **Data Memory:** Data memory is the main memory of our processor which stores values as 4-bit data. Total capacity of the data memory is 16x4 bits.

Instruction Set:

Table-1: Instruction Set with given Instruction ID

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

Table-2: Instruction Set with Opcode

Instruction ID	Instruction Category	Instruction Type	Instruction	Opcode
K	Logic	R	nor	0000
F	Logic	I	andi	0001
L	Memory	I	lw	0010
E	Logic	R	and	0011
N	Control	I	beq	0100
H	Logic	I	ori	0101
J	Logic	R	srl	0110
I	Logic	R	sll	0111
G	Logic	R	or	1000
D	Arithmetic	I	subi	1001
O	Control	I	bneq	1010
M	Memory	I	sw	1011
B	Arithmetic	I	addi	1100
A	Arithmetic	R	add	1101
C	Arithmetic	R	sub	1110
P	Control	J	j	1111

Circuit Diagram:

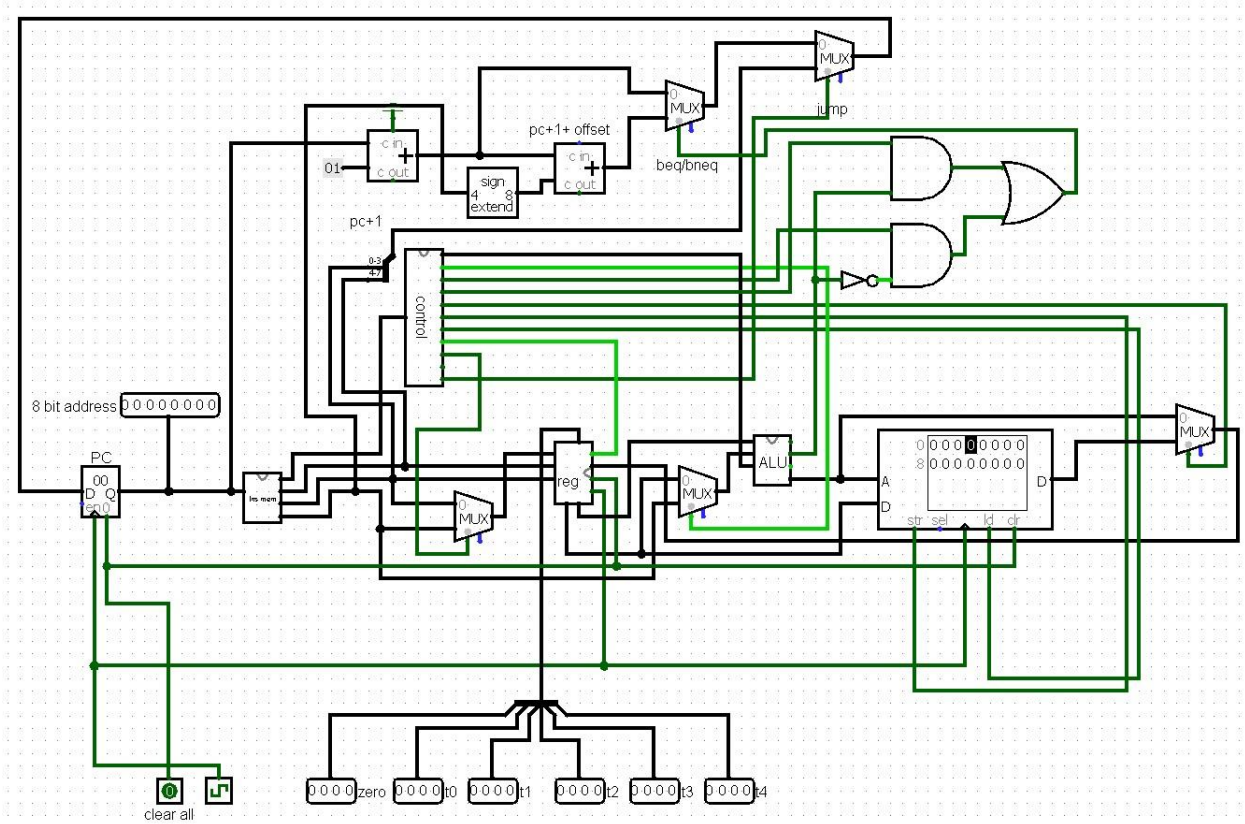

Figure-1: Complete Block Diagram of 4-bit MIPS Processor

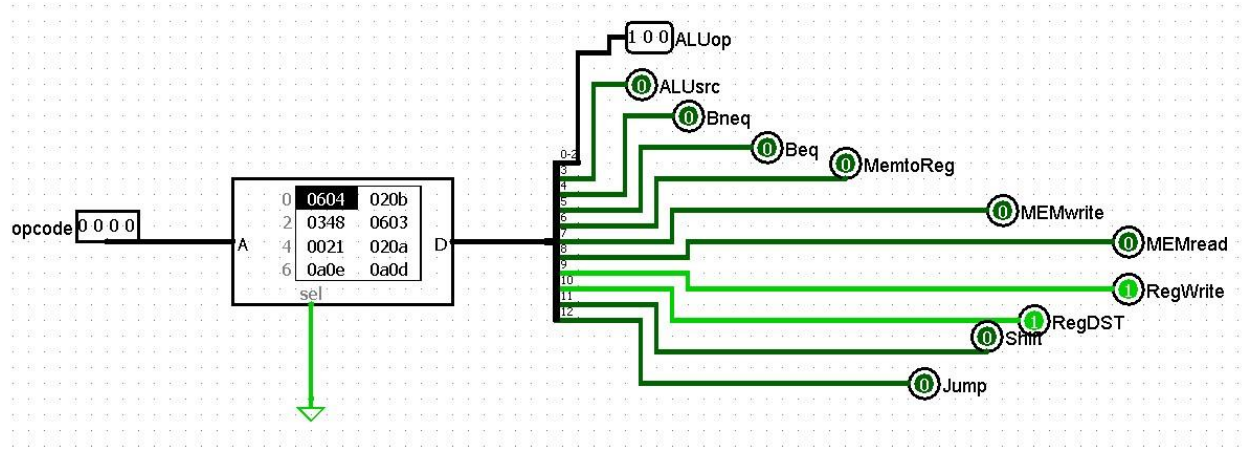

Figure-2: Control Unit

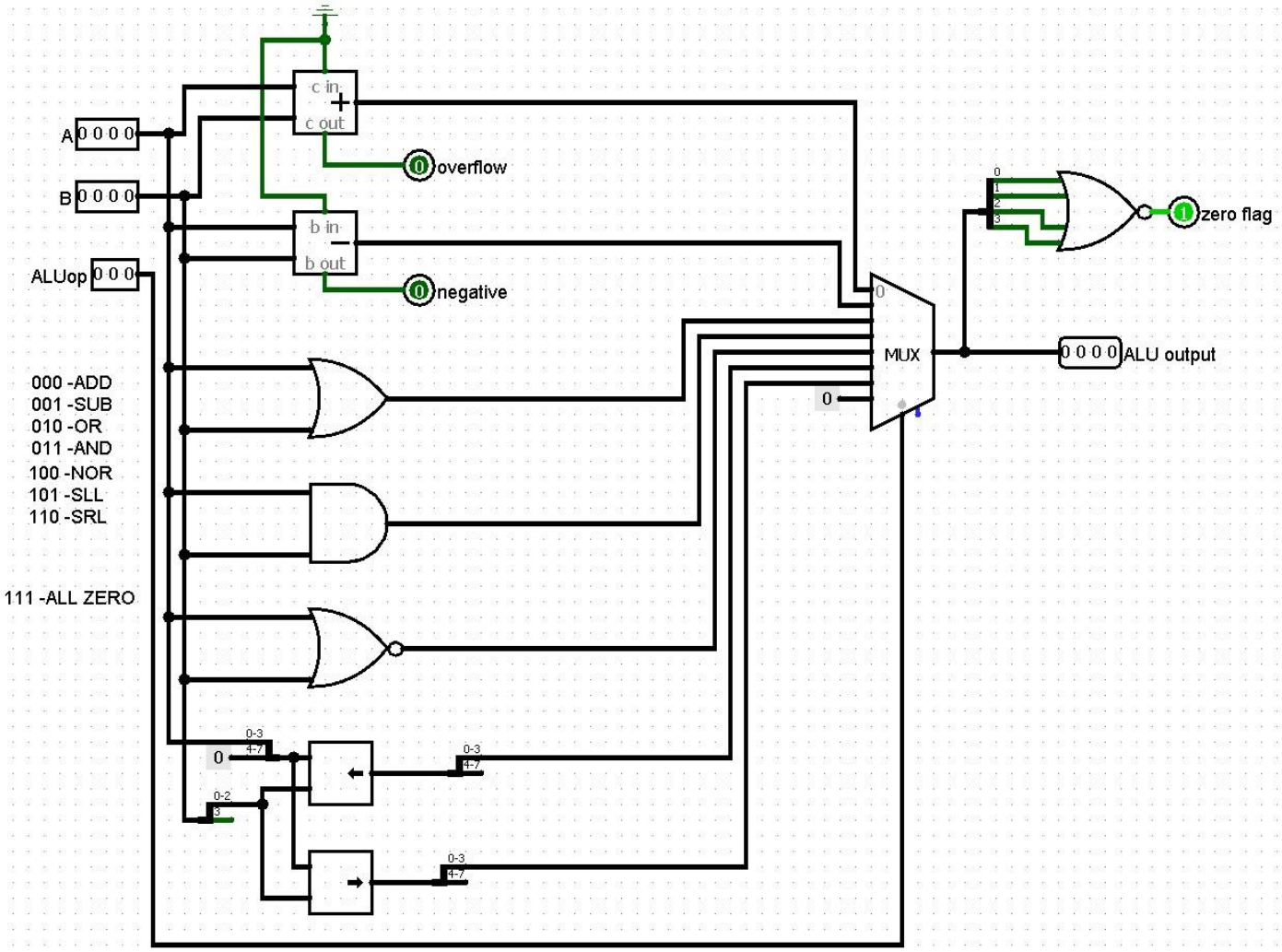
A 0 0 0 0
B 0 0 0 0
ALUop 0 0 0
c in
c out
overflow
b in
b out
negative
000 -ADD
001 -SUB
010 -OR
011 -AND
100 -NOR
101 -SLL
110 -SRL
111 -ALL ZERO
MUX
zero flag
0 0 0 0 ALU output

Figure-3: ALU

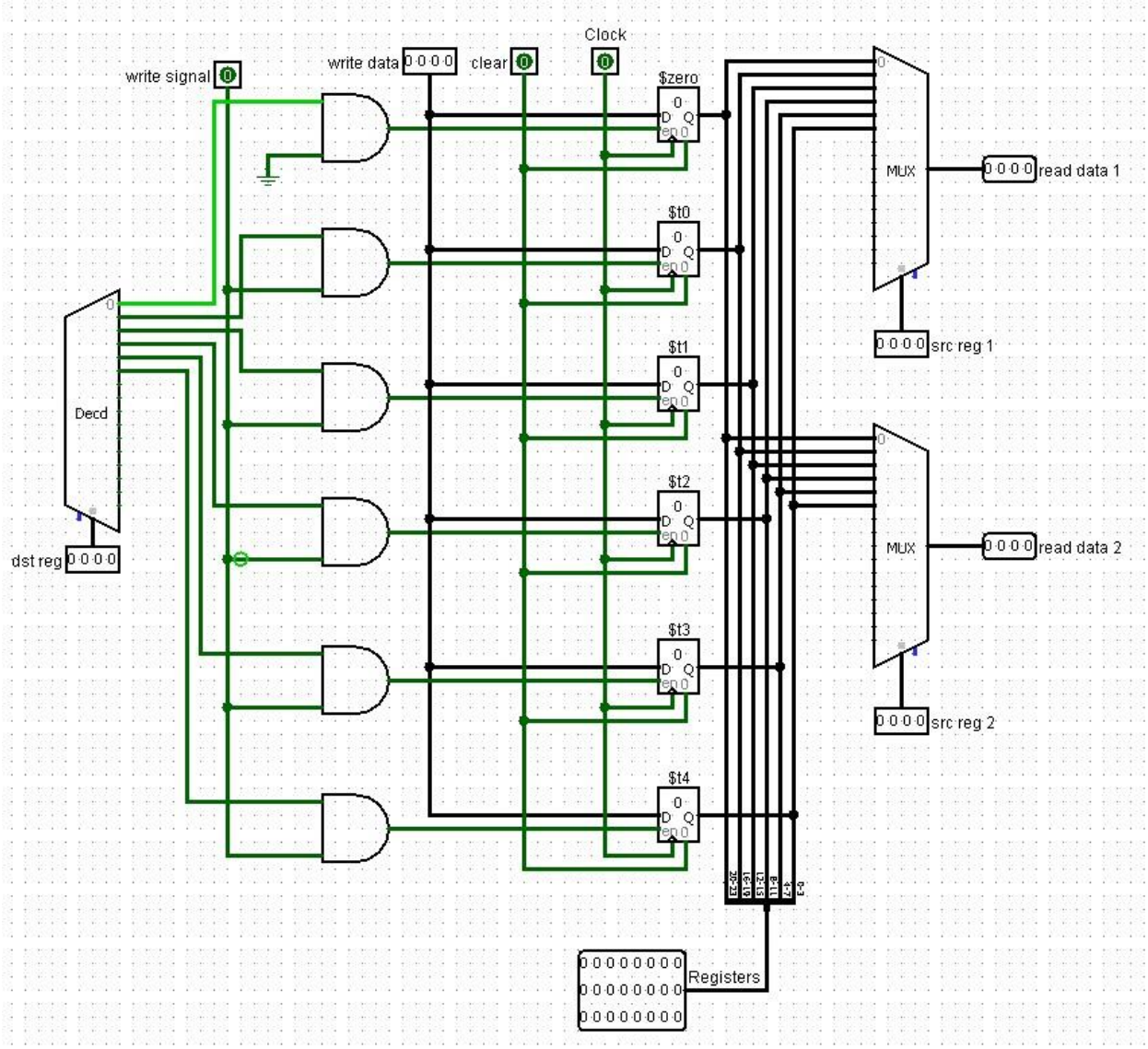

Figure-4: Register File

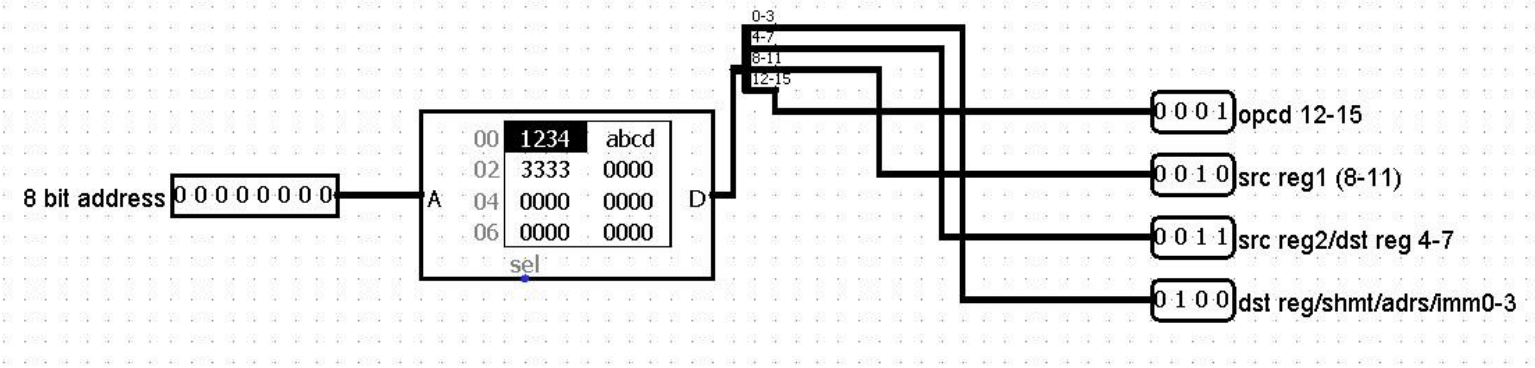

Figure-5: Instruction Memory

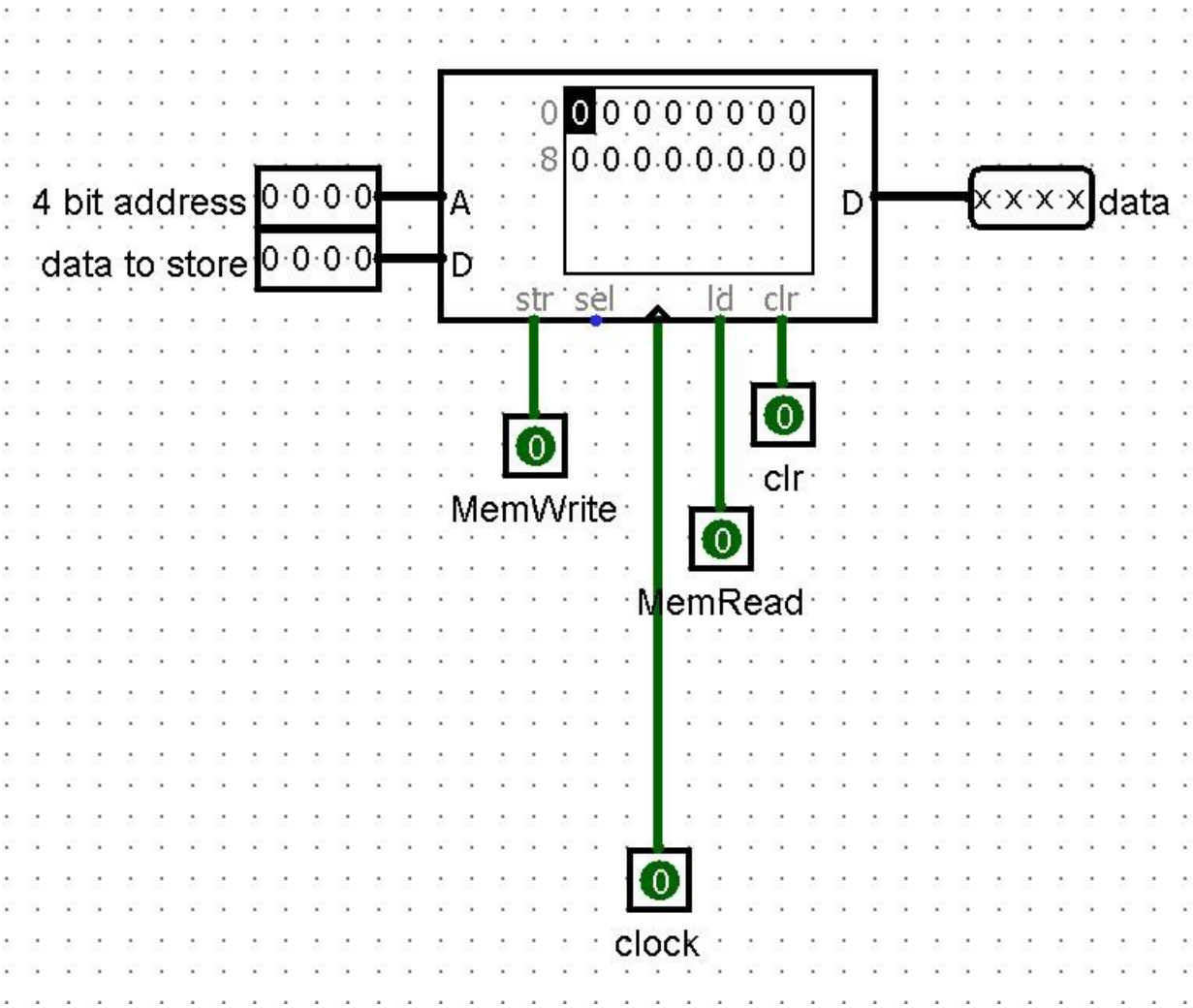

Figure-6: Data Memory

Writing and Executing Program:

- The code to convert assembly language to machine language is written in c++.
- The code reads the assembly code from an input file (“in.txt”) and generates the machine code in hex (“out”).
- The hex code is then loaded into the instruction memory (ROM) of our machine.
- Then each instruction is run per clock pulse.

IC used with Count:

Gate	Gate Count	IC	IC Count
AND (2-bit)	9	7408	3
OR (2-bit)	2	7432	1
NOT (2-bit)	1	7404	1
NOR (4-bit)	1	404002	1
MUX (2-to-1)	5	74157	2
MUX (8-to-1)	1	74151	1
MUX (16-to-1)	2	74150	2
ADDER (4-bit)	1	74283	1
ADDER (8-bit)	2	7483	1
Right Shifter (8-bit)	1	Right Shifter (8-bit)	1
Left Shifter (8-bit)	1	Left Shifter (8-bit)	1
Subtractor (4-bit)	1	Subtractor (4-bit)	1
decoder (4-to-16)	1	74154	1
Register (8-bit)	7	Register (8-bit)	7
RAM (4-bit)	1	RAM (4-bit)	1
ROM (13-bit)	1	ROM (13-bit)	1
ROM (16-bit)	1	ROM (16-bit)	1
Clock	1	Clock	1

Simulator:

Logisim Version 2.7.1

Discussion:

Our software simulation varies from our hardware implementation. For the simulation, we have used 1-RAM, 2-ROMs, 2-Shifters and 3-ALUs but in hardware implementation, we have used ATMega32 since we can program ATMega32 to add, subtract and store data. This will greatly reduce the number of components we have to use for hardware implementation. Therefore, our design is optimized.