North South University

Department of Electrical & Computer Engineering

LAB REPORT

Course Name: **CSE332L- Computer Organization and Architecture Lab**

Experiment Number: 07

Experiment Name: **Build a single cycle Datapath**

Experiment Date: 17/08/2022

Report Submission Date: 23/08/2022

Section: 02

Group Number: 01

Student Name: **Md. Baker**	Score
Student ID: **1911672642**	
Remarks:	

Exp Name: Build a single cycle Datapath

Objectives:

- We have combined all the (R-Type, I-Type, J- Type) Datapath's into one SINGLE CYCLE DATAPATH.
- We have run Specific instructions in the Datapath (lw, sw, add, sub)
- We have run Specific instructions in the Datapath(jump)

Theory:

Single Datapath's is equivalent to the original single-cycle Datapath The data memory has only one Address input. The actual memory operation can be determined from the MemRead and MemWrite control signals. There are separate memories for instructions and data. There are 2 adders for PC-based computations and one ALU. The control signals are the same.

The simplest Datapath is the single cycle Datapath. The basic components are a register file to store the data and functional units to operate on the data such as an adder/subtractor, logical unit, and a barrel shifter. We have constructed all of these components from basic gates and switches and should be familiar with their operation. The issue now is how can we compose larger systems with these components. How large should the data be? How many bits? In our example we will pick 32 bits, a number that is compatible with Datapath's found in the majority of modern microprocessors, controllers and signal processing chips.

The branch Datapath (jump is an unconditional branch) uses instructions. Each instruction causes slightly different functionality to occur along the Datapath. We run specific instructions in the Datapath using the control unit table below. The Control Unit is the part of the computer's central processing unit (CPU), which directs the operation of the processor. It was included as part of the Von Neumann Architecture by John von Neumann. It is the responsibility of the Control Unit to tell the computer's memory, arithmetic/logic unit and input and output devices how to respond to the instructions that have been sent to the processor. It fetches internal instructions of the programs from the main memory to the processor instruction register, and based on this register contents, the control unit generates a control signal that supervises the execution of these instruction.

Circuit Diagrams:

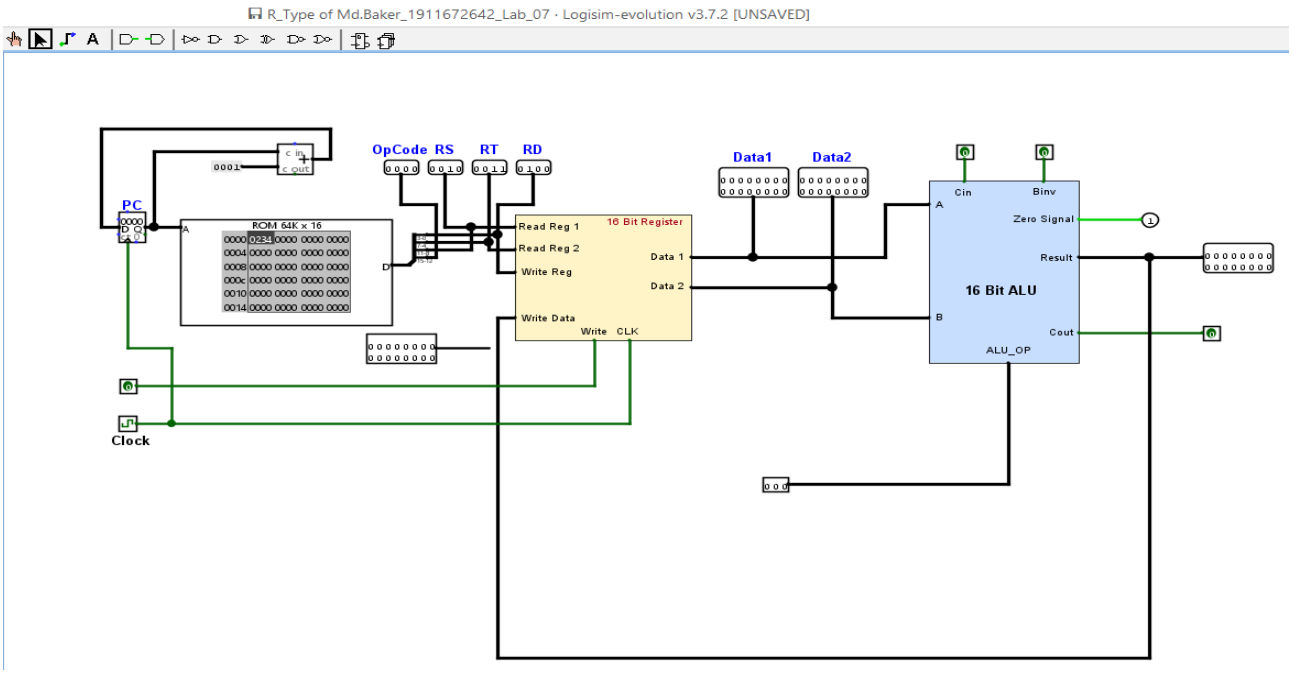


Figure: R-Type Datapath

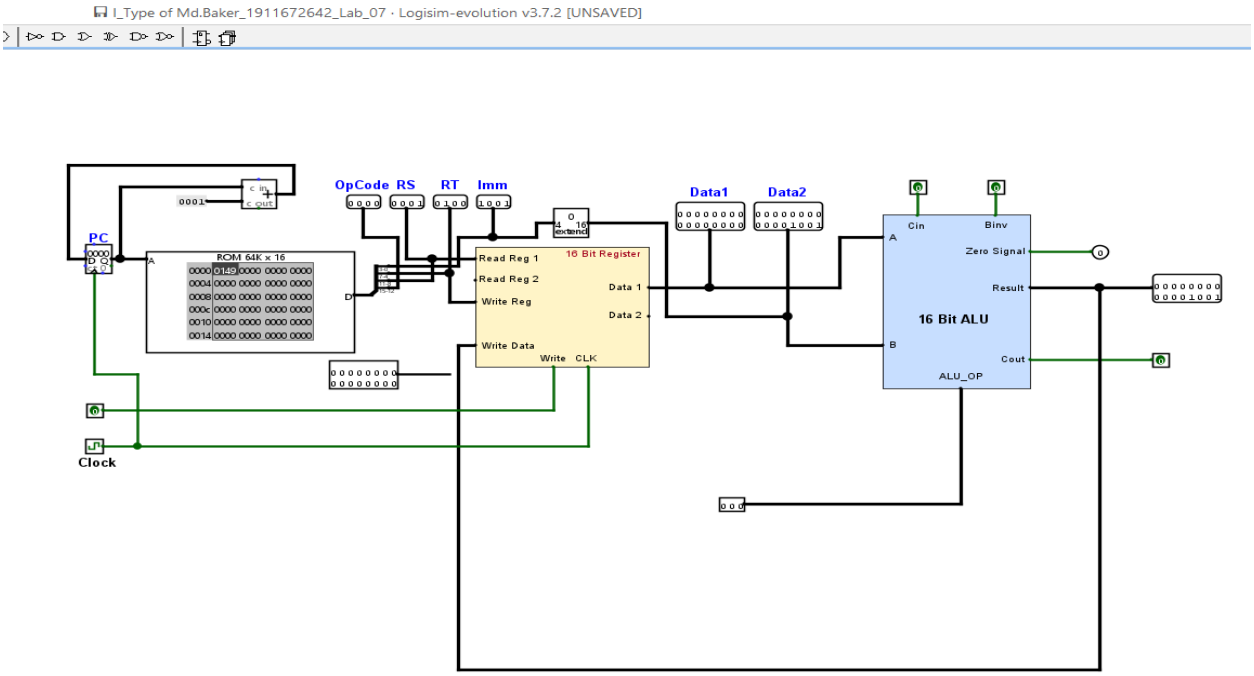


Figure: I - Type Datapath

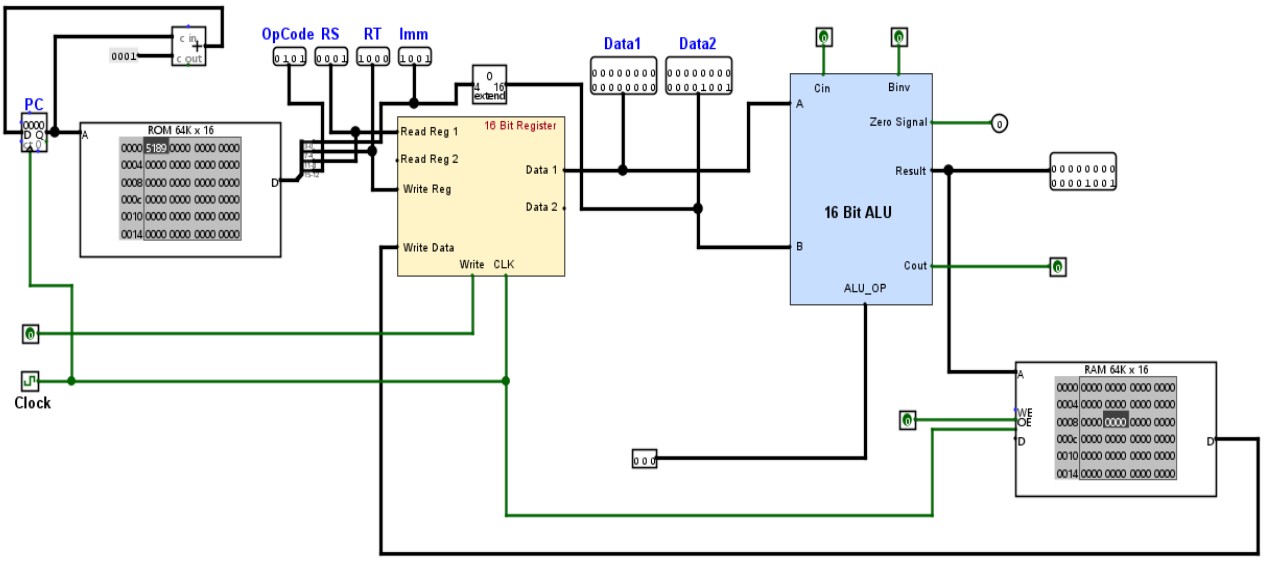
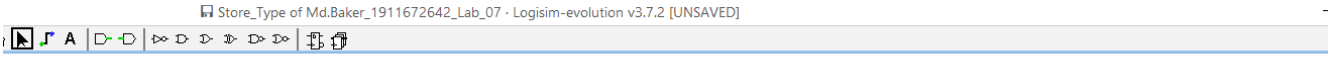


Figure: LW- Datapath



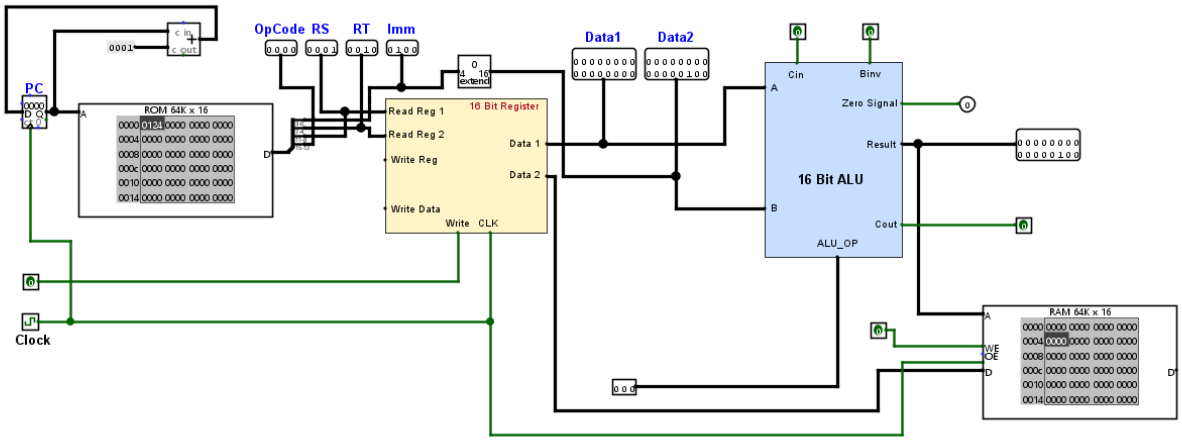


Figure: SW - Datapath

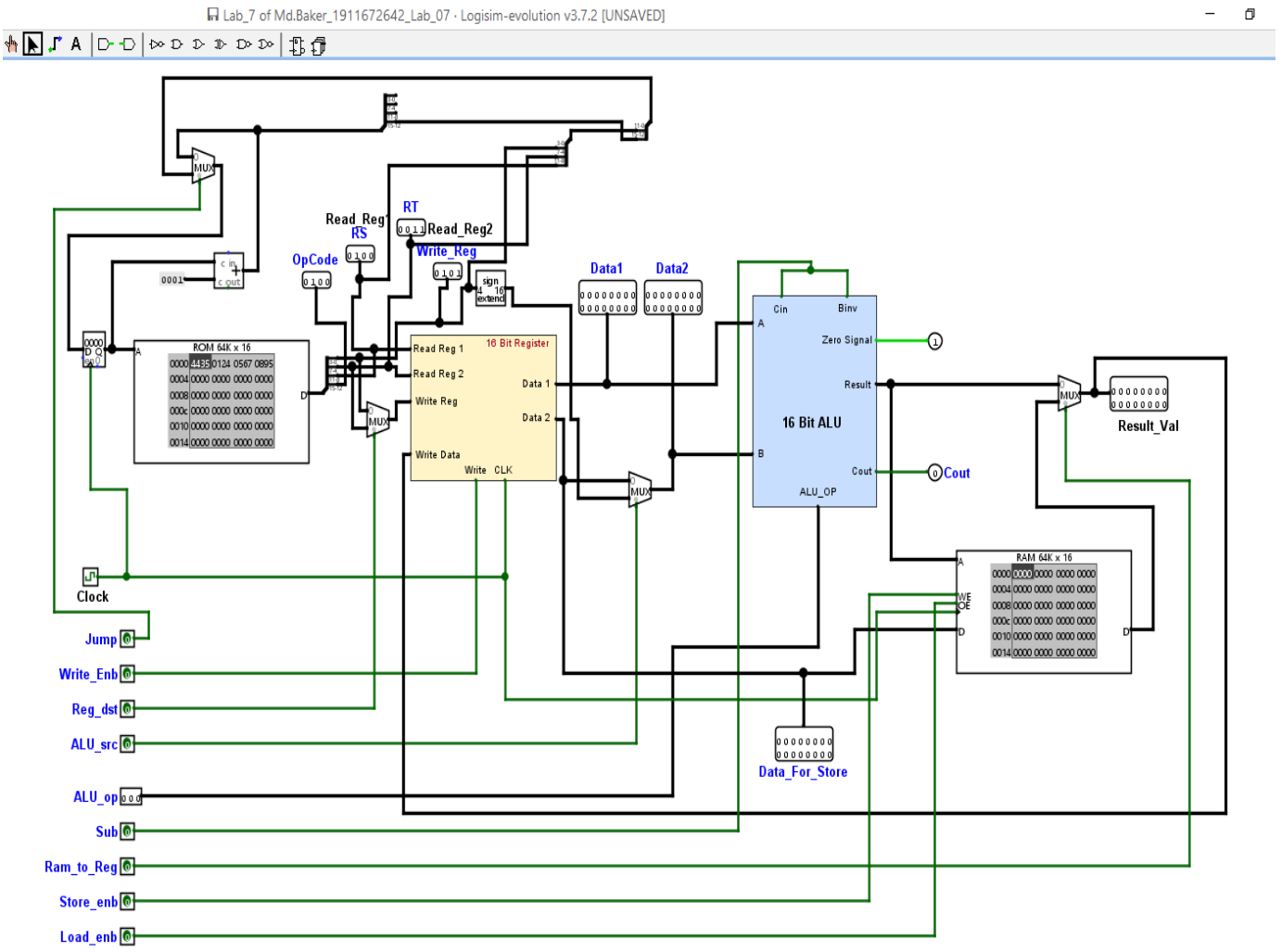


Figure: Single Cycle Datapath

Discussion:

In this lab, we built load-store type, jump type data-path and gathered all the data-path (R type, I-type, load-store, j-type) together.

Load-store type data-path:

load-store both are I-type instructions. In I-type instruction RD value will go as immediate value to directly ALU's second input. So, we need to use an extender to extend RD 4-bit to 16-bit. In the register file input, we have shotted RD with RT value. Now in ALU first input come from the rs register and another value immediately come to the ALU. Then we have used RAM, which first input is the result of ALU and 2nd input is RT. Then we have connected an input named (store) to str of RAM for storing the in-memory file. And another input named (load) to Id of RAM for loading in the register file. And the result of the RAM is connected to the writing data of the register file.

Jump data-path:

In jump instruction, first is op-code and then the rest of the instruction bits are target where our program counter (PC) have to jump. So, our lab instruction length -16-bit 12-bit will use as a target. So, we have to shot RS, RT, RD and make all these to total 12-bit. Then we have to add this 12-bit with our current instruction. For current instruction, we will use the adder output which kept instruction track in the instruction fetch lab. Now, the problem is our instruction track adder output is 16-bit but our target bit is 12-bit. To add both, we need to extend the 12-bit target to 16-bit input by bit extender. Now if we connect this adder result to directly PC (program counter), then it will always run like jump type. And we always don't want to jump. So, we need to add a multiplexer with 1 selector bit in which the first line is a normal instruction track counter and the 2nd line is jump instruction and that mux output is connected to PC.

All data-path together:

Now we have R-type, I-type, load-store type and jump type data-path. Gather all these was not difficult. We just added a few multiplexers to our circuit. First, we used MUX(Reg_dst) with 1-selector bit to denote the instruction is R-type or I-type in the register file inputs RT and RD. In I-type RD is an immediate value. So, we gave RS and RT to normal connection and in RD we used a mux outline. That mux first line is normal RD register addressing for R-type. And second line RD is shotted with RT for I-type instruction. Selector bit MUX = 0 - R-type. And MUX ==1 is I-type.

Then the second mux we used in the ALU's second input(B). In ALU, the first data is always coming from RS. And for 2nd data, in R-type format we use RT or in I-type

format, we use RD's extended value as immediate. So, here we used a 2 to 1 line mux which the first line is coming to form the register file data 2 and the 2ndline is immediate value extender output and the mux result is connected to ALU's B input. Selector bit
(MUX-2 (ALU src)) = if 0 - R-type-> register file data 2.
if 1 is I-type -> immediate value

After that, we have used another mux for writing into the register file. Sometimes we need to write ALU's result (for R-type, immediate type) or sometimes from RAM's output (for data loading in register from memory). Here we also used a 2 to 1 line mux which the first line is ALU result and 2nd line is RAM sending data and the mux output is connected to the register file input - writing data.

Selector bit (MUX-3 (Ram to Reg)) = if 0 is ALU result if 1 is RAM sending data. We had another mux in jump type. Selector bit (MUX- 4 (Jump)) = if 0 is normal program counter. if 1 is Jump instruction

We have checked few jump, load and store type instructions. Jump instruction run: j target PC will jump into the target value. load-store instruction run: lw $rt, immediate($rs) in load type, rs is the source register and rt is the destination register. And mem [rs inner value + immediate] - value will store in rt. sw rs, immediate($rt) where rs is the source register. The memory address of rs + immediate will store the value of rt. If We wanted to run all operations automatically, so we will design the control unit which would run the instruction according to the instructions.