CSE 306 Assignment 4
A1
Group 5

Md. Zarif Ul Alam : 1705010
Jamilus Sheium : 1705012
Naeem Ahmed : 1705014
Md. Shadmim Hasan Sifat : 1705021
Solaiman Ahmed : 1705022

July 4, 2021

1 Introduction

A processor or processing unit is a digital circuit which performs operations on some external data source, usually memory or some other data stream. The term is frequently used to refer to the Central Processing Unit(CPU) in a system. A central processing unit is the electronic circuitry that executes instructions comprising a computer program.The CPU performs basic arithmetic, logic, controlling, and input/output (I/O) operations specified by the instructions in the program.

Principal components of a CPU include the arithmetic logic unit (ALU) that performs arithmetic and logic operations, processor registers that supply operands to the ALU and store the results of ALU operations, and a control unit that orchestrates the fetching (from memory) and execution of instructions by directing the coordinated operations of the ALU, registers and other components.

Instruction pipelining is a technique for implementing instruction-level parallelism within a single processor. Pipelining improves performance by increasing instruction throughput, as opposed to decreasing the execution time of individual instruction.

In this assignment, we designed an 8-bit processor that supports pipelined datapath for a subset of MIPS instruction set. In this design,each instruction was divided into five stages :

1. Instruction Fetch (IF)
2. Instruction Decode (ID)
3. Execution and Address Calculation (EX)
4. Data Memory Access (MEM) and
5. Write Back (WB).

There are situations in pipelining when the next instruction cannot execute in the following clock cycle. These events are called hazards. For this assignment, we considered EX hazard, MEM hazard and double data hazard.

2 Complete Block Diagram

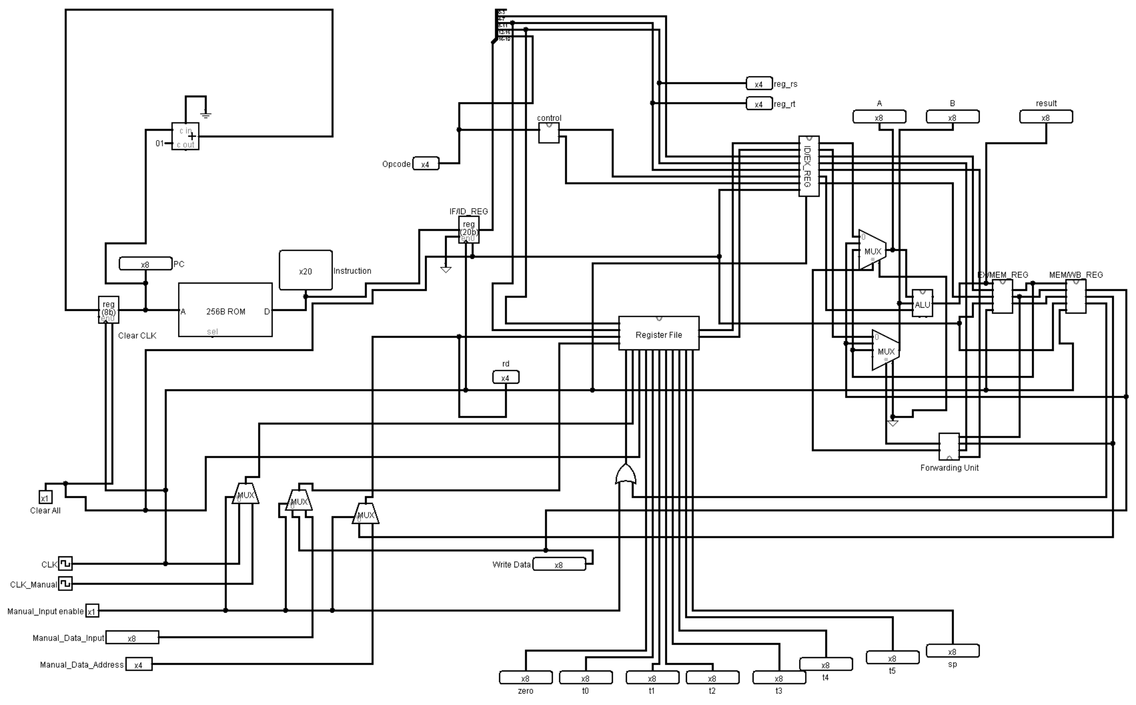

Figure 1: Complete Block diagram of an 8-bit MIPS processor

3 Block diagrams and size of pipeline registers

The sizes of pipelined registers :

- IF/ID 20 bits
- ID/EX 32 bits
- EX/MEM 13 bits
- MEM/WB 13 bits

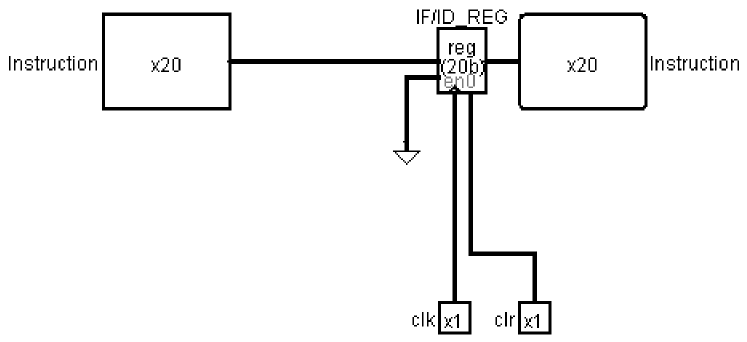

Figure 2: IF/ID Register

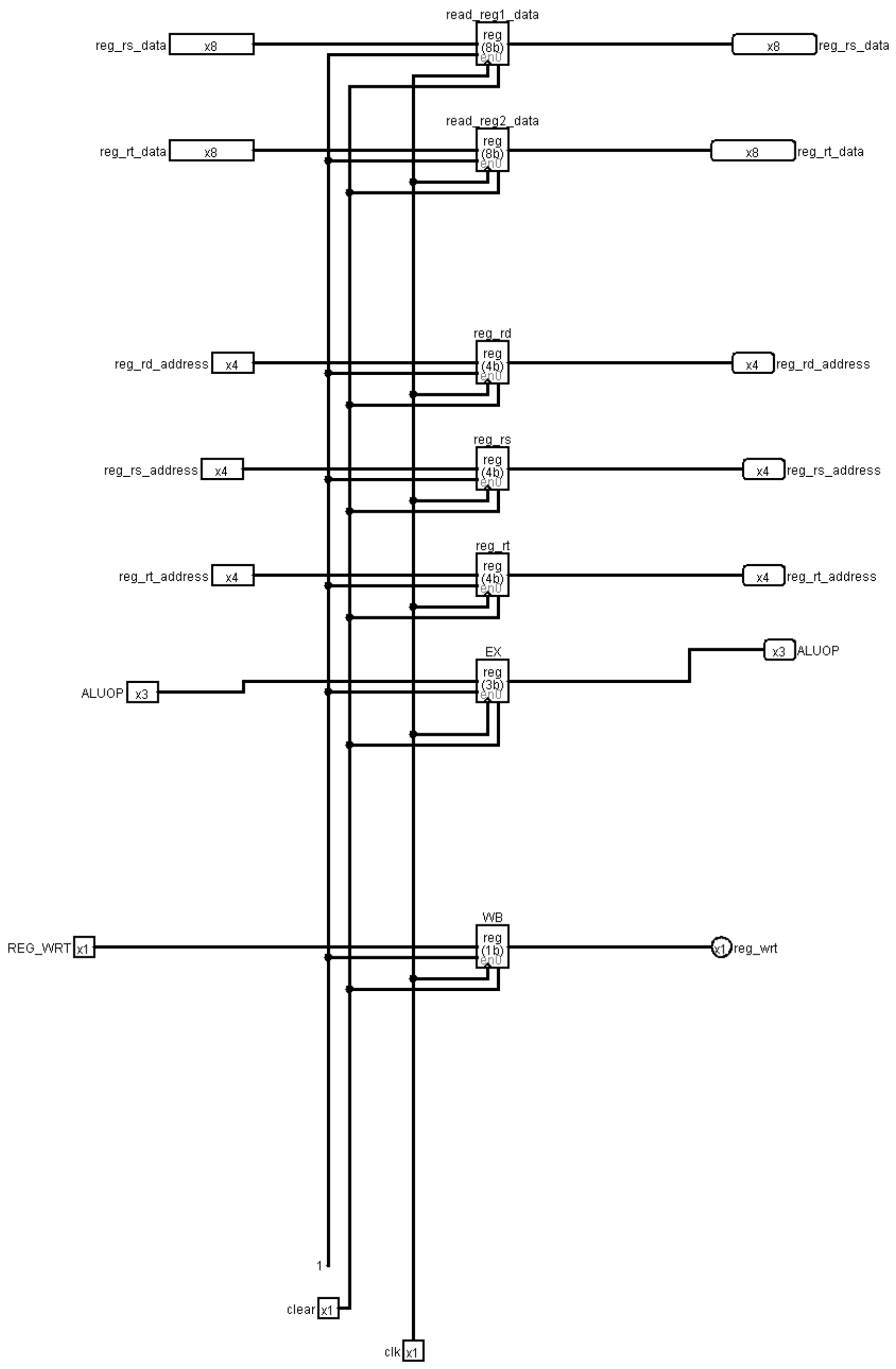

Figure 3: ID/EX Register

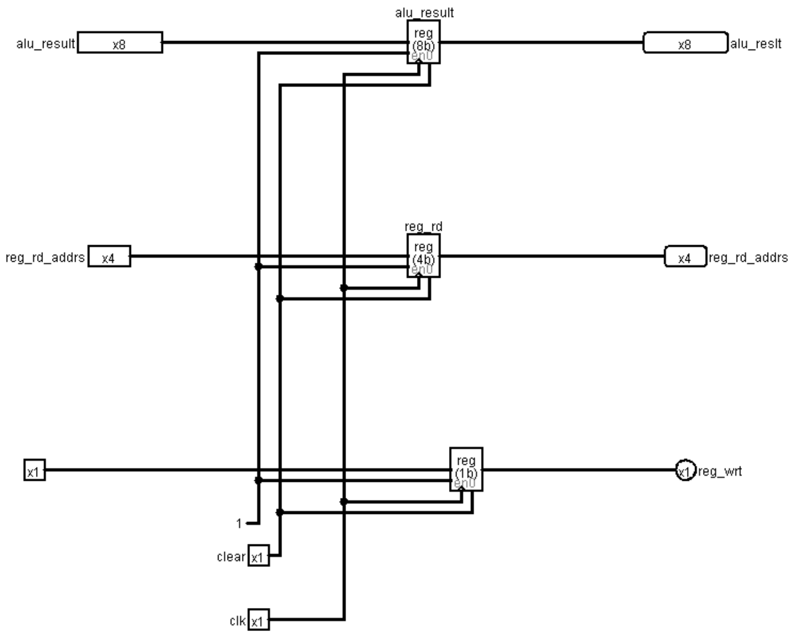

Figure 4: EX/MEM Register

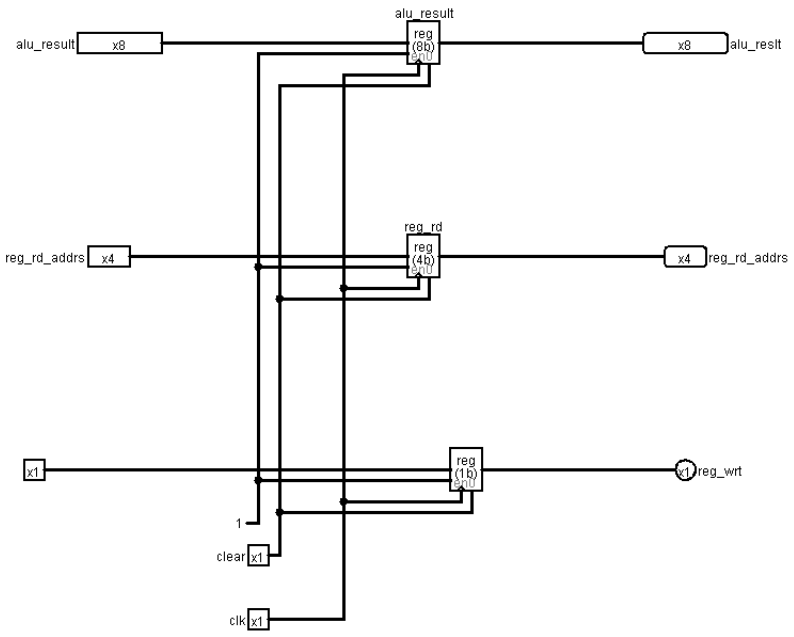

Figure 5: MEM/WB Register

4 Mechanism and block diagram of forwarding unit

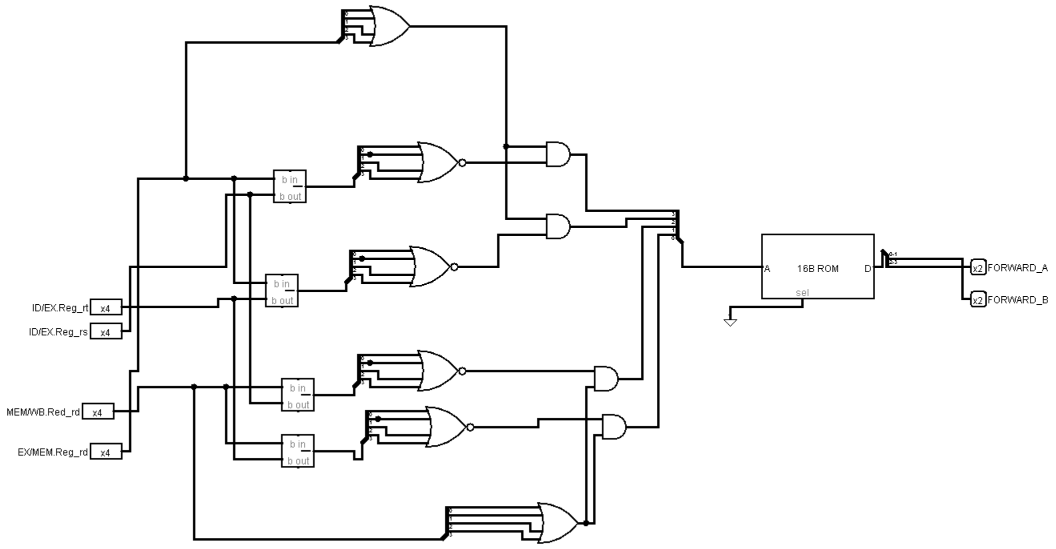

Figure 6: Forwarding Unit

The data hazards considered for this assignment are EX hazard, MEM hazard and Double Data hazard.

1. EX Hazard : The dependent instruction is in the EX stage and the prior instruction is in MEM stage.

   If $(EX/MEM.RegWrite$ and $(EX/MEM.RegisterRd \neq 0)$ and $(EX/MEM.RegisterRd = ID/EX.RegisterRs))$

   $$ForwardA = 10$$

   If $(EX/MEM.RegWrite$ and $(EX/MEM.RegisterRd \neq 0)$ and $(EX/MEM.RegisterRd = ID/EX.RegisterRt))$

   $$ForwardB = 10$$

2. MEM Hazard : The dependent instruction is in the EX stage and the prior instruction is in the WB stage.

   If $(MEM/WB.RegWrite$ and $(MEM/WB.RegisterRd \neq 0)$ and $(MEM/WB.RegisterRd = ID/EX.RegisterRs))$

   $$ForwardA = 01$$

   If $(MEM/WB.RegWrite$ and $(MEM/WB.RegisterRd \neq 0)$ and $(MEM/WB.RegisterRd = ID/EX.RegisterRt))$

   $$ForwardB = 01$$

3. Double Data Hazard : The dependent instruction is in the EX stage and it depends on two prior instructions, one of which is in the MEM stage, and the other is in the WB stage. For this new case, MEM Hazard condition is modified this way:

   If $(MEM/WB.RegWrite$ and $(MEM/WB.RegisterRd \neq 0)$ and $(MEM/WB.RegisterRd = ID/EX.RegisterRs))$ and $(EX/MEM.RegisterRd = ID/EX.RegisterRs))$

   $$ForwardA = 10$$

   If $(MEM/WB.RegWrite$ and $(MEM/WB.RegisterRd \neq 0)$ and $(MEM/WB.RegisterRd = ID/EX.RegisterRt))$ and $(EX/MEM.RegisterRd = ID/EX.RegisterRt))$

   $$ForwardB = 10$$

MUX will select values for ALU inputs as follows

ForwardA	ALU Input A
00	ID/EX.reg_rs_data
10	EX/MEM.ALUresult
01	MEM/WB.ALUresult

ForwardB	ALU Input B
00	ID/EX.reg_rt_data
10	EX/MEM.ALUresult
01	MEM/WB.ALUresult

5 ICs used with count as a chart

Gate	Gate Count	IC	IC Count
MUX(2-to-1)	6	74157	2
MUX(4-to-1)	2	74153	2
MUX(8-to-1)	1	74151	1
MUX(16-to-1)	2	74150	2
Adder(1-bit)	2	7480	1
Adder(4-bit)	4	7483	4
Adder(8-bit)	1	7483	2
Decoder(4-to-16)	1	74154	1
AND(2-bit)	13	7408	4
NOR(2-bit)	1	7402	1
NOR(4-bit)	4	4002	2
NOR(8-bit)	1	4078	1
Register(8-bit)	22	Register(8-bit)	22
ROM(256X20)	1	ROM(256X20)	1
ROM(16X13)	1	ROM(16X13)	1
Clock	2	Clock	2
Right-Shifter(8-bit)	1	Right-Shifter(8-bit)	1
Left-Shifter(8-bit)	1	Left-Shifter(8-bit)	1

6 Simulator

Logisim Version 2.7.1

7 Discussion

While implementing the circuit, we had to change our design several times.Some designs required a lot of ICs. To optimize number of ICs in our design, we had to discard those. In this assignment we had to deal with only R-type instructions. So we optimized our control unit for only R-type instructions . Also we discarded unnecessary components from the circuit and optimized the usage of registers. Again, we invested enough of our time in cross-checking corner cases for each sample test case cautiously.Considering all these aspects, we finally implemented the most optimized design we could find.