CSE306 - Computer Architecture Sessional

Assignment 3

8-bit MIPS Design and Simulation

Submitted by:

Group 2

1705062	Jawad Ul Kabir
1705063	Md. Mahfuzur Rahman Rifat
1705065	Taseen Mubassira
1705066	Ataf Fazledin Ahamed
1705067	Nishat Farhana Purbasha

21 June, 2021

1 Introduction

Microprocessor without Interlocked Pipe-lined Stages (MIPS) is a Reduced Instruction Set Computer (RISC) Instruction Set Architecture (ISA). The instructions of MIPS are rigid and follow fixed formats. There are 3 types of instruction of MIPS. Each instruction is 20-bits long and their formats are:

- R-Type

Opcode	Src Reg 1	Src Reg 2	Dst Reg	Shft Amnt
4-bits	4-bits	4-bits	4-bits	4-bits

- I-Type

Opcode	Src Reg	Dst Reg	Address/Immediate
4-bits	4-bits	4-bits	8-bits

- J-Type

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

2 Problem Specification

In this assignment, an 8-bit processor has to be designed that implements the MIPS instruction set. Each instruction will take 1 clock cycle to be executed. The length of the clock cycle will be long enough to execute the longest instruction in the MIPS instruction set. The main components of the processor are as follows: instruction memory, data memory, register file, ALU, and a control unit. Additional components such as multiplexers, adders etc can added as required by your design

- Address bus and data bus are multiplexed.
- Each of data and address has a size of 8-bits.
- An 8-bit ALU will be required, hence the name 8-bit MIPS.
- The register file must include the following temporary registers: **$zero, $t0, $t1, $t2, $t3, $t4**. Each register has a size of 8-bits. The assembly code that will be provided to simulate your design will use only the above mentioned registers.
- The control unit should be micro-programmed. The control signals associated with the operations should be stored in a special memory (you can use a separate ROM for this purpose) units as Control Words.
- All clocks required in the circuit must be provided from a single clock source. Each instruction should be fetched and executed in a single clock cycle.

3 Instruction Set

Control bit representation:

12	11	10	09	08	07	06 - 04	03	02	01
Reg(D)	Jump	Br Eq.	Br Neq.	Mem(R)	MemToReg	ALUOp(3)	Mem(W)	ALU(S)	Reg(W)

ID	Name	Type	Opcode	Control	Control (Dec)	Control (Hex)
K	nor	R	0000	100000 100 001	2081	821
A	add	R	0001	100000 000 001	2049	801
G	or	R	0010	100000 010 001	2065	811
B	addi	I	0011	000000 000 011	3	003
L	sw	I	0100	000000 000 110	6	006
I	sll	R	0101	100000 101 001	2089	829
P	j	J	0110	010000 000 000	1024	400
F	andi	I	0111	000000 011 011	27	01B
N	beq	I	1000	001000 001 000	520	208
J	srl	R	1001	100000 110 001	2097	831
M	lw	I	1010	000011 000 011	195	0C3
D	subi	I	1011	000000 001 011	11	00B
C	sub	R	1100	100000 001 001	2057	809
H	ori	I	1101	000000 010 011	19	013
E	and	R	1110	100000 011 001	2073	819
O	bneq	I	1111	000100 001 000	264	108

4 Diagram of 8-bit MIPS

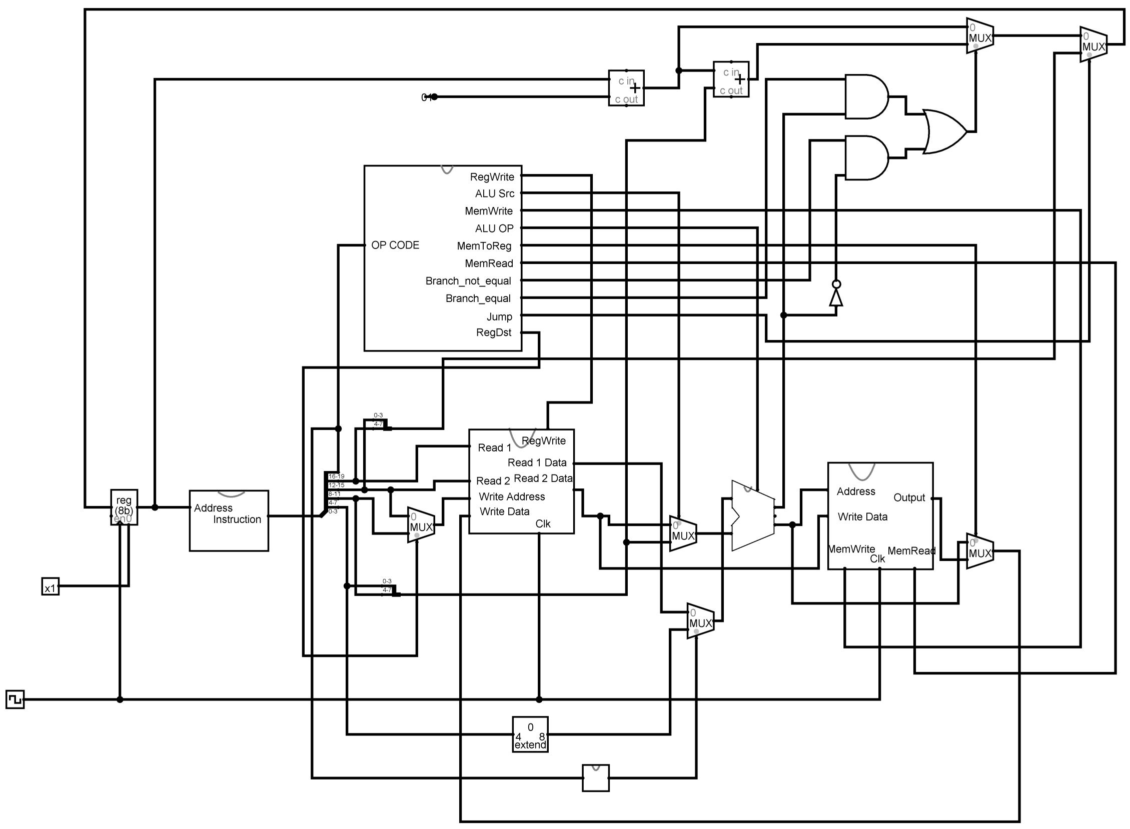

Figure 1: 8-bit MIPS

5 Diagram of the Main Components

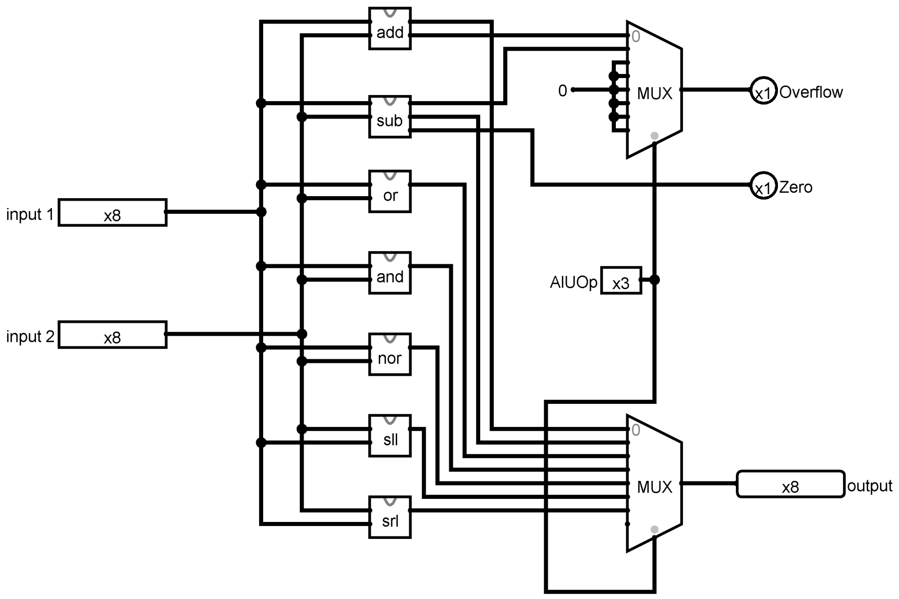

Figure 2: 8 bit ALU

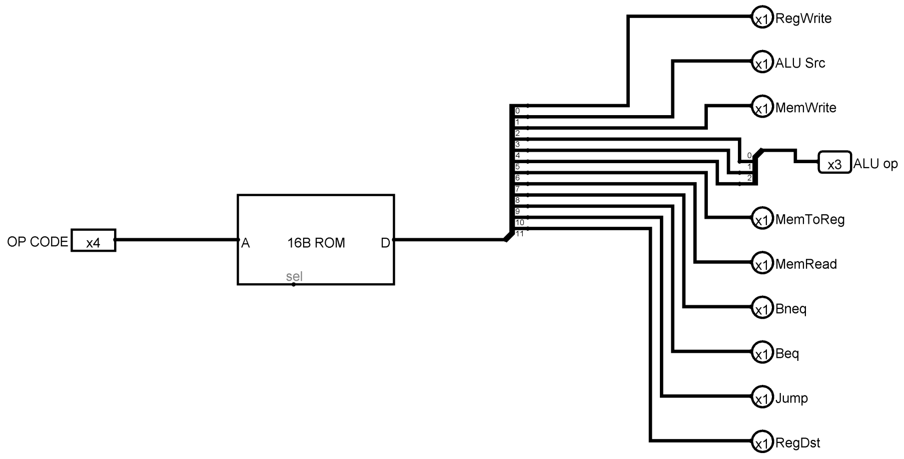

Figure 3: Control Memory

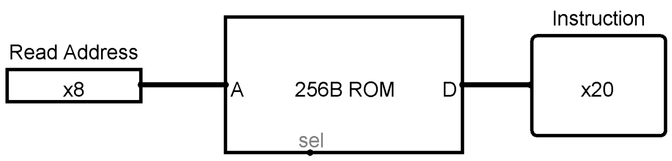

Figure 4: Instruction Memory

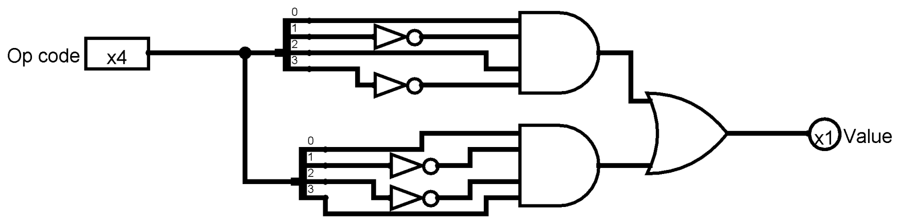

Figure 5: Shift Logical Unit

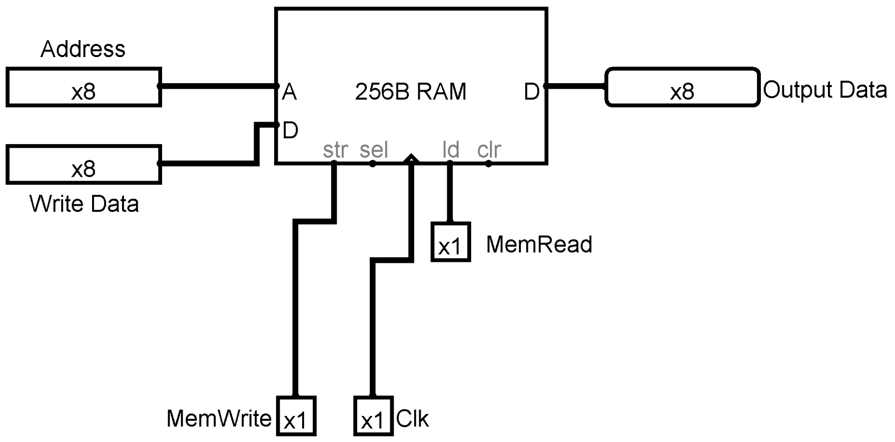

Figure 6: Data Memory

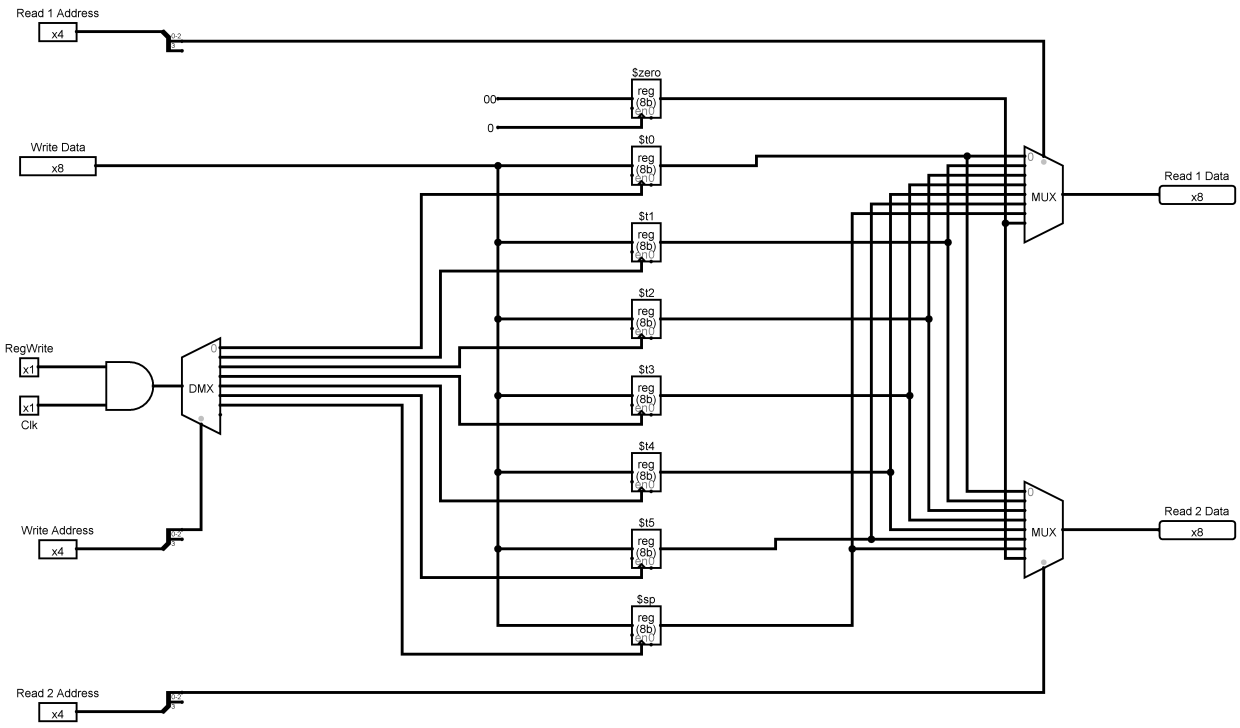

Figure 7: Register File

6 Approach to Implement the *Push* and *Pop* Instructions

push and **pop** instructions are converted into several MIPS instructions (**lw, sw, addi, subi**). For **push $tx** type instructions, first the value of $tx register is stored in the head of the stack memory. Then the value of $sp is decreased by 1-

```
sw $tx, 0($sp)
subi $sp, $sp, 1
```

For **push n($tx)** type instructions, first the value of n($tx) is loaded into $t5 register and then pushed like before-

```
lw $t5, n($tx)
sw $t5, 0($sp)
subi $sp, $sp, 1
```

For **pop $tx** type instructions, the reverse of push is done. First the value of $sp register is increased by by 1. Then the value is loaded from the stack into $tx register-

```
addi $sp, $sp, 1
lw $tx, 0($sp)
```

7 ICs and Components Used

Name	**IC**	**Quantity**
8 bit Register	-	9
ROM	-	2
RAM	-	1
8 bit 1:8 DEMUX	-	1
8 bit 8:1 MUX	-	4
8 bit 2:1 MUX	-	5
4 bit 2:1 MUX	-	1
Shifter	-	2
8 bit Negator	-	1
Bit Extender (4 to 8 bit)	-	2
4 bit Adder	IC 7483	8
AND	IC 7408	2
OR	IC 7432	1
NOT	IC 7404	1

8 Simulator Info

Logisim - Java Platform (Version 2.7.1)

9 Discussion

- A register file containing eight different 8-bit-register was designed. The registers are as following: **$zero, $t0, $t1, $t2, $t3, $t4, $t5, $sp**. The $sp register is used to store the stack pointer and the $zero register as holding constant 0 value.
- MIPS processor instructions implemented by us is consisted of 20 bits whereas typical MIPS instructions are 32 bit long. Those 20 bits are interpreted differently according to main three MIPS instruction type.
- All clocks in our designed circuit was provided from a single clock source. And each instructions were fetched and executed in a single clock cycle. The exception is of course- the **push and pop** feature. Since the operations of stack are not possible within a single clock pulse, a total of 3 clock pulse were required to execute the push-pop operation of the stack.
- Two different ROMs were used for Instruction memory and Control Memory. And a RAM was used as Data Memory since it provides read/write feature.