CSE 306 (Computer Architecture Sessional)

Experiment No: 03

Name of the experiment:

8-bit MIPS Design and Simulation

Group No.	**06**
Section	A1
Department	CSE
Group Members	1705026 1705027 1705028 1705029 1705030
Date of Performance:	17-06-2021
Date of Submission:	19-06-2021

Introduction:

The prime objective of this assignment was to design an8-bit processor which implements the MIPS instruction set. It took 1 clock cycle to execute each instruction. The length of the clock cycle was long enough so that the longest instruction on the MIPS instruction set was executed in a single clock cycle. Instruction memory, Data memory, Register file, ALU, Control unit were the main components of the processor. These components were designed and properly connected using some available multiplexers, adders and wires.

Instruction Set:

Decimal (Opcode)	Instruction ID	Category	Type	Instruction
0(0000)	P	Control-unconditional	J	jump
1(0001)	O	Control-conditional	I	bneq
2(0010)	F	Logical	I	andi
3(0011)	M	Memory	I	lw
4(0100)	B	Arithmetic	I	addi
5(0101)	N	Control-conditional	I	beq
6(0110)	D	Arithmetic	I	subi
7(0111)	E	Logical	R	and
8(1000)	C	Arithmetic	R	sub
9(1001)	A	Arithmetic	R	add
10(1010)	G	Logical	R	or
11(1011)	I	Logical	R	sll
12(1100)	H	Logical	I	ori
13(1101)	K	Logical	R	nor
14(1110)	L	Memory	I	sw
15(1111)	J	Logical	R	srl

Control Bits:

Decimal (Opcode)	Control Bits	Control Bits In Hex
0(0000)	000 0111 00010b	0E2h
1(0001)	000 0001 00100b	024h
2(0010)	110 0011 00000b	C60h
3(0011)	110 0000 11000b	C18h
4(0100)	110000000000b	C00h
5(0101)	000 0001 00100b	024h
6(0110)	110 0001 00000b	C20h
7(0111)	100 0011 00001b	861h
8(1000)	100 0001 00001b	821h
9(1001)	100000000001b	801h
10(1010)	100 0010 00001b	841h
11(1011)	110 1101 00001b	DA1h
12(1100)	110 0010 00000b	C40h
13(1101)	100 0100 00001b	881h
14(1110)	011 0000 00000b	600h
15(1111)	110 1110 00001b	DC1h

ALU Opcode:

Opcode	Function
000	ADD
001	SUB
010	OR
011	AND
100	NOR
101	SLL
110	SRL

Complete Block Diagram:

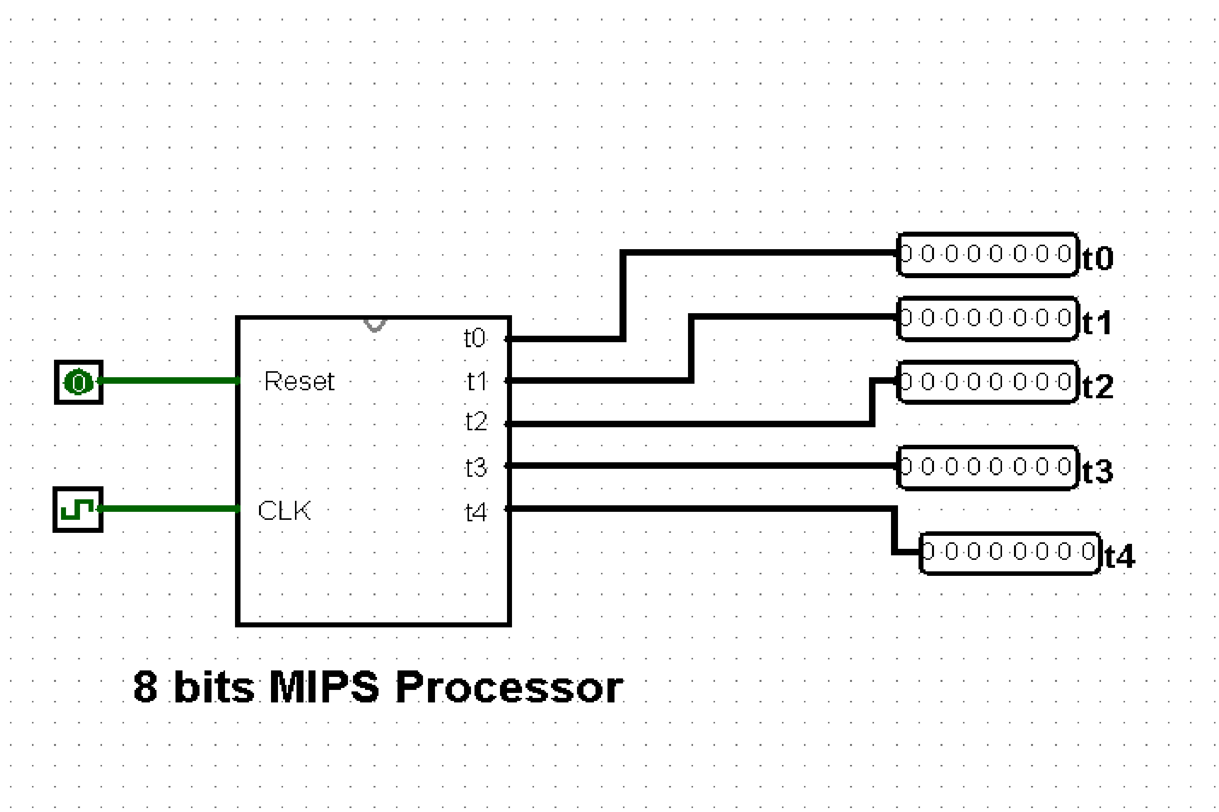

Figure 1:Block Diagram

Main Circuit:

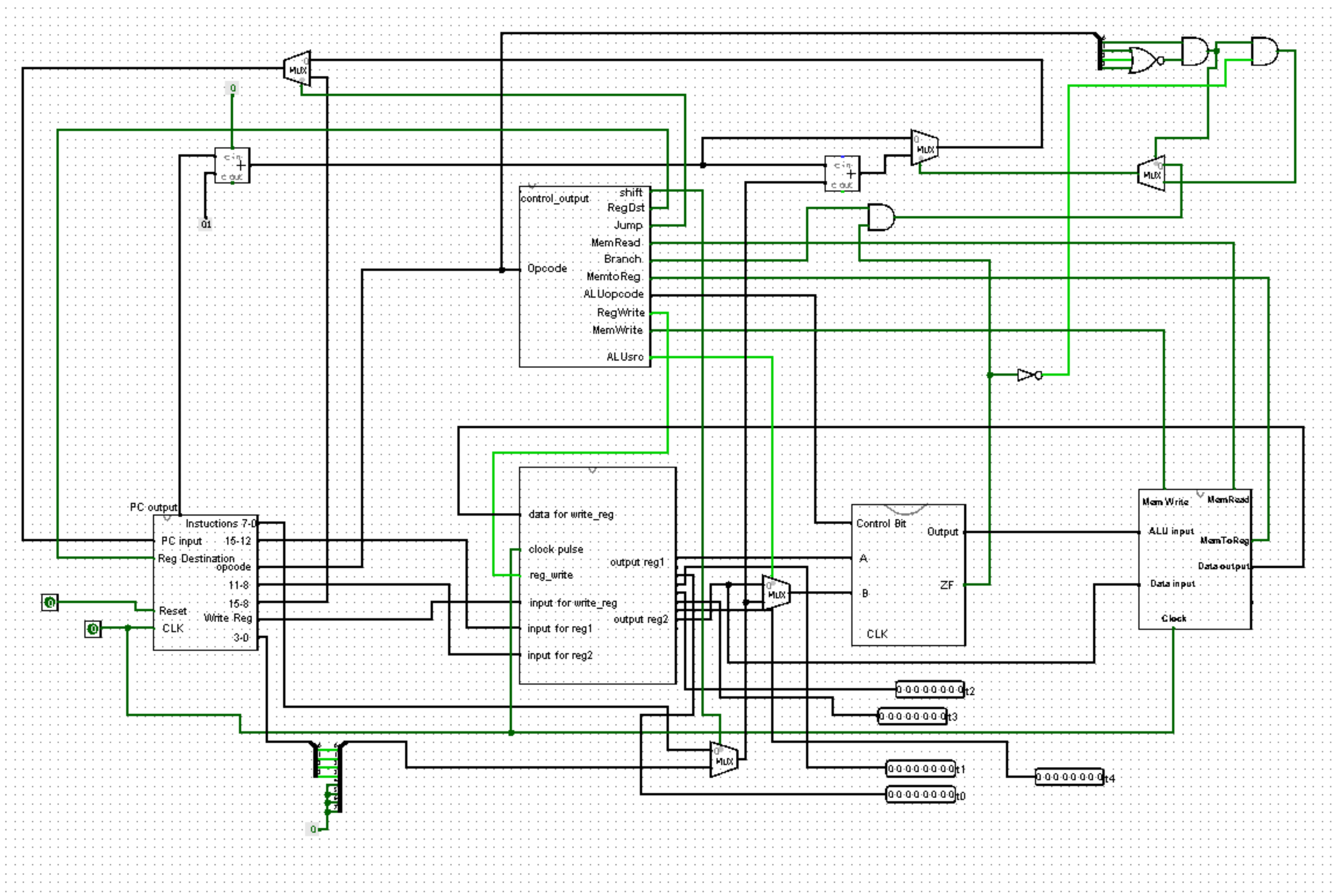

Figure 2: 8 Bit MIPS Processor

Detailed Circuit Diagram:

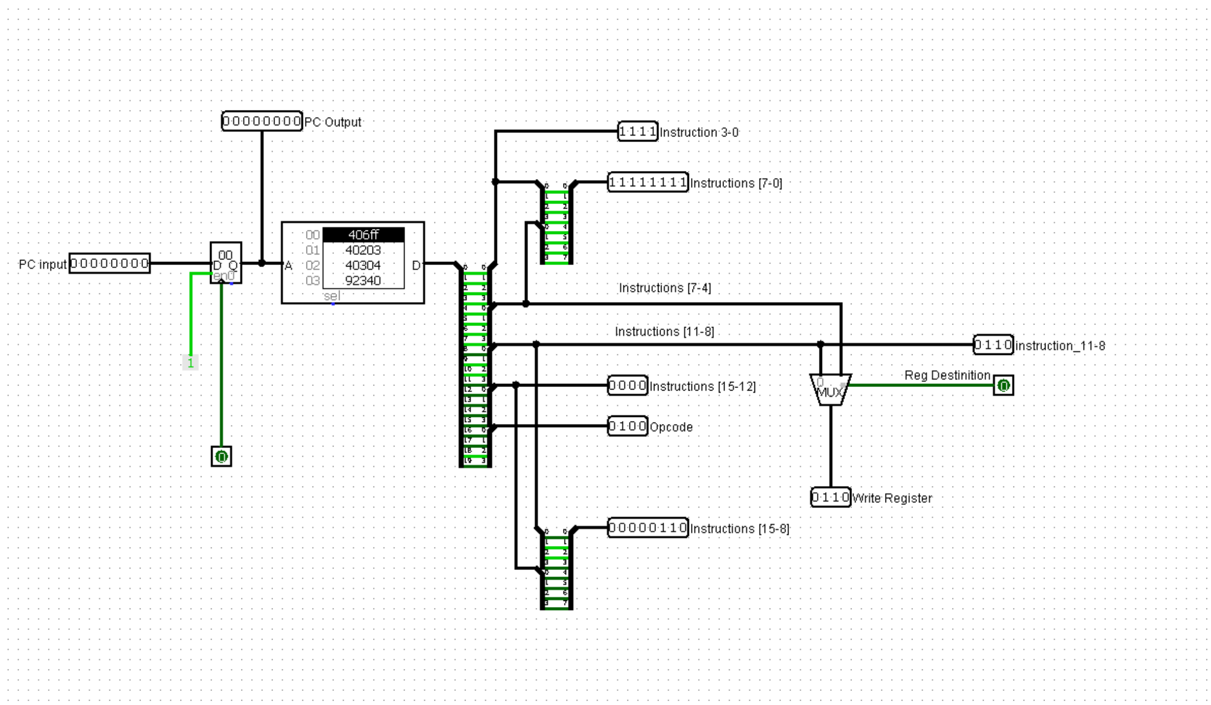

Figure 2(b): Instruction Memory

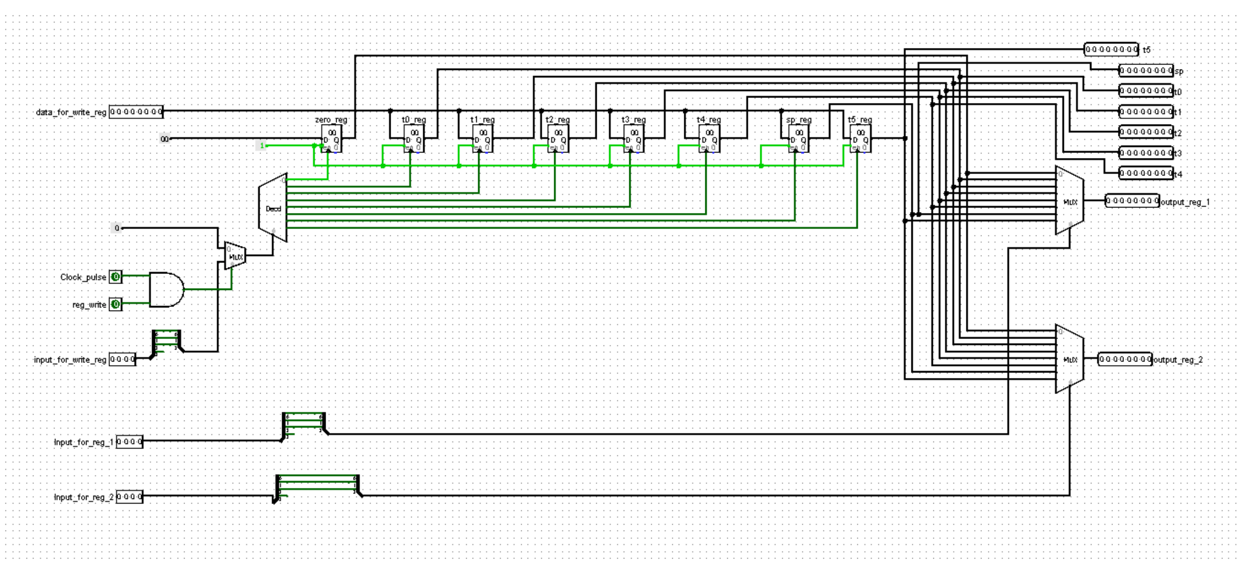

Figure 2(c): Registers

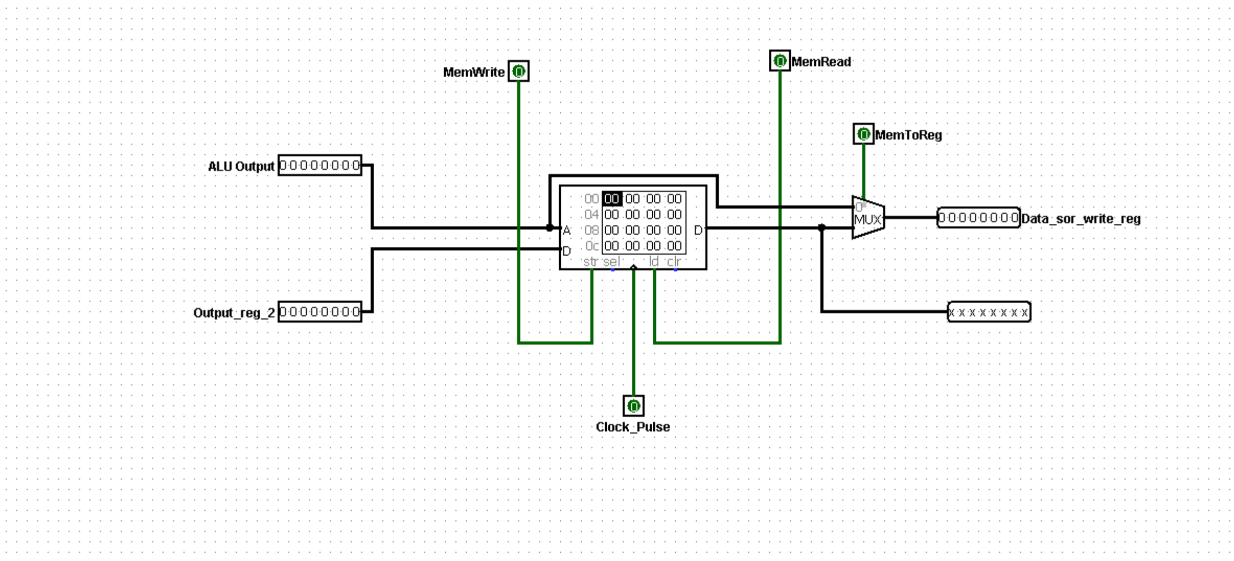

Figure 2(e): Data memory

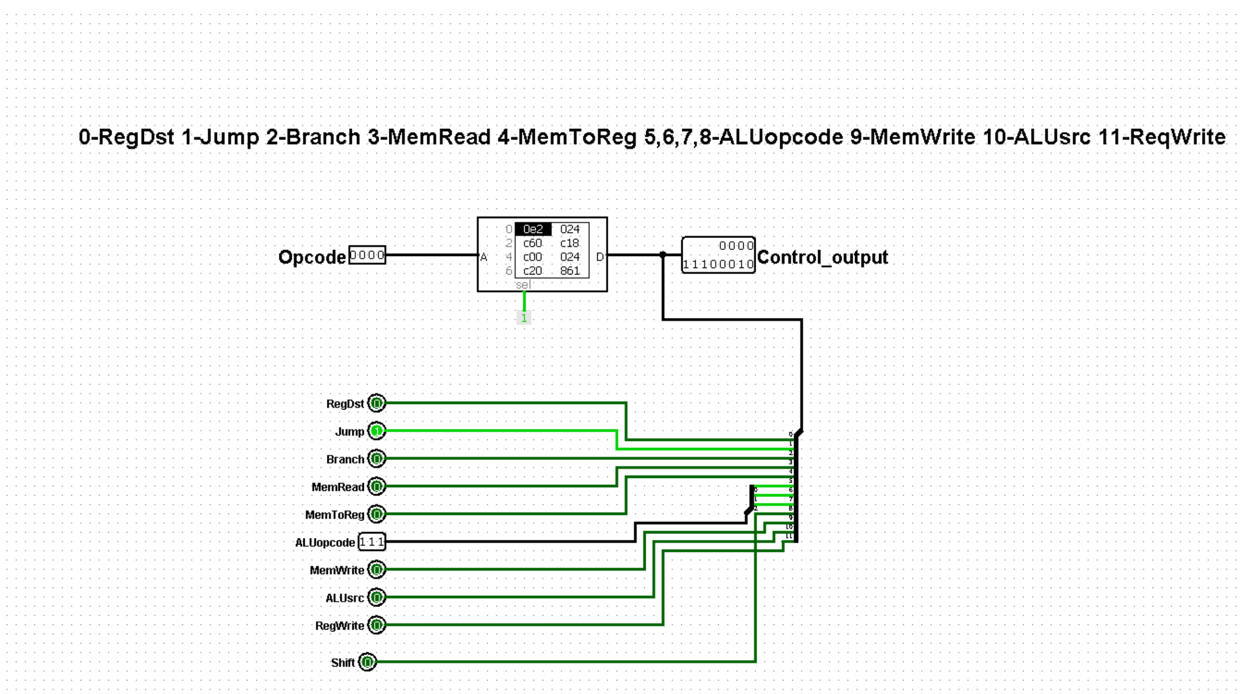

Figure 2(a): Control Unit

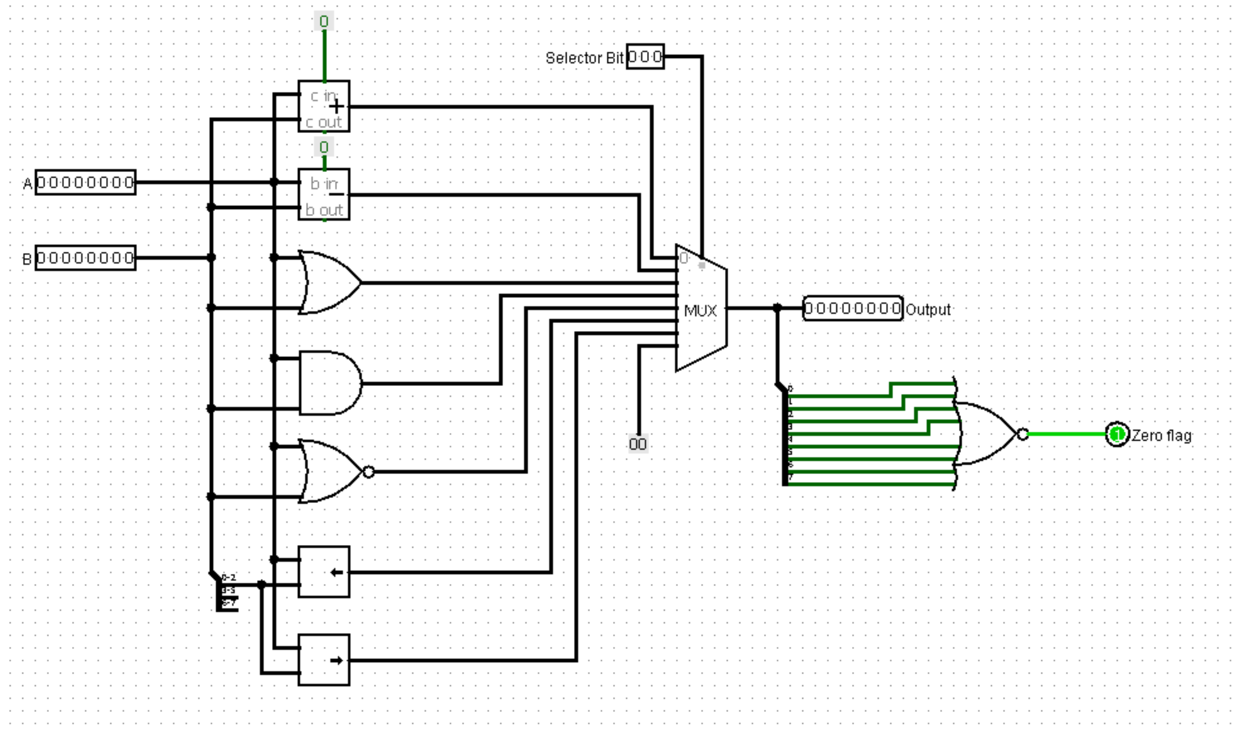

Figure 2(d): ALU 8bit

Approach to implement the push and pop instructions:

There is no push, pop instructions in MIPS, but it was used as a pseudo instruction. A pseudo instruction is an instruction which is not available directly but can be converted to multiple simple available instructions by the assembler.

If we got a push $t0 instruction, we converted it to a subi and a sw instruction. First we subtracted 1 from the stack to allocate memory for the data to be pushed, then stored the data which is the value of $t0 on the top of the stack.

push $t0 was converted to

subi $sp, $sp, 1

sw $t0, 0($sp)

so push $t0 required two clocks.

similarly if we got a push 3($t0) instruction, we took the following approach:

We subtracted 1 from the stack but before inserting the data we used a lw instruction because here $t0 contains an address in the data memory and we wanted to store the data located at

$t0+3 address of the memory in $t5 register. The $t5 register was used in the circuit only for this purpose.

So push 3($t0) is converted to

subi $sp, $sp, 1

lw $t5, 3($t0)

sw $t5, 0($sp)

so push 3($t0) required three clocks.

For pop instruction, we performed the opposite operation of push. We first loaded the data from the top of the stack in the register, then added 1 to the stack pointer.

So pop $t0 was converted to

lw $t0, 0($sp)

addi $sp, $sp, 1

so pop $t0 required two clocks.

IC Count:

Used IC	Operation	Count
IC 74LS04	NOT	1
IC 74LS08	AND	3
IC 74LS32	OR	2
IC 74LS02	quad 2-input NOR gate	3
IC 74LS27	triple 3-input NOR gate	2
IC 74LS83	4 bit Full Adder	6
NA	8 bit Full subtractor	1
IC 74LS157	2:1 Multiplexer	8
IC 74LS151	8:1 Multiplexer	3
IC 74LS138	3-bit to 8-line Decoder	1
NA	8 bit register	9
NA	256B*20 ROM	1
NA	16B*12 ROM	1
NA	256B*8 RAM	1

Total Chips Needed: 42

Simulator Used: Logisim

Version Number: 2.7.1

Discussion:

In this assignment, an 8-bit processor that implements MIPS instruction set was designed using Logisim. The main components of the processor were designed as per the need of the specifications provided. The register file has 8 registers in total with 7 temporary registers and 1 stack pointer register. An extra temporary register was used to implement the push instruction associated with memory. One extra addi instruction wasadded at the beginning of each instruction file to initialize the stack pointer register to 0XFF. A ROM was used as the instruction memory. On the other hand, both read and write are necessary in the data memory, hence a RAM was used here. A separate ROM was used for control unit and the control signals were mapped against the opcode of the individual instructions. An 8 bit ALU that takes two 8 bit inputs was used. As both of the inputs of the ALU are 8 bit long, the 4 bit shift amount was extended to 8 bits and then sent to the ALU in case of sll and srl instructions. Here, a separate register was kept as PC. As in byte addressing mode all the instructions are 8 bit long, PC was incremented by 1 in case of normal flow and the address was fed to the instruction memory.

Using minimal number of ICs in order to reduce complexity was the priority while designing the circuit. Outputs of some of the intermediate gates and ICs were reused to minimize the circuit even more.

The connections were made carefully and the components were placed at fair distances. Messiness was avoided as much as possible to ensure higher readability. Suitable labels were used at different parts of the circuit to make the functions of individual parts more understandable. Finally, the circuit was tested several times to make sure it did not have any sort of error.