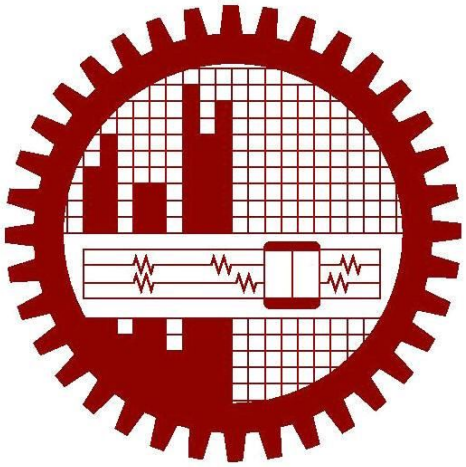

Bangladesh University of Engineering and Technology

CSE 306

Offline 3

8-bit MIPS Design and Simulation

GROUP : 01
SECTION : A2

Prepared By:

1605031
1605033
1605038
1605041
1605042

DEPARTMENT of COMPUTER SCIENCE and ENGINEERING

Introduction:

In this assignment we have to design an 8-bit MIPS and simulate it in a software. Both data and address is of 8-bits. We have to use 8-bit ALU. We have to design the temporary registers: $zero, $t0, $t1, $t2, $t3, $t4 and $sp which are of 8-bits and we assume the assembly code that is provided to simulate our design uses only these mentioned registers. The Control unit is microprogrammed. We have written the conversion from assembly code to MIPS machine code in cpp for this assignment.

Instruction Set:

Opcode 3	Opcode 2	Opcode 1	Opcode 0	Instruction	Instruction ID
0	0	0	0	add	A
0	0	0	1	or	G
0	0	1	0	bneq	O
0	0	1	1	srl	J
0	1	0	0	j	P
0	1	0	1	nor	K
0	1	1	0	andi	F
0	1	1	1	beq	N
1	0	0	0	sub	C
1	0	0	1	sll	I

1	**0**	**1**	**0**	**lw**	**M**
1	**0**	**1**	**1**	**addi**	**B**
1	**1**	**0**	**0**	**sw**	**L**
1	**1**	**0**	**1**	**and**	**E**
1	**1**	**1**	**0**	**ori**	**H**
1	**1**	**1**	**1**	**subi**	**D**

How to write and execute a program in this machine :

At first we have to create a text file where we have to write the assembly code. Then we wrote a cpp program which converts the given assembly code to our MIPS machine code and writes it in a hex file. Then if we load the hex file in our main circuit and give appropriate clock signal, in each clock signal one instruction from the MIPS machine code will be read and then be executed in the circuit.

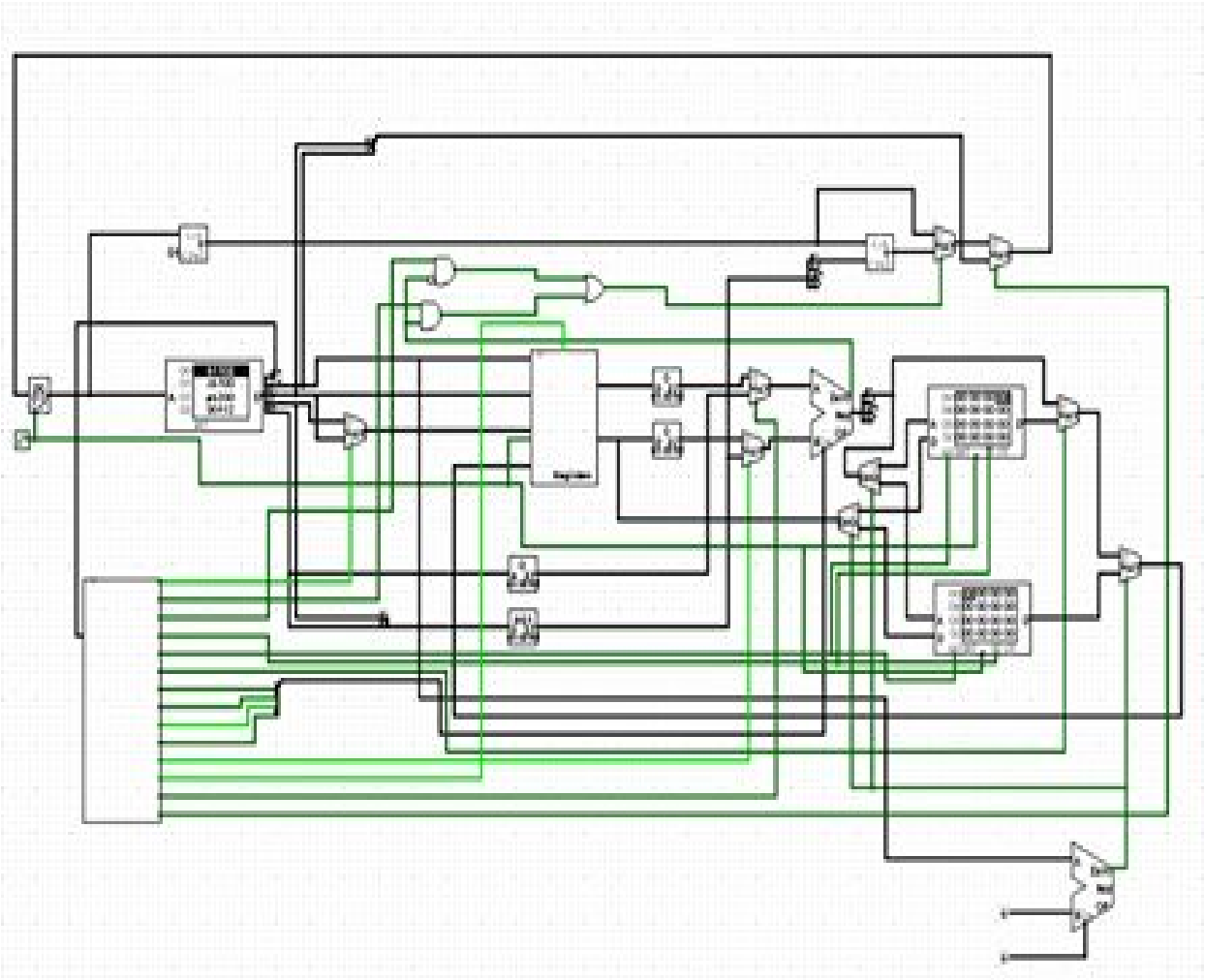

Fig: Circuit Diagram

IC's used with count:

Multiplexer	8
Demultiplexer	4
AND gates	28(IC's : 7)
OR gates	9 (IC's : 3)

NOT gates	35 (IC's : 6)
ADDER	2
ALU	2
BIT Extender	4
RAM	2
ROM	1
Registers	8

Simulator used:

Logisim 2.7.1

Discussion:

a) We tried to follow the MIPS datapath taught in our class.

b) We tried to make our design clean and easily understandable.

c) We made sure the wirings are connected properly in the simulation.

d) We used all the required controls in our simulation.

e) At first we couldn't save the values properly in our registers, but later we found out that we didn't connect the clock input properly to the registers. We found out the problem and solved it.

f) We sometimes mixed up the input to the MUX's and thus got wrong values. But we realized what we did wrong and corrected it.

g) We had to set the source registers and destination registers properly for each type of instruction. We made sure that the registers are getting proper values according to each instruction type.

h) At first we set the RegWrite control to 1 to every register. So when one register value was updated, all the other register values updated as well. We found our mistake and solved it.

i) We forgot to set the enable to the ram. So nothing was written to data memory at first. Finally we set the enable accordingly and our circuit was working.

j) We used two different controls for beq and bneq instruction. But by mistake we set the control for beq instruction to 1 for both the beq and bneq instructions which caused our circuit to malfunction. But finally we could find the fault in our design and fixed it.

Although we faced some problems while designing and implementing the 8-bit MIPS, we finally could make our circuit work properly.