# CSE 306

# Computer Architecture Sessional

# Assignment on 8-bit MIPS Design and Simulation

#### Group: 6

#### Section: A2

#### Roll:

#### 1605055

#### 1605056

#### 1605057

#### 1605058

#### 1605060

##### Introduction:

In this session we will construct a RISC architecture computer. This computer can run subset of MIPS assembly language. Here both data and address bus are 8 bits in width as well as the ALU. So, it is a 8-bit computer.

##### Instruction Set:

Our MIPS computer use this following instruction set:

|  |  |
| --- | --- |
| Opcode | Operation |
| 0 0 0 0 | or |
| 0 0 0 1 | bneq |
| 0 0 1 0 | sub |
| 0 0 1 1 | andi |
| 0 1 0 0 | srl |
| 0 1 0 1 | subi |
| 0 1 1 0 | addi |
| 0 1 1 1 | sw |
| 1 0 0 0 | and |
| 1 0 0 1 | lw |
| 1 0 1 0 | sll |
| 1 0 1 1 | add |
| 1 1 0 0 | ori |
| 1 1 0 1 | j |
| 1 1 1 0 | beq |
| 1 1 1 1 | nor |

##### Circuit:

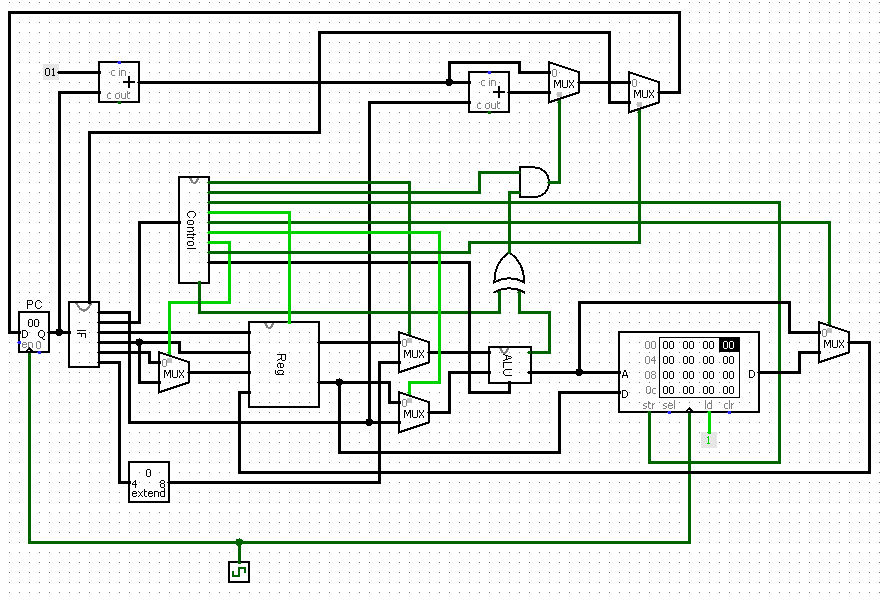


Figure 1: Circuit Diagram

##### How to Run Program:

1. Our assembler makes an image file which can be uploaded in Logisim.
2. After opening IF sub-circuit we have to upload Logisim image file and reset the simulation.
3. After enabling simulation, we have to enable ticks so that the computer will get required clock cycles.
4. The program will execute automatically.

Special Features:

1. Registers: We have implemented total 16 registers. So this computer can work with not only mentioned temporary registers but also storage and stack pointer registers.
2. Additional Instruction: Some instructions are implemented which can be used throughout assembly code. They are:
   1. mov: Performs move operation like 8086 assembly.
   2. inc: Increments by one.
   3. dec: Decrements by one.
   4. push: Pushes to stack using $sp.
   5. pop: Pops from stack using $sp.
   6. slt: Works as “set on less than command” in MIPS .
   7. muli: Can perform intermediate multiplication.

These instructions are performed by doing basic instructions. So, they may take several clock cycles.

##### ICs and Their Count:

It is hard to exactly determine ICs and their count when the simulation is implemented in Logisim. Here is a rough estimation:

Name Quantity

1. OR gate: 1
2. AND gate: 2
3. NOT gate: 1
4. 8 bit adder: 3
5. ROM: 2
6. RAM: 1
7. 8 bit register: 17
8. XOR gate: 2
9. 8 bit 2-1 MUX: 22
10. 8 bit 8-1MUX: 1
11. 8 bit 16-1 MUX: 2
12. 4-16 decoder: 1
13. Shifter: 2
14. 8 bit comparator: 1
15. Sign extender: 3

#### Discussion:

All instruction provided in the specification takes one clock cycle. So, CPI of this computer is 1. But additional derived instructions make take more than one clock cycle. For example: push $t0 instruction is formed into two basic instructions: add $sp, $sp, -1

sw $t0, 0($sp)

So, push instruction will take two clock cycles.

In addition, stack and memory shares same address space. As all registers are set to zero in the beginning of the simulation, first push operation stores the value at the address of FF.