CSE 306
Computer Architecture Sessional

Assignment 03: 8 bit MIPS Design and Simulation
Report of Group 01 Section B1

Prepared by:
1705061 - Maksudur Rahaman Rana
1705064 - Tanvir Raihan
1705081 - Md. Kamrujjaman
1705083 - Hozifa Rahman Hamim
1705090 - Ashrafi Zannat Ankon
1305050 - Zaki Tahmeed

Introduction

In this assignment, we need to design an 8-bit MIPS processor and simulate it using Logisim. The design contains the basic blocks like Progran Counter(PC), Instruction Memory, Register File and Data Memory. Additionally, we can use 8-bit ALU, multiplexers,separate memory devices -RAM,ROM and registers. The main objective of this assignment is to understand the flow of execution of diffrent instruction formats(R,I,J). This implementation requires single-cycle execution of the instructions.

Instruction Set

The assigned instruction set for our group is : IEMFPDJNKLCHGBAO, which includes

- **7 R-format instructions** : I(sll),E(and),J(srl),K(nor),C(sub),G(or),A(add)
- **8 I-format instructions** : M(lw),F(andi),D(subi),N(beq),L(sw),H(ori),B(addi),O(bneq)
- **1 J-format instruction** : P(j)

The instruction set defines the opcode for the instructions in an ascending order i.e. I has opcode 0,E with opcode 1 and upto O with opcode 15. The opcodes plays an important part in identifying the type of instruction MIPS gets from the Instruction Memory.

Complete Block Diagram of the MIPS processor

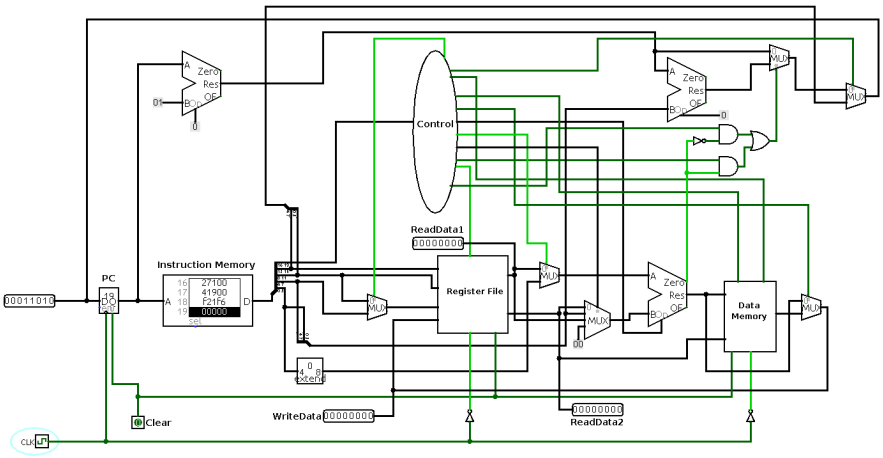

Figure 1: Block Diagram of the 8-bit MIPS Processor

Block Diagram of the Main Components

Instruction Memory with PC

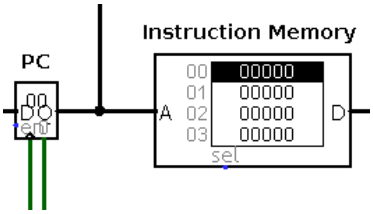

Figure 2: PC with Instruction Memory

The value of PC control the Instruction flow by indicating the instruction to be executed. Generally, the value of PC increments by 1 for 8bit MIPS, but branch instruction or jump instruction changes the value of PC differently. We have used a ROM to read instructions corresponding to the address PC sends to it.

Register File

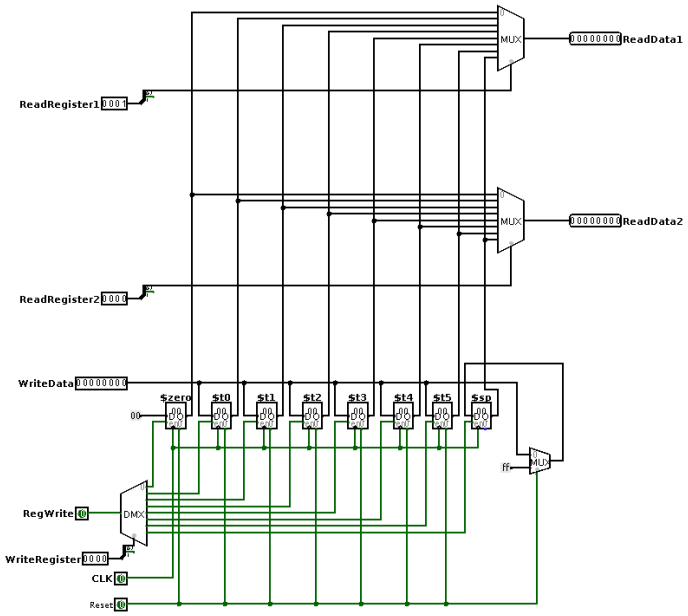

Figure 3: Register File

The register file holds all the temporary registers required to execute the instructions. The register file is used to read data from the registers as well as

to write data to the register when RegWrite is enabled. Write register input is used to indicate which register will be written with the data register file receives.

Data Memory with Stack

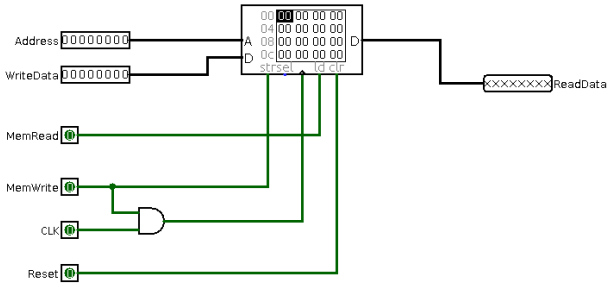

Figure 4: Data Memory

The data memory is used to read from and write to memory. It has two input signals: MemRead and MemWrite. Data is read from memory if MemRead signal is enabled by the control unit and data is write to memory if MemWrite is enabled. Data Memory unit shares memory with stack. The ROM is maintained in such a way that data memory starts from 0x00 and stack memory starts from 0xFF and the memories grow in opposite direction. In case of push and pop operation,stack memory is accessed and read and written accordingly.

Control Unit

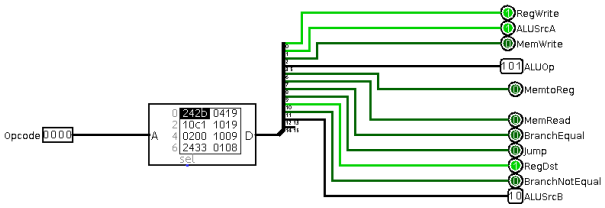

Figure 5: Control Unit of the 8 bit MIPS

The control unit in figure 5 uses the opcode(first 4 bits of the instruction) to enable the lines and generate the signals needed to execute the instruction.The control unit generates or enables 11 signals in total. These are :

- **RegWrite**(enables register writing)
- **ALUSrcA**
- **ALUSrcB**

- **MemRead**(enables memory read)
- **MemWrite**(enables memory write)
- **ALUOp**(determines the operations of the ALU)
- **MemtoReg**(sends data from memory to register)
- **BranchEqual**(if branch condition evaluates to true)
- **BranchNotEqual**(branch condition evaluates to false)
- **Jump**(unconditional jump)
- **RegDst**(determines which register will hold the result of the instruction)

For example, to execute an add instruction, the control unit receives the opcode 1110. The control unit enables the RegWrite and RegDst line,sets ALUOP to 000,ALUSrcA to 0,ALUSrcB to 00 and disables the other lines.

Approach to Implement Push and Pop Instructions

In order to implement the push and pop operations, we have used multiple MIPS instructions. To implement push operation without offset, we have broken that instruction into 2 simpler instructions : store word operation and then decrementing the stack pointer(sp) using subi. For implementing push operation with offset, we have broken it into 3 simple instructions. First to load the address with offset into a temporary register, then storing the value of that register using store word operation and finally a decrement operation to adjust the stack pointer. In case of pop operation, we have again divided it into 2 simple instructions: first increment the value of stack pointer to adjust it to point to the value to be popped and then a load word operation to load the value pointed by stack pointer into the desired register.

ICs with count as Charts

IC No./Name	Count
7432(OR)	1
7408(AND)	1
7404(NOT)	1
74HC157(2:1 MUX))	6
8 bit ALU	3
8 bit Register	9
ROM(8*20)	1
RAM(8*8)	1
ROM(4*16)	1

Simulator Used along with version number

Name : Logisim **Version** : 2.7.1

Discussion

The main objective of this assignment is to design an 8-bit MIPS and understand how it executes different types of instructions and also the execution flow of those instructions. The design consists of Instruction Memory,Data Memory,Register File. These units have been designed separately with attention and caution.After that, we connect them together to make a fully functioning 8-bit MIPS processor. Then We check all types of operations using our test data. We have faced some problems while implementing srl and sll operations because of the 8-bit ALU we used. But later figured it out. We have understood the execution flow of the instructions after this assignment.