# CSE 331 - Computer Organization

Melihcan Çilek – 1801042092

Assignment 4 Report

In this assignment we had to design a small version of MIPS processor its name MiniMIPS which contains most of the portions of real MIPS processor. In this processor, we have 8 registers each holding 32 bit numbers and 16 bit instructions which are R and I type instructions.

We had to build a Opcode-FunctionCode-Controls Table and using that table, we had to build equations that we need for control unit of the MiniMIPS processor. After that, we had to design our units and modules which are Program counter, Instruction Memory, Instruction-parser, Muxes, Control unit, Register file module, Sign-Extender modüle, ALU modüle, Data Memory Unit and Program incrementer unit.

I'm going to explain all of them in my program

1. Program counter unit (program_counter): In this unit, we increment our program counter according to input that is coming to our module. If reset input is not 1, everytime program counter writes data that is coming from pc_in input.

2. Instruction Memory (instruction_memory): We read all instructions from "instructions.mem" file and save them to mem[30] memory register. After that every index that is coming from program counter outs a instruction that is on the instruction registers.

3. Instruction Parser (instruction_seperator): We have to parse instruction coming out from instruction memory because control unit will decide which type of signals will it create so we split our 16 bit instructions to rs, rt, rd, imm, opcode, funcCode parts so that we can move these wires appropriate parts

4. Muxes (mux8x1(for 32 bit), mux2x1_3bit, mux2x1(for 32 bit) ): We use these modules for making choices between two or more choices and their select signals are coming from control unit.

5. Control unit (MIPScontrol) : Control unit making decissions according to its inputs which are opCode and func for my program. It chooses output signals according to equations created by truth table which named "MiniMIPSFunctionTable". Equations are below. I couldn't pass these equations to online because of that these are on my notebook.

RegDst = ((OpCode(3)·Opcode(2)·OpCode(1)·OpCode(0))' (nand)

RegWrite = ((Opcode(3)'·OpCode(2)·Opcode(1)'·Opcode(0)) + (Opcode(3)'·Opcode(2)·Opcode(1)·
Opcode(0)') + (Opcode(3)·Opcode(2)'·Opcode(1)'·Opcode(0)))'

ALUSrc = RegDst'

ALUOp(2) = ((OpCode(3)'·Opcode(2)'·OpCode(1)'·OpCode(0)'·func(2)'·func(1)'·func(0)
+ (OpCode(3)'·OpCode(2)'·OpCode(1)'·OpCode(0)'·Func(2)·Func(1)')
+ (OpCode(3)'·OpCode(2)'·OpCode(1)) + (Opcode(3)'·Opcode(2)·OpCode(1)'·OpCode(0)')
+ (OpCode(3)'·OpCode(2)·OpCode(1)·OpCode(0)))

ALUOp(1) = (OpCode(3)'·Opcode(2)'·OpCode(1)'·OpCode(0)'·Func(2)'·Func(1)'·Func(0))
+ (OpCode(3)'·OpCode(2)'·OpCode(1)'·OpCode(0)'·Func(2)'·Func(1)·Func(0)')
+ (OpCode(3)'·OpCode(2)'·OpCode(1)'·OpCode(0)'·Func(2)·Func(1)'·Func(0))
+ (OpCode(3)'·OpCode(2)'·OpCode(1))
+ (OpCode(3)'·OpCode(2)·OpCode(1)'·OpCode(0)) + (OpCode(3)'·OpCode(2)·OpCode(1)·OpCode(0)')

ALUOp(0) = ((OpCode(3)'·OpCode(2)'·OpCode(1)'·OpCode(0)'·Func(2)'·Func(1)·Func(0))
+ (OpCode(3)'·OpCode(2)'·OpCode(1)'·OpCode(0)'·Func(2)·Func(1)'))
+ (OpCode(3)'·OpCode(2)'·OpCode(1)·OpCode(0)) + (OpCode(3)'·OpCode(2)·OpCode(1)'·OpCode(0)')

MemWrite = ((OpCode(3)·Opcode(2)'·OpCode(1)'·OpCode(0))

MemRead = ((OpCode(3)·OpCode(2)'·OpCode(1)'·OpCode(0)')

MemToReg = MemToRead

PCSrc = ((OpCode(3)'·OpCode(2)·OpCode(1)'·OpCode(0).

---

6. Register file (mips_registers): In this structure, we holds 8 x 32bit registers for making calculations fast and hold temporary values. If write enable signal not coming, we pass data 1 and data 2 to read_data_1 and read_data_2 according to read_reg_1 and read_reg_2 index values. At the beginning of the program, we save registers random binary numbers with reading file “registers.mem” and save all the datas to “registers_out.mem” file after making any change for being able to see them.

7. Sign Extender Module (_6_to_32_bit_sign_extender): We need that module because we take immediate type 6 bit and we have to extend this to 32bit for using our offset value inside ALU.

8. ALU Module (alu32) : ALU module that we designed in our previous assignment. We can use same structure in this assignment without making any change.

9. Data Memory Unit (mips_memory) : In this unit, we store 256 x 32 bit data which we may need for next steps of the program or we might not need. If MemWrite signal coming from control unit, it writes to memory inside, but if MemRead signal is coming from control unit, it reads content that coming with address index and push out with data_out output register.

10. Program Incrementer unit (PC, adder_with_1, adder_for_branch, and_branch, branch_mux) : This unit is for taking offset coming from immidiate side and compares with zero bit of ALU and decide if opcode is branch or not, and making decissions according to that using and branch_mux in my project.

For compile that Project, in modelsim, we have to select “workspace/MiniMIPS_testbench.v” folder and after that we open for simulation and run.

After run, we will have output like below

MIPS -> opCode=0000, regDst=1, regWrite=1, ALUSrc=0, ALUOp=110, MemWrite=0, MemRead=0, MemToReg=0, PCSrc=0

Seperator -> opCode=0000, rs = 001 ,rt = 010,rd = 011,imm = 011000,func = 000

Memory -> mem[address] = 00000000000000000000000000000000, address = 00000000000000000000000000000000

PC -> next = 00000000000000000000000000000001, current = 00000000000000000000000000000000

instr memory -> mem0=0000001010011000, mem1=0000011100010000,

Registers ->

# reg[0] = 00000000000000000000000000000000,

# reg[1] = 00000000000000000000000000010001,

# reg[2] = 00000000000000000000000000100010,

# reg[3] = 00000000000000000000000000000011,

# reg[4] = 00000000000000000000000000000100,

# reg[5] = 00000000000000000000000000000101,

# reg[6] = 00000000000000000000000000000110,

# reg[7] = 00000000000000000000000000000111

These Show steps and registers. We can follow our registers with this and compare with “registers_out.mem” folder to see the changes.