CSE 331 COMPUTER ORGANIZATION ASSIGNMENT 4

GÖKÇE NUR ERER 171044079

Single Cycle MIPS

Except the modules below, all the other modules are from the assignment 3's modules. So they are not included in this report.

PS: The .mem files included in the project does not include end results. They include the register/memory contents before the program execution.

instruction_memory.v

This module represents instruction memory in single cycle MIPS datapath. It reads instructions when PC changes. It reads the instruction which is shown by PC location from instruction.mem file and outputs it.

data_memory.v

This module represents data memory in single cycle MIPS datapath. It reads data when address is given/changed to the module from the ALU and also memRead signal is 1 and outputs it from this module. It writes the input data to the address calculated by ALU if memWrite signal is 1.

alu_control.v

This module produces ALU select bits according to the ALUOp it gets from the Control Unit. The logical expressions of each ALU select bit is provided below in the schematics.

control_unit.v(changed from previous project)

This module produces all control signals needed for datapath to work. These signals are

- RegWrite
- RegDst
- Branch
- ALUSrc
- ALUOp
- MemRead
- MemToReg
- MemWrite
- Jump

Control unit decides on this control signals by looking at the instruction's opcode. The logical expressions of each control signal is provided below in the schematics.

mips32_single_cycle.v

This module represents the whole single cycle datapath. First it gets PC value from the nextPC module and uses this PC value to read the instruction at that PC value from the instruction memory. After instruction is read, necessary instruction pieces are given to the registers module and also to the control unit so the opcode of the instruction can decide the control signals. After that from the registers block the related register datas are outputted. After that, a mux right after the registers module chooses if rt content or sign extended immidient value will be used for ALU calculations. Also there is another mux which decides if the first mux result or the shamt will enter the ALU calculations.

After that, ALU result is given to the data memory just in case the instruction is a memory related one, ALU result represents address at that point. At sw instruction the rt content is written in to the memory so rt content is directly connected to the write data entrance of the data memory block. Data memory outputs a data out if the instruction is a lw so that the value can be written to the register. Right after the data memory block there is a mux which decides if the ALU result will be returned to the registers or the memory content. This mux takes memToReg signal to decide so.

This module only gets clock value since every other module is inside of it.

mips_registers.v(changed from previous project)

The difference between the old assignment and this one is register block reads registers when reading register addresses change.

nextPC.v

This module calculates the next PC value with given control signals like Branch, Jump and gets instruction to calculate certain addresses and also uses ALUResult which is specificly zero if there is equality. If branch and ALUResult in an and gate results 1 branch equal address is given to the PC, if jump is 1 then jump address is given to PC. The initial value of PC is 0 since programs start from the first instruction.

If there is no branch or jump the next PC result proceeds as PC+1

If there is a branch the next PC result proceeds as PC+1+BranchAddress

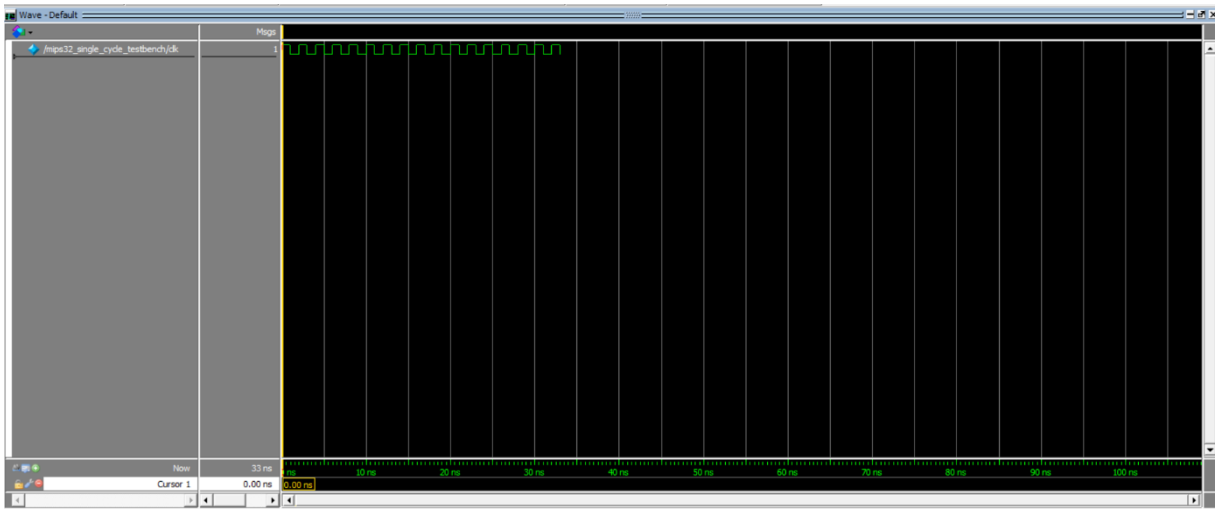
If there is a jump the next PC result proceeds as {PC+1[31:28],2'b0,instruction[25:0]} , instruction [25:0] which is the jump address from the instruction.

sign_extend_imm.v

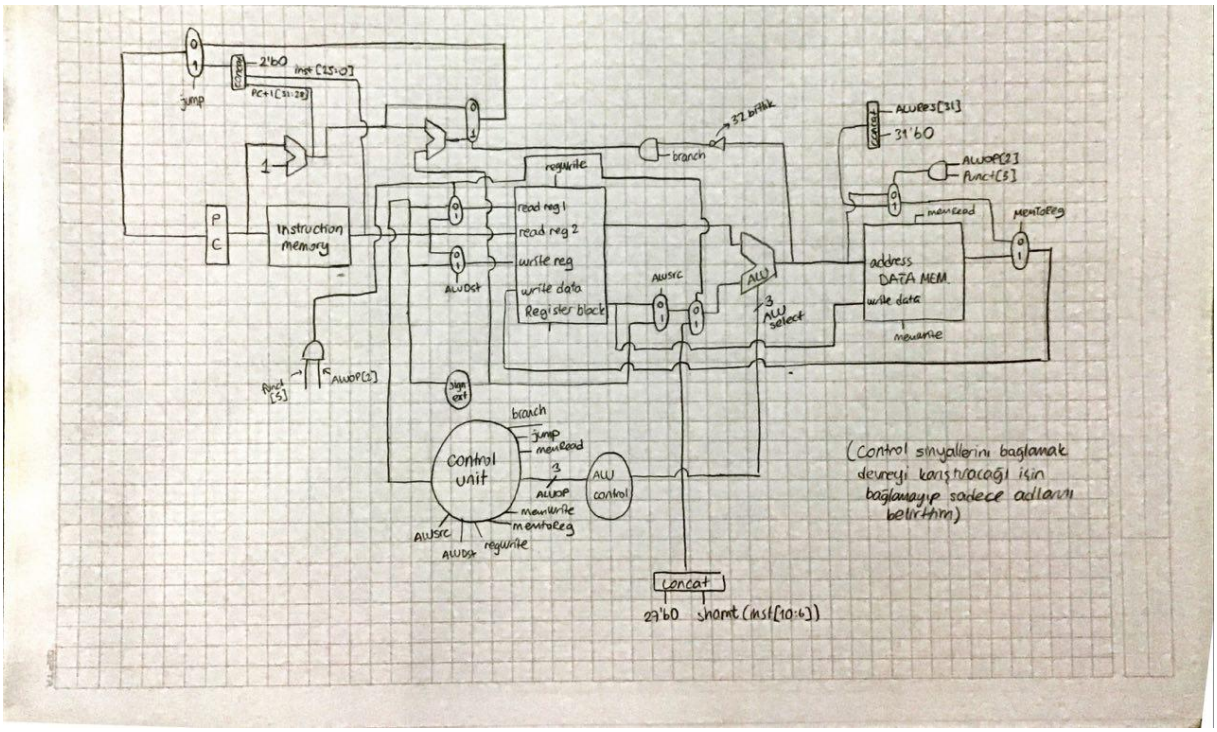
This module extends a 16 bit long binary number to 32 bit. This extended value is used in I type instruction calculations.

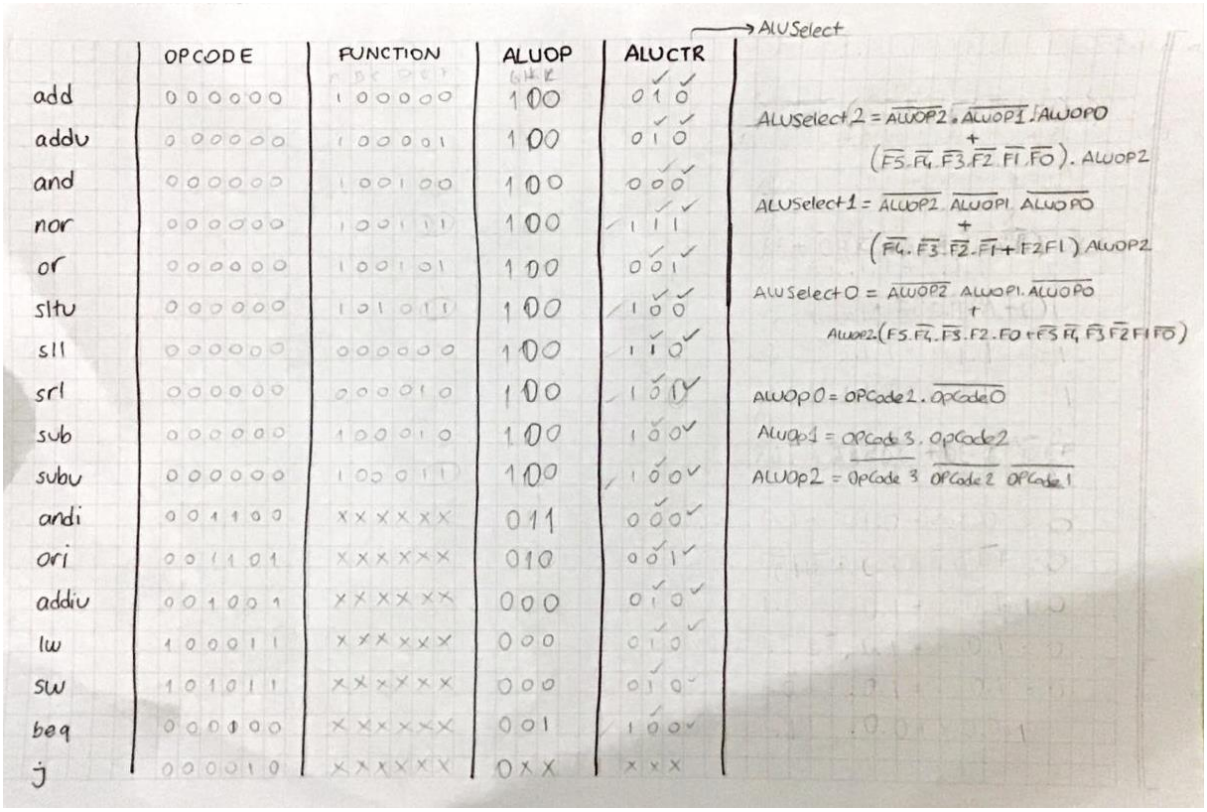
Simulation Results

```
Transcript
sim:/mips32_single_cycle_testbench/result
VSIM 5> step -current
# For all the results please check data.mem and registers.mem files also all the instructions are in the instr
uction.mem file
# j jumpadd=00000000000000000000000000000010
# clk=1
# clk=0
# lw rs=0 rt=1 imm=0000000000000001, this instruction will write 111..11 on rt register
# clk=1
# clk=0
# beq rs=4 rt=5 branch=0000000000000100
# clk=1
# clk=0
# add rs=3 rt=4 rd=5 this instruction overrides 5th register, before this execution 4th and 5th registers were
equal
# clk=1
# clk=0
# addu rs=3 rt=4 rd=6
# clk=1
# clk=0
# and rs=7 rt=8 rd=9
# clk=1
# clk=0
# nor rs=10 rt=11 rd=12
# clk=1
# clk=0
# or rs=13 rt=14 rd=15
# clk=1
# clk=0
# sltu rs=16 rt=17 rd=18
# clk=1
# clk=0
# sll rs=19 rt=20 rd=21 shamt=2
# clk=1
# clk=0
# srl rs=22 rt=23 rd=24 shamt=2
# clk=1
# clk=0
# sub rs=25 rt=26 rd=27
# clk=1
# clk=0
# subu rs=25 rt=26 rd=28
# clk=1
# clk=0
# andi rs=1 rt=29 imm=00..00
# clk=1
# clk=0
# ori  rs=3 rt=30 imm=11..11
# clk=1
# clk=0
# addiu rs=5 rt=31 imm=00..1
# clk=1
# clk=0
# sw rs=0 rt=1 imm=00...01
# clk=1
# clk=0
```



Schematics





	OPCODE	FUNCTION	ALUOP	ALUCTR
add	000000	100000	100	010
addu	000000	100001	100	010
and	000000	100100	100	000
nor	000000	100111	100	111
or	000000	100101	100	001
sltu	000000	101011	100	100
sll	000000	000000	100	110
srl	000000	000010	100	101
sub	000000	100010	100	100
subu	000000	100011	100	100
andi	001100	XXXXXX	011	000
ori	001101	XXXXXX	010	001
addiu	001001	XXXXXX	000	010
lw	100011	XXXXXX	000	010
sw	101011	XXXXXX	000	010
beq	000100	XXXXXX	001	100
j	000010	XXXXXX	0XX	XXX

ALUSelect

ALUSelect2 = $\overline{ALUOP2} \cdot \overline{ALUOP1} \cdot ALUOP0 + (\overline{F5}.\overline{F4}.\overline{F3}.\overline{F2}.F1.\overline{F0}).ALUOP2$

ALUSelect1 = $\overline{ALUOP2}.\overline{ALUOP1}.\overline{ALUOP0} + (\overline{F4}.\overline{F3}.\overline{F2}.\overline{F1} + F2F1).ALUOP2$

ALUSelect0 = $\overline{ALUOP2}.ALUOP1.\overline{ALUOP0} + ALUOP2.(F5.\overline{F4}.\overline{F3}.F2.F0 + \overline{F5}F4\overline{F3}\overline{F2}F1\overline{F0})$

ALUOP0 = OPCode2 . $\overline{OPCode0}$

ALUOp1 = OPCode3 . OpCode2

ALUOp2 = OpCode3 $\overline{OPCode2}$ $\overline{OPCode1}$