Computer Organization 2018

Homework 3 MIPS CPU

Deadline:2018/5/28 23:55

Overview

The goal of this homework is to help you understand how a single-cycle MIPS work and how to use Verilog hardware description language (Verilog HDL) to model electronic systems. In this homework, you need to implement modules of CPU and make your codes be able to execute all MIPS instructions. You need to follow the instruction table in this homework and satisfy all the homework requirements. In addition, you need to verify your CPU by using Modelsim.

General rules for deliverables

- Please use **Modelsim** as the simulation platform. No team work.
- Files TestBench.v, Add.v, PC.v, CPU.v, IM.v, DM.v, Reg.v, ALU.v, ALU_ctrl.v, MUX_2_to_1.v, sign_extend.v are provided.
- You need to submit one report to answer questions and also the source codes.
- When submitting your homework, compress all files into a single **zip** file, and upload the compressed file to Moodle.
  - Please follow the file hierarchy shown in Figure 1.

F740XXXXX (your id)(folder)

SRC (folder) * Store your source code

F740XXXXX_Report.docx (Project Report. The report template is already included. Follow the template to complete the report.)

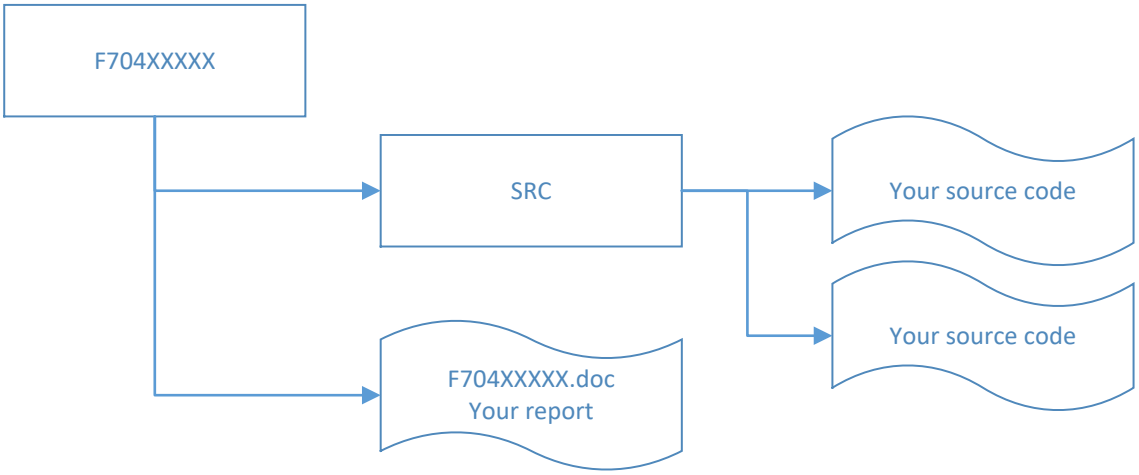


Figure 1. File hierarchy for homework submission

Instruction format:

Instruction	op [25:20]	rs [19:17]	rt [16:14]	constant or address [13:0]
lw	6'b100011	SSS	TTT	YYYYYYYYYYYYYY
sw	6'b101011	SSS	TTT	YYYYYYYYYYYYYY

Instruction	op [25:20]	rs [19:17]	rt [16:14]	constant or address [13:0]
beq	6'b000100	SSS	TTT	YYYYYYYYYYYYYY
addi	6'b001000	SSS	TTT	YYYYYYYYYYYYYY

Instruction	op [25:20]	Target[19:0]
j	6'b000010	XXXXXXXXXXXXXXXXXXXX

Homework Description

- **Module**

a. TestBench module

"TestBench" is not a part of CPU, it is a file that controls all the program and verify the correctness of our CPU. The main features are: send periodical signal CLK to CPU, set the initial value of Reg and DM, print the value of Reg and DM at each cycle, finish the program.
The initial value are Reg[i]=0, DM[i]=0, (i=0~7).
※You do not need to modify this module

b. CPU module

Here are new wires:
PC_In represents Program Counter input
PC_Out represents Program Counter Output
Jump represents the instruction is j or not
Branch represents the instruction is beq or not
Jump_Address represents the PC value of the target.
PC_Count_Add_Src1 represents the input of PC_Count_Adder
PC_Count_Add_Src2 represents the input of PC_Count_Adder
PC_Count_Add_Result represents the output of PC_Count_Adder
Branch_Add_Src1 represents the input of Branch_Adder
Branch_Add_Src2 represents the input of Branch_Adder
Branch_Add_Result represents the output of Branch_Adder
Branch_Select represents the control signal of MUX_Branch
※You need to modify this module:
Implement the datapath of Branch and Jump.

c. PC module

"PC" is the abbreviation of "Program Counter". It is responsible for controlling next instruction ready to be executed. PC_In will assign to PC_Out in the end of every cycle.

※You do not need to modify this module

d. IM module

"IM" is the abbreviation of "Instruction Memory". This module saves all the instructions and send the instructions to CPU according to PC.

※You do not need to modify this module

e. Reg module

"Reg" is Register module. It will read or write data according to input signals. The name of wires please refer to CPU module.

※You do not need to modify this module

f. DM module

Same as Reg, but the target here is Data Memory.

Real memory is continuous spaces using byte as unit, but in order to easily implement this lab, we use what we use in Reg to simulate DM. That is using 16 bits as a unit to represent memory and use "grid" (DM[0] is a grid, DM[1] is a grid…) as continuous spaces.

※You do not need to modify this module

g. Decoder module

Decoder is the module decodes signal according to OP code that passed in. Signals are send to each module for them to know what they should do.

※You need to modify this module:

Add the control signal of Branch and Jump.

Redesign the control signal of ALU_Src、Reg_Write、ALU_OP.

opcode	ALUOp
lw	00
sw	00
beq	01
R-type	10

h. ALU_ctrl module

Input ALU_OP and Funct and use these two input to ouput ALU_CTRL.

※You need to modify this module:

Set ALU_CTRL according to ALU_OP and funct.

opcode	ALUOp	Operation	funct	ALU function	ALU control
lw	00	load word	XXXXXX	add	0010
sw	00	store word	XXXXXX	add	0010
beq	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		set-on-less-than	101010	set-on-less-than	0111

i. MUX_2_to_1 module

Choose the output according to select signal.

※You do not need to modify this module

j. sign_extend module

Extend the immediate_in(14 bits) to 16bits but don't change its sign.

※You do not need to modify this module

k. ALU module

Perform the operation according to ALU_CTRL from ALU_ctrl module.

※You need to modify this module:

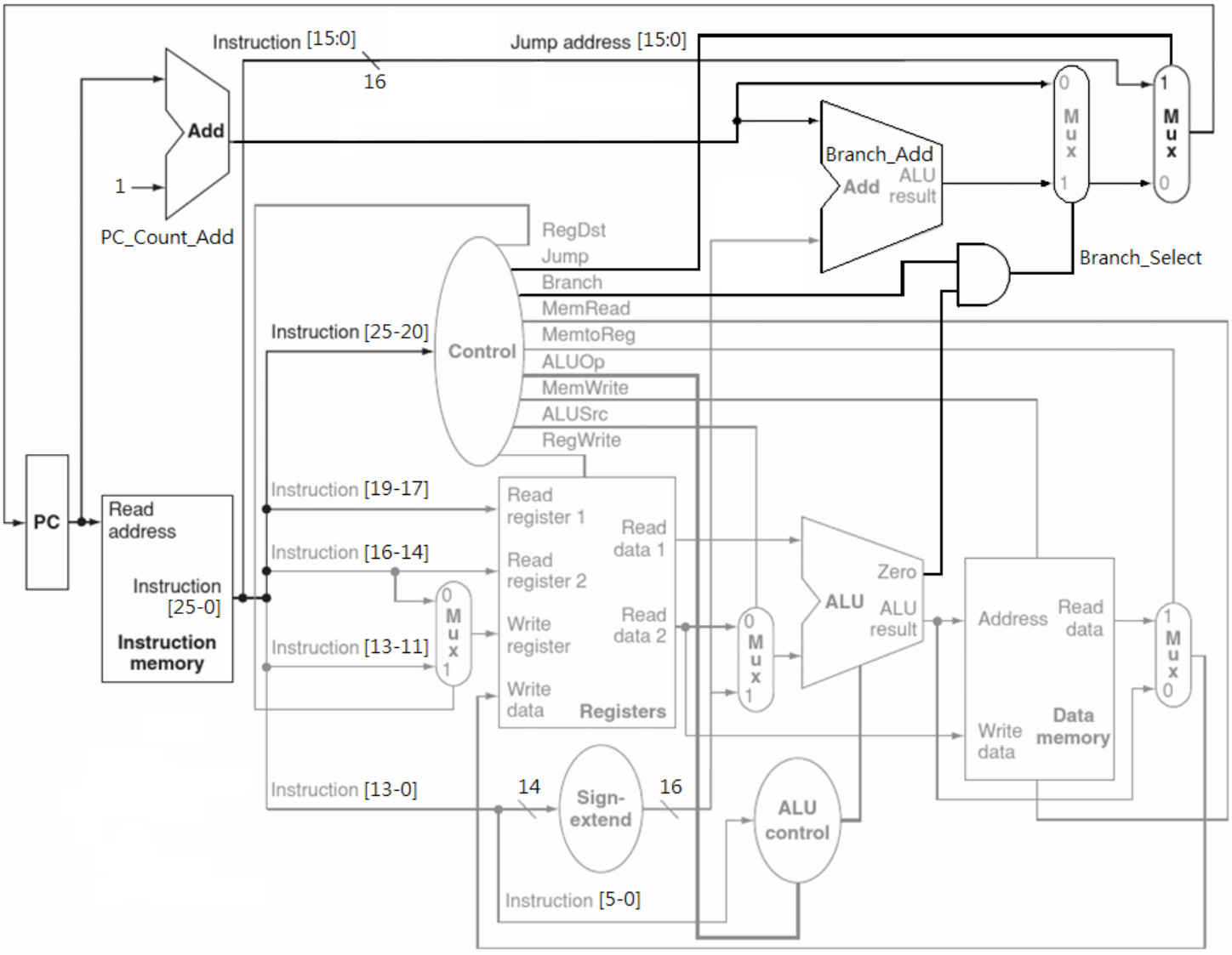
Add control signal 'Zero'. If ALU_Result = 0 then Zero = 1 else Zero = 0.

l. Add module

A normal Adder.

※You do not need to modify this module

reference graph



test instruction(test1.txt)

DM[0]~DM[7] are initialized to 0, R[0]~R[7] are initialized to 0.

```
0    lw     R[1], 0(R[0])
1    beq    R[0], R[1], 3
2    addi   R[0], R[0], 1
3    add    R[2], R[0], R[2]
4    j      1
5    sw     R[2], 1(R[1])
```

initial values:

Reg[0]	Reg[1]	Reg[2]	Reg[3]	Reg[4]	Reg[5]	Reg[6]	Reg[7]
0	0	0	0	0	0	0	0

DM[0]	DM[1]	DM[2]	DM[3]	DM[4]	DM[5]	DM[6]	DM[7]
5	0	0	0	0	0	0	0

Simulation result:

Reg[0]	Reg[1]	Reg[2]	Reg[3]	Reg[4]	Reg[5]	Reg[6]	Reg[7]
5	**5**	**15**	0	0	0	0	0

DM[0]	DM[1]	DM[2]	DM[3]	DM[4]	DM[5]	DM[6]	DM[7]
5	0	0	0	0	0	**15**	0

Homework Requirements

1. Complete the Single cycle CPU that can execute add, sub, and slt instructions from the *MIPS ISA* section.
2. Verify your CPU with the benchmark and take a snapshot (e.g. Figure.)

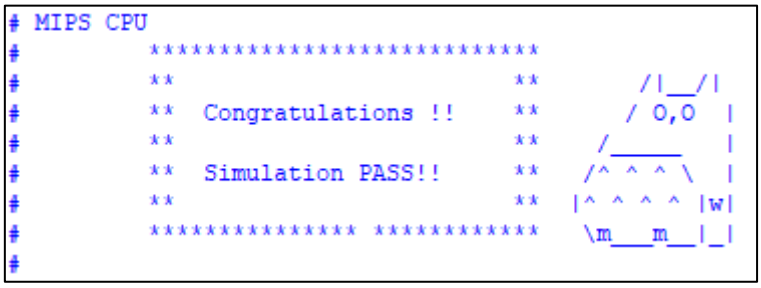


Figure. **S**napshot of correct simulation

a. Using **waveform** to verify the execute results.

b. Please annotate the waveform

3. Finish the Project Report.
    a. Complete the project report. The report template is provided.

> **Important**
>
> When you upload your file, please make sure you have satisfied all the homework requirements, including the **File hierarchy**, **Requirement file** and **Report format**.
>
> If you have any questions, please contact us.