Computer Organization 2017

HOMEWORK II

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 a single cycle MIPS CPU that can execute all the instructions shown in the MIPS ISA section. 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. TAs will provide test fixtures that will run a MIPS program for the CPU. We will use hidden test fixtures to test your CPU. Please implement all the modules and use the test fixtures provided by TAs to verify the CPU. You also need to take a snapshot and explain it, including the wires, signals, etc. in your report.

General rules for deliverables

- You need to complete this homework INDIVIDUALLY. You can discuss the homework with other students, but you need to do the homework by yourself. If you copy your codes from someone else, you will not get any scores.
- 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.)

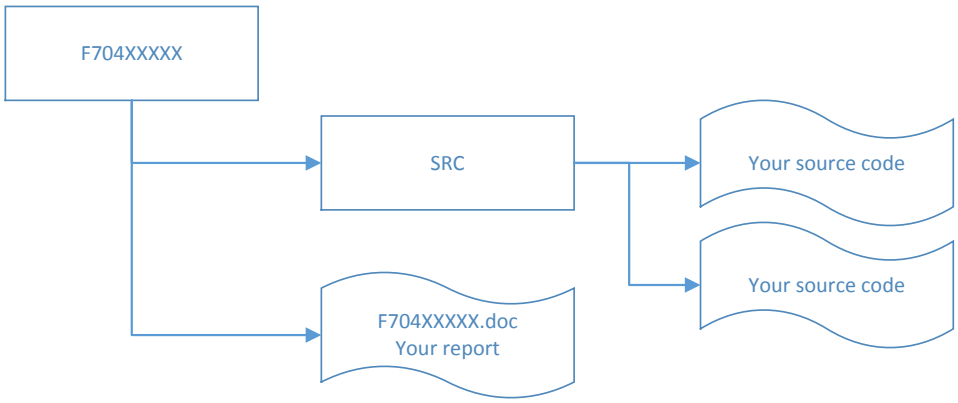

Figure 1. File hierarchy for homework submission

- **Important!** DO NOT submit your homework in the last minute. Late submission is not accepted.
- You should finish all the requirements (shown below) in this homework and Project report.

Homework Description

1. The single-cycle CPU use one cycle to execute instruction. There are five main components in CPU: controller, regfile, arithmetic logic unit (ALU), program counter(PC), and jump controller.
2. The controller control most of the multiplexers, DM write enable, ALU, Jump Controller, and regfile. The file Controller.v implements the controller.
3. The regfile is used to store data between memory and the functional units. The file Regfile.v implements the regfile.
4. The arithmetic logic unit do arithmetic and bitwise operations. The file ALU.v implements the arithmetic logic unit.
5. The program counter is stored in the PC module. The file PC.v implements the program counter. It is triggered by positive clock edges.
6. The jump controller select a memory address and send to PC. The file Jump_Ctrl.v implements the jump controller.
7. The instruction memory is used to store instructions and it is implemented in IM.v files. The data memory is used to store data and it is implemented in DM.v file.
8. The MIPS ISA is shown in the following. Figure 2 shows the R-type instructions in the MIPS ISA. Figure 3 shows the I-type instructions in the MIPS ISA. Figure 4 shows the J-type instructions in the MIPS ISA.

R-type Instruction Machine Code Format

opcode	Mnemonics	SRC1	SRC2	DST	funct	Description
000000	nop	00000	00000	00000	000000	No operation
000000	add	$Rs	$Rt	$Rd	100000	Rd = Rs + Rt
000000	sub	$Rs	$Rt	$Rd	100010	Rd = Rs – Rt
000000	and	$Rs	$Rt	$Rd	100100	Rd = Rs & Rt
000000	or	$Rs	$Rt	$Rd	100101	Rd = Rs | Rt
000000	xor	$Rs	$Rt	$Rd	100110	Rd = Rs ^ Rt
000000	nor	$Rs	$Rt	$Rd	100111	Rd = ~(Rs | Rt)
000000	slt	$Rs	$Rt	$Rd	101010	Rd = (Rs < Rt)?1:0
000000	sll		$Rt	$Rd	000000	Rd = Rt << shamt
000000	srl		$Rt	$Rd	000010	Rd = Rt >> shamt
000000	jr	$Rs			001000	PC=Rs
000000	jalr	$Rs			001001	R[31] = PC + 8 ; PC=Rs

Figure 2. R-type MIPS instructions

I-type Instruction Machine Code Format

opcode	Mnemonics	SRC1	DST	SRC2	Description
001000	addi	$Rs	$Rt	imm	Rt = Rs + imm
001100	andi	$Rs	$Rt	imm	Rt = Rs & imm
001010	slti	$Rs	$Rt	imm	Rt = (Rs < imm) ? 1 : 0
000100	beq	$Rs	$Rt	imm	If(Rs == Rt) PC=PC+4+imm
000101	bne	$Rs	$Rt	imm	If(Rs != Rt) PC=PC+4+imm
100011	lw	$Rs	$Rt	imm	Rt = Mem[Rs + imm]
100001	lh	$Rs	$Rt	imm	data = Mem[Rs + imm] Rt = data[15:0] <- Sign-extend 16bits
101011	sw	$Rs	$Rt	imm	Mem[Rs + imm] = Rt
101001	sh	$Rs	$Rt	imm	data <- Rt[15:0] Sign-extend 16bits Mem[Rs + imm] = Rt

Figure 3. **I**-type MIPS instructions

J-type Instruction Machine Code Format

opcode	Mnemonics	Address	Description
000010	j	jumpAddr	PC = jumpAddr
000011	jal	jumpAddr	R[31] = PC + 8 ; PC = jumpAddr

Figure 4. **J**-type MIPS instructions

9. Figure 5 shows the datapath of the single-cycle CPU. This single-cycle CPU is similar to the single-cycle in the textbook. However, to simply the CPU design, only smaller instruction and data memory are used and only 18-bit addresses are needed to obtain data in the memory.

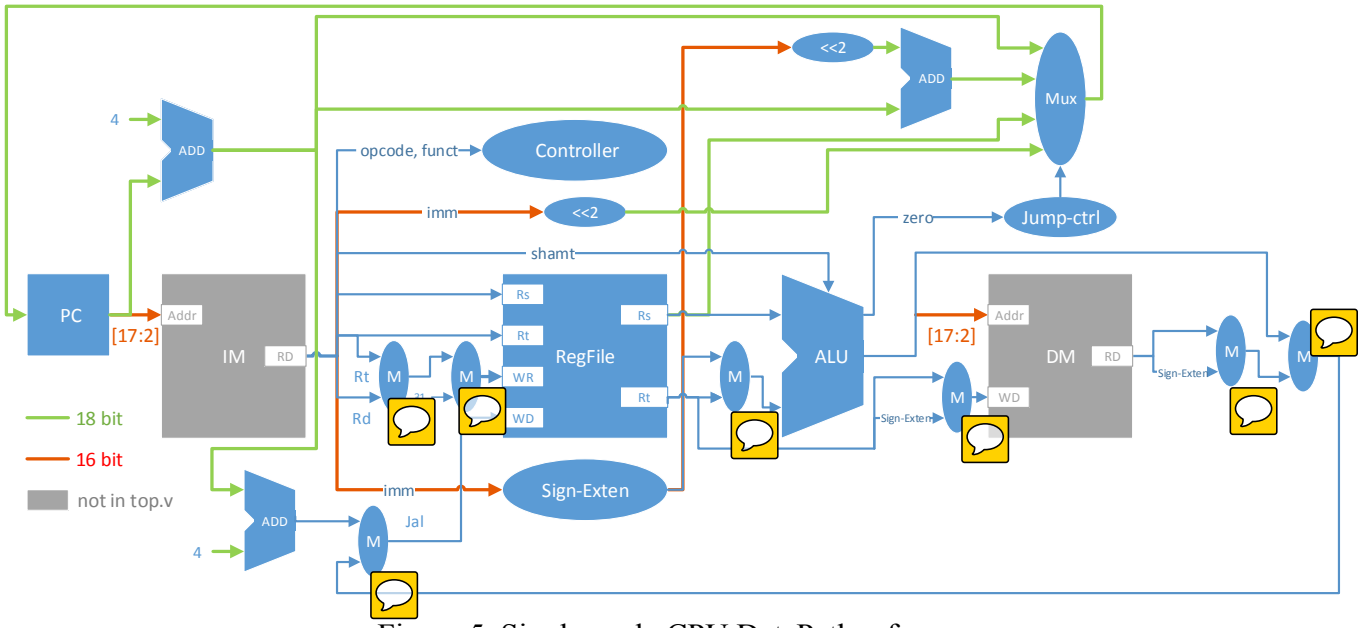

Figure 5. Single cycle CPU DataPath reference

Homework Requirements

1. Complete the Single cycle CPU that can execute all the instructions from the *MIPS ISA* section.
   a. top.v (your single-cycle cpu)
   b. Controller.v
   c. Regfile.v
   d. ALU.v
   e. PC.v
   f. Jump_Ctrl.v
2. Verify your CPU with the benchmark and take a snapshot (e.g. Figure 6)

```
VSIM 16> run -all
# [ testfuxture1.v ] Rtype test START !!
# ================================================================================
#
#  \(^o^)/  The Rtype result of DM_data is PASS!!!
#
# ================================================================================
# [ testfuxture1.v ] Itype test START !!
# ================================================================================
#
#  \(^o^)/  The Itype result of DM_data is PASS!!!
#
# ================================================================================
# [ testfuxture1.v ] Jtype test START !!
# ================================================================================
#
#  \(^o^)/  The Jtype result of DM_data is PASS!!!
#
# ================================================================================
#
#
#         Single Cycle CPU
#         ****************************
#         **                        **       /|__/|
#         **  Congratulations !!    **      / O,O  |
#         **                        **    /_____   |
#         **  Simulation PASS!!     **   /^ ^ ^ \  |
#         **                        **  |^ ^ ^ ^ |w|
#         ************** ************   \m___m__|_|
#          student ID :
#
#
```

Figure 6. Snapshot of correct simulation

3. Take snapshot
    a. Using waveform to verify the execute results.
    b. Please annotate the waveform (as shown in Figure 7)

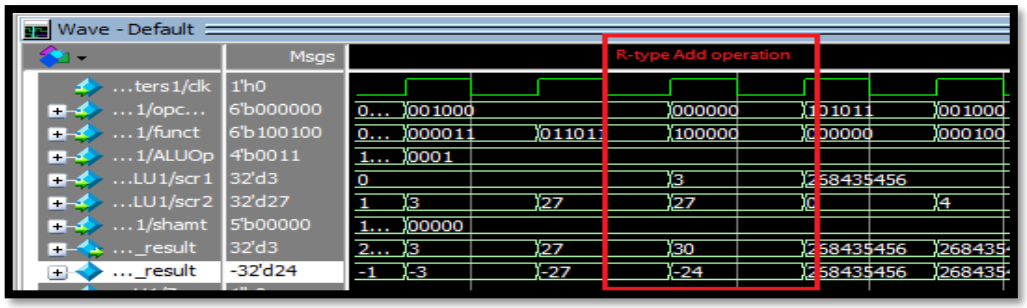

Figure 7. Waveform snapshots

    c. At least list the following modules in the waveform snapshot and explain how it works. Of course, listing more modules is better.
        i. Controller, ALU, Regfile, Jump_Ctrl
4. Finish the Project Report.
    a. Complete the project report. The report template is provided.
    b. If your CPU datapath is different from Figure 5, submit the Verilog files of your CPU design and explain how it works.

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.