Computer Organization Lab 3

教授:蔡文錦
TAs:林浩君、薛乃仁、許承壹

Table of Contents

Objectives

In this lab, we are going to implement a simple single cycle CPU with memory unit, which can run R-type, I-type and jump instructions.

Demands

There are plenty of modules that you need to write in this lab:

“Simple_Single_CPU.v”, “Adder.v”, “ALU.v”, “ALU_Ctrl.v”, “Decoder.v”, “Instr_Memory.v”, “Mux2to1.v”, “Mux3to1.v”, “Program_Counter.v”, “Reg_File.v”, “Shifter.v”, “Sign_Extend.v”, “Zero_Filled.v”, “Data_Memory.v”, “Testbench.v”

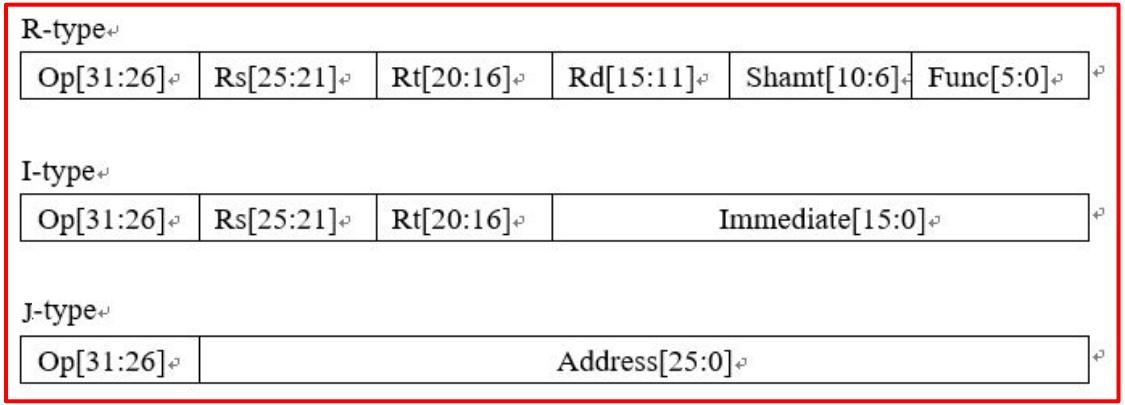
All these files are provided, please make use of them and **don’t create additional .v files**.

Requirement Description

We are going to implement these instructions, and we recommend you to write then in such sequence:

- R-type: add, sub, and, or, nor, slt, sll, srl, sllv, srlv, jr
- I-type: addi, lw, sw, beq, bne, blt, bnez, bgez
- J-type: jump, jal

Instruction Format:



R-type Instruction set and ADDI (9% * 5)

Instruction	Example	Meaning	Op Field[31:26]	Shamt	Function Field
add	add rd, rs, rt	Reg[rd]=Reg[rs]+Reg[rt]	6'b000000	x	6'b100011
sub	sub rd, rs, rt	Reg[rd]=Reg[rs]-Reg[rt]	6'b000000	x	6'b010011
and	and rd, rs, rt	Reg[rd]=Reg[rs]&Reg[rt]	6'b000000	x	6'b011111
or	or rd, rs, rt	Reg[rd]=Reg[rs]|Reg[rt]	6'b000000	x	6'b101111
nor	nor rd, rs, rt	Reg[rd]=~(Reg[rs]|Reg[rt])	6'b000000	x	6'b010000
slt	slt rd, rs, rt	If (Reg[rs]<Reg[rt]) Reg[rd]=1 else Reg[rd]=0	6'b000000	x	6'b010100
sll	sll rd, rt	Reg[rd]= (unsigned) Reg[rt]<<(unsigned) amt	6'b000000	amt	6'b010010
srl	srl rd, rt	Reg[rd]= (unsigned) Reg[rt]>>(unsigned) amt	6'b000000	amt	6'b100010
addi	addi rt,rs,imm	Reg[rt]=Reg[rs] + imm	6'b010011	x	x

Advanced Set: Part 1 (10%)

- sllv (Shift left logical variable) and srlv (Shift right logical variable)

Instruction	Example	Meaning	Op Field[31:26]	Shamt	Function Field
sllv	sllv rd, rt, rs	Reg[rd]= (unsigned) Reg[rt]<<(unsigned) Reg[rs]	6'b000000	x	6'b011000
slrv	srlv rd, rt, rs	Reg[rd]= (unsigned) Reg[rt]>>(unsigned) Reg[rs]	6'b000000	x	6'b101000

I-type Instruction and Jump (25%)

Instruction	Example	Meaning	Op Field[31:26]
lw	lw rt, imm(rs)	Reg[rt] <= Mem[Reg[rs] + imm]	6'b011000
sw	sw rt, imm(rs)	Mem[Reg[rs] + imm] <= Reg[rt]	6'b101000
beq	beq rs, rt, imm	if(Reg[rs]==Reg[rt]) then PC = PC+4+(imm<<2)	6'b011001
bne	bne rs, rt, imm	if(Reg[rs]!=Reg[rt]) then PC = PC+4+(imm<<2)	6'b011010
jump	jump address	PC ={PC[31:28],address[25:0]<<2}	6'b001100

Advanced Set: Part 2 (10%)

- jal (Jump and link)
  - In MIPS, 31th register is used to save return address for function call Reg[31] save PC+4 and perform jump
- jr (Jump to the address in the register rs)

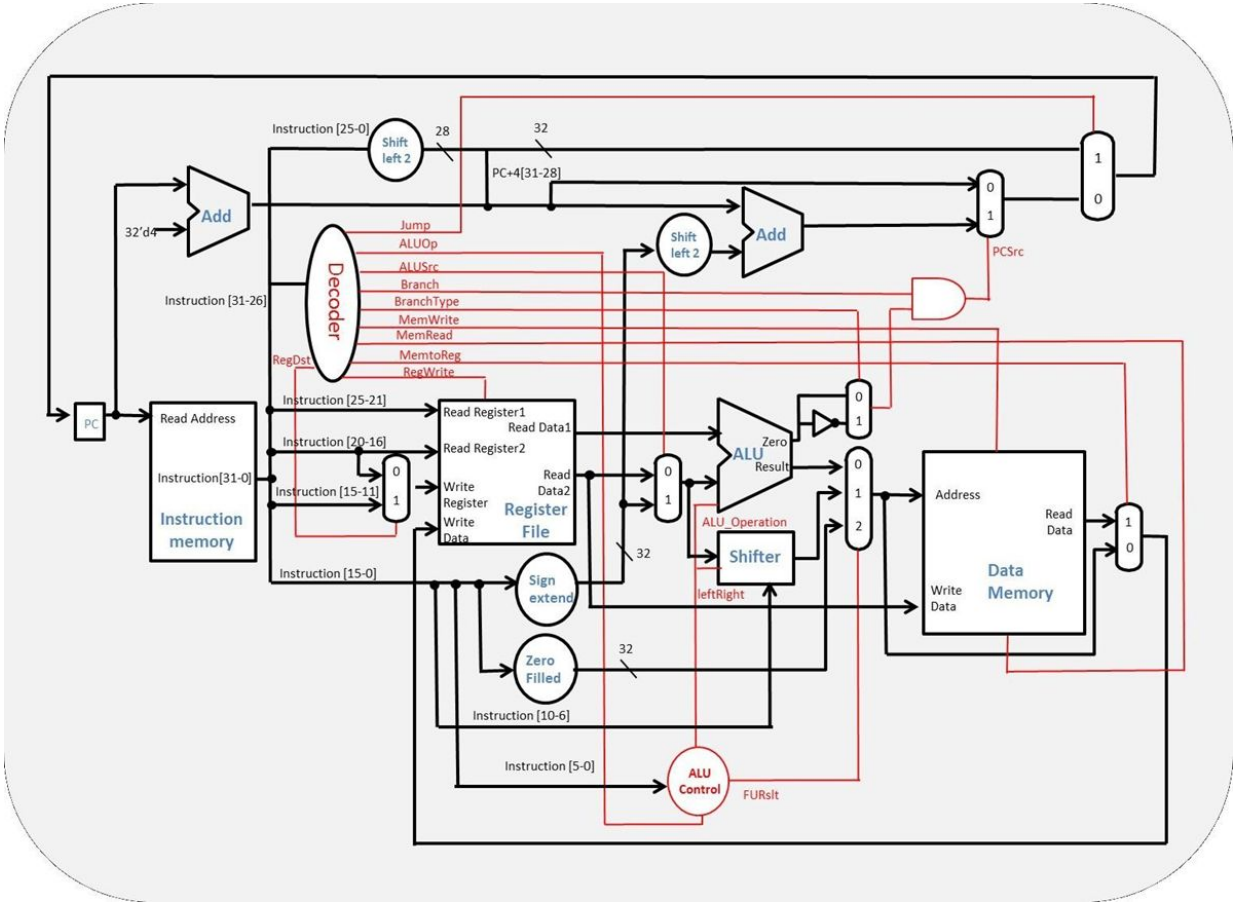
Instruction	Example	Meaning	Op Field[31:26]	Function Field[5:0]
jal	jal address	Reg[31]=PC+4 PC={PC[31:28],address[25:0]<<2}	6'b001111	x
jr	jr rs	PC=Reg[rs]	6'b000000	6'b000001

Advanced Set: Part 3 (10%)

- blt(branch on less than): if(Reg[rs]<Reg[rt]) then branch
- bnez(branch non equal zero): if(Reg[rs]!=0) then branch
- bgez(branch greater equal zero): if(Reg[rs]>=0) then branch

Instruction	Example	Meaning	Op Field[31:26]
blt	blt rs, rt, imm	if(Reg[rs]<Reg[rt]) then PC = PC+4+(imm<<2)	6'b011100
bnez	bnez rs, 0, imm (set rt = 0, note: Reg[0] always equal zero)	if(Reg[rs]!=0) then PC = PC+4+(imm<<2)	6'b011101
bgez	bgez rs, 0, imm (set rt = 0, note: Reg[0] always equal zero)	if(Reg[rs]>=0) then PC = PC+4+(imm<<2)	6'b011110

Architecture



- You should follow the architecture to implement your cpu (but with some tweak).

Report (10%)

There’s no restriction about the report format, but please answer the following questions:

1. Which part of the module in simple-single-CPU is redundant? Can you design a new instruction to use this module? (5%)
2. Which instruction is redundant and why? (5%)

Testing Data (for student)

- testcases/test_[1-3].txt: basic instructions using R-type and addi.
- testcases/test_4.txt: basic instructions using I-type and jump.
- **No testing for instruction in advance sets! Try to write your own testing data.**

Submission

- Compress all the ***.v files and HW3_{studentID}.pdf** into one zip file, and name your zip file as **HW3_{studentID}.zip (e.g. HW3_0811510.zip)**
- **Wrong format will have 10% penalty.**
- **Any assignment work by fraud will get a zero point.**
- **No late submission (deadline: 8/8 23:59)!**

```
❯ zipinfo -1 HW3_0987654321.zip
Adder.v
ALU_Ctrl.v
ALU.v
Data_Memory.v
Decoder.v
Instr_Memory.v
Mux2to1.v
Mux3to1.v
Program_Counter.v
Reg_File.v
Shifter.v
Sign_Extend.v
Simple_Single_CPU.v
Testbench.v
Zero_Filled.v
HW3_0987654321.pdf
```