Computer Organization Project 3

Part I ： Implement a 5-stage pipelined processor with R-format instructions.

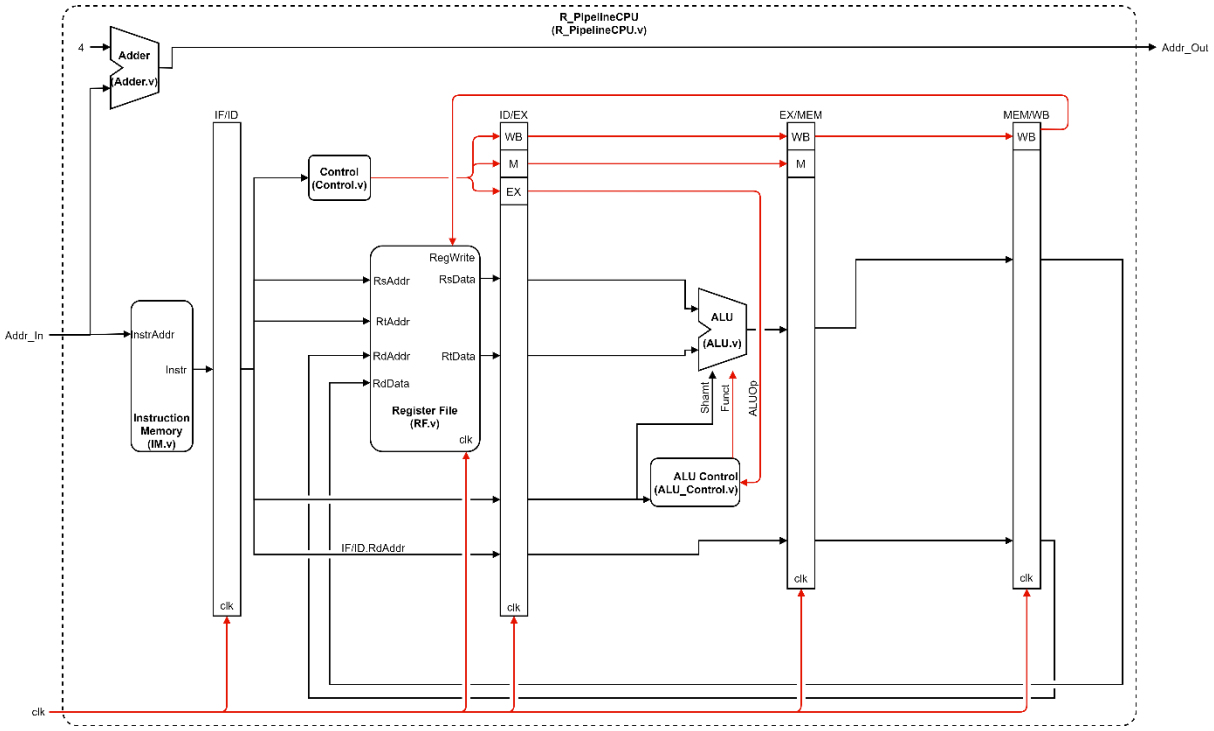


Figure 1：Architecture of a 5-stage pipelined processor with R-format instructions

Implements a 32-bits processor and supports the following R-format instructions.

Instruction	Example	Meaning	OpCode	funct
Add unsigned	Addu $Rd, $Rs, $Rt	$Rd = $Rs + $Rt	000000	001001
Sub unsigned	Subu $Rd, $Rs, $Rt	$Rd = $Rs – $Rt	000000	001010
Or	Or $Rd, $Rs, $Rt	$Rd = $Rs | $Rt	000000	010010
Shift right logical	Srl $Rd, $Rs, shamt	$Rd = $Rs >> shamt	000000	100010

Note: Please refer to HW1 for the method of converting text instructions into 32-bits execution codes.

Note: When executing the R-format instruction, ALUOp is set as "10". Then, the ALU Control recognizes the "funct_ctrl" and converts the corresponding ALU function code "funct".

I/O Interface

```
module R_PipelineCPU (
        output  wire    [31:0]  Addr_Out,
        input   wire    [31:0]  Addr_In,
        input   wire            clk
);
```

Part II ： Implement a 5-stage pipelined processor with I-format instructions.

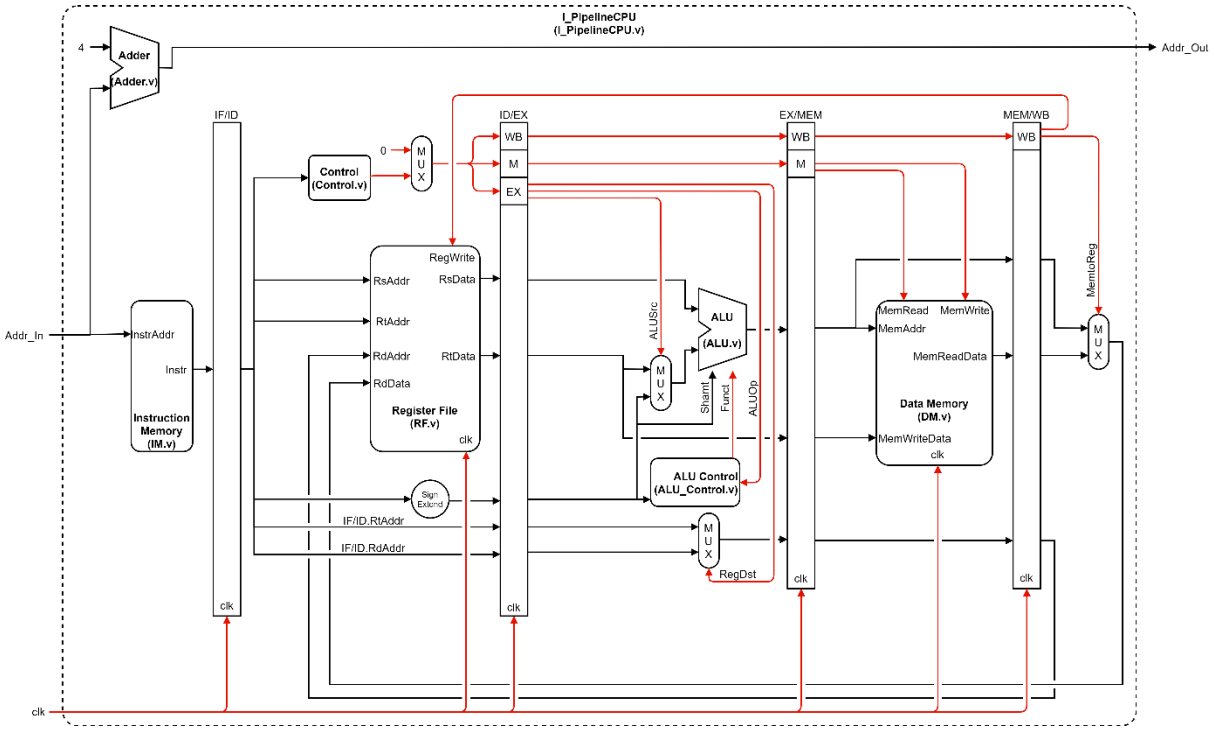


Figure 2：Architecture of a 5-stage pipelined processor with R-format and I-format instructions

Implement a 32-bits processor that supports the R-format of the previous part and supports the following I-format instructions.

Instruction	Example	Meaning	OpCode
Add imm. unsigned	Addiu $Rt, $Rs, Imm.	$Rt = $Rs + Imm.	001100
Sub imm. unsigned	Subiu $Rt, $Rs, Imm.	$Rt = $Rs – Imm.	001101
Store word	Sw $Rt, Imm. ($Rs)	Mem.[$Rs+Imm.] =$Rt	010000
Load word	Lw $Rt, Imm. ($Rs)	$Rt =Mem.[$Rs+Imm.]	010001

Note: When executing the I-format instruction, ALUOp is set as "00" or "01". Then, ALU Control ignores the "funct_ctrl", and triggers the ALU to perform "addition" or "subtraction" and outputs the corresponding "funct".

I/O Interface

```
module I_PipelineCPU (
        output  wire    [31:0]  Addr_Out,
        input   wire    [31:0]  Addr_In,
        input   wire            clk
);
```

Part III ：Implement a 5-stage pipelined processor with forwarding and hazard detection.

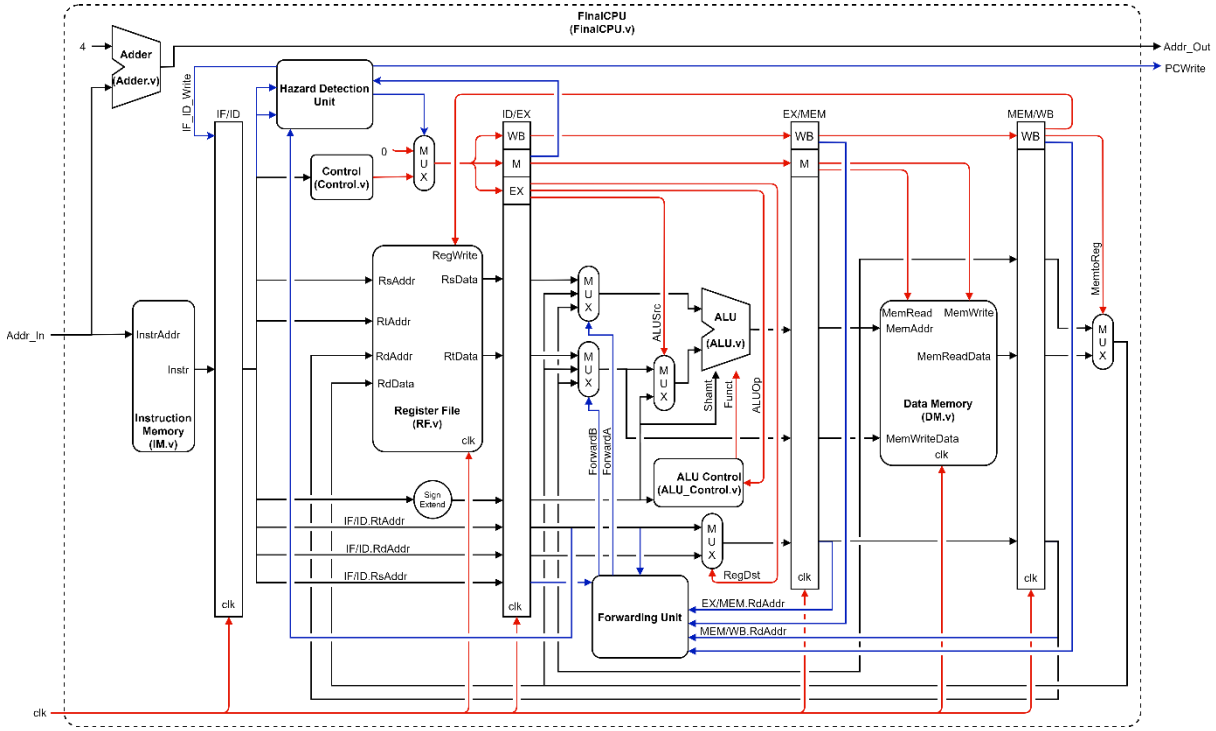


Figure 3：Architecture of a 5-stage pipelined processor supporting forwarding and hazard detection

Implement a 32-bit processor to support R-format and I-format of the first two parts, and support forwarding and hazard detection.

I/O Interface

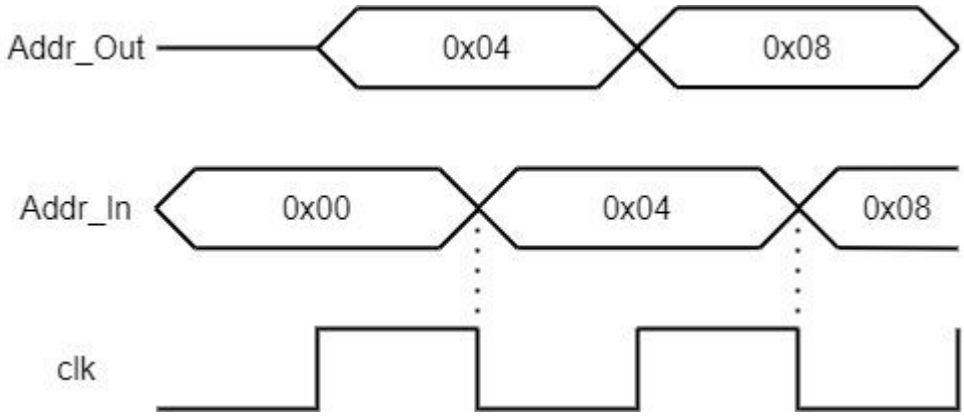
```
module FinalCPU (
        output  wire            PCWrite,
        output  wire    [31:0]  Addr_Out,
        input   wire    [31:0]  Addr_In,
        input   wire            clk
);
```

Testbench Description

a. **Initialize**
Execute Testbench ("tb_R_ PipelineCPU.v", "tb_I_ PipelineCPU.v", "tb_ FinalCPU.v") to initializes Instruction Memory, Register File, and Data Memory, respectively, according to "/testbench/IM.v", "/testbench/RF.v", "/testbench/ DM.v".

b. **Clock**
Generate a periodic clock (clk) to drive the CPU module in the testbench.

c. **Addressing and Termination**
Testbench initializes Addr_In signal to 0 and assigns Addr_Out to Addr_In before each positive edge of clk.（Part III will use PCWrite as the control signal, and Addr_In will be updated only when it is "1"）. When Addr_In is greater than or equal to the maximum address space （0x7D）of the current job instruction, Testbench ends the execution and outputs the current register and memory content ("/testbench/RF.out", "/testbench/DM.out"). The following figure shows the basic waveform of Testbench action:



Note: When the system Addr_Out fails or the program is in an infinite loop, please terminate the simulation and determine the problem manually.

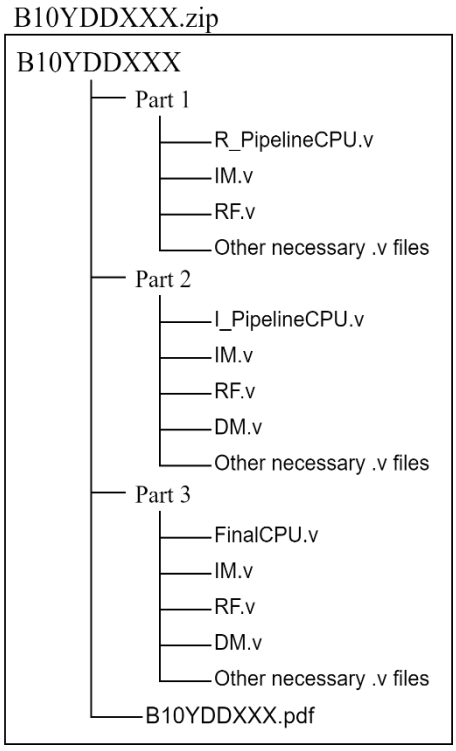
Submission

Report (B10YDDXXX.pdf) : (no more than 30 pages)

a. Cover.
b. Screenshots and descriptions of each module code.（No screenshots are required for extended modules, but a brief description of their functions is required）
c. Screenshots and descriptions of the execution results ("/testbench/RF.out", "/testbench/DM.out") of the sample programs ("/testbench/IM.dat") in each part.
d. Conclusion and insights.
e. Export it as a PDF format and name the file by the student ID - "B10YDDXXX.pdf ".

Compressed files (B10YDDXXX.zip):

- Report (B10YDDXXX.pdf)
- Part I
    a. R_PipelineCPU.v
    b. IM.v
    c. RF.v
    d. Other necessary .v files
- Part II
    a. I_PipelineCPU.v
    b. IM.v
    c. RF.v
    d. DM.v
    e. Other necessary .v files
- Part III
    a. FinalCPU.v
    b. IM.v
    c. RF.v
    d. DM.v
    e. Other necessary .v files



Score :

a. Main program: Part 1 (30%), Part 2 (30%), Part 3 (10%). All programs are tested by an external testbench.
b. Screenshots of each program and describe the process and method (15%).
c. The execution results of the sample programs in each part, screenshots, and explanations (10%).
d. Format/Conclusion/Completeness (5%).
e. No plagiarism.

Submission time: Upload to Moodle before 13:00 on 111/05/19