

2017/5/4

Final Project – Pipelined CPU



Outline

- Goal
- Grade
- Project Structure

Goal

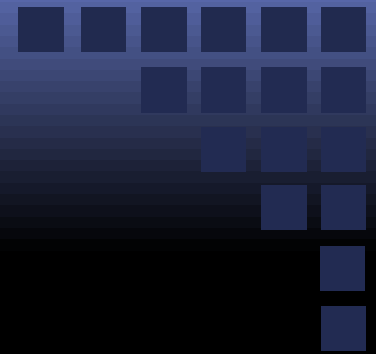
- Implement a pipelined CPU
 - Modify the single-cycle CPU design to a pipelined CPU.
(Please use released modules to implement.)
- Implement **forwarding** into pipelined CPU
- Instruction set:
ADD, SUB, AND, OR, SLT, SLTI, ADDI, LW, SW, BEQ.
(BEQ instruction will fetch the next instruction; LW instruction will not cause hazard problem.)

Assembly Example – with forwarding

```
addi $1, $0, -10
addi $2, $0, 9
and  $7, $1, $2
addi $3, $0, 1
addi $4, $0, 2
sub  $8, $3, $4
or   $9, $4, $8
addi $5, $0, 16
addi $6, $0, 8
slt  $10, $5, $6
or   $11, $6, $10
```

⌘ Note that we will check cycle counts to make sure that the CPU is pipelined version.

Binary code



```
001000000000000011111111111110110
00100000000000001000000000000001001
000000000001000100011100000100100
00100000000000001100000000000000001
00100000000000010000000000000000010
0000000000110010001000000000100010
000000000100010000100100000100101
00100000000000010100000000000010000
00100000000000011000000000000001000
000000000101001100101000000101010
000000000110010100101100000100101
```

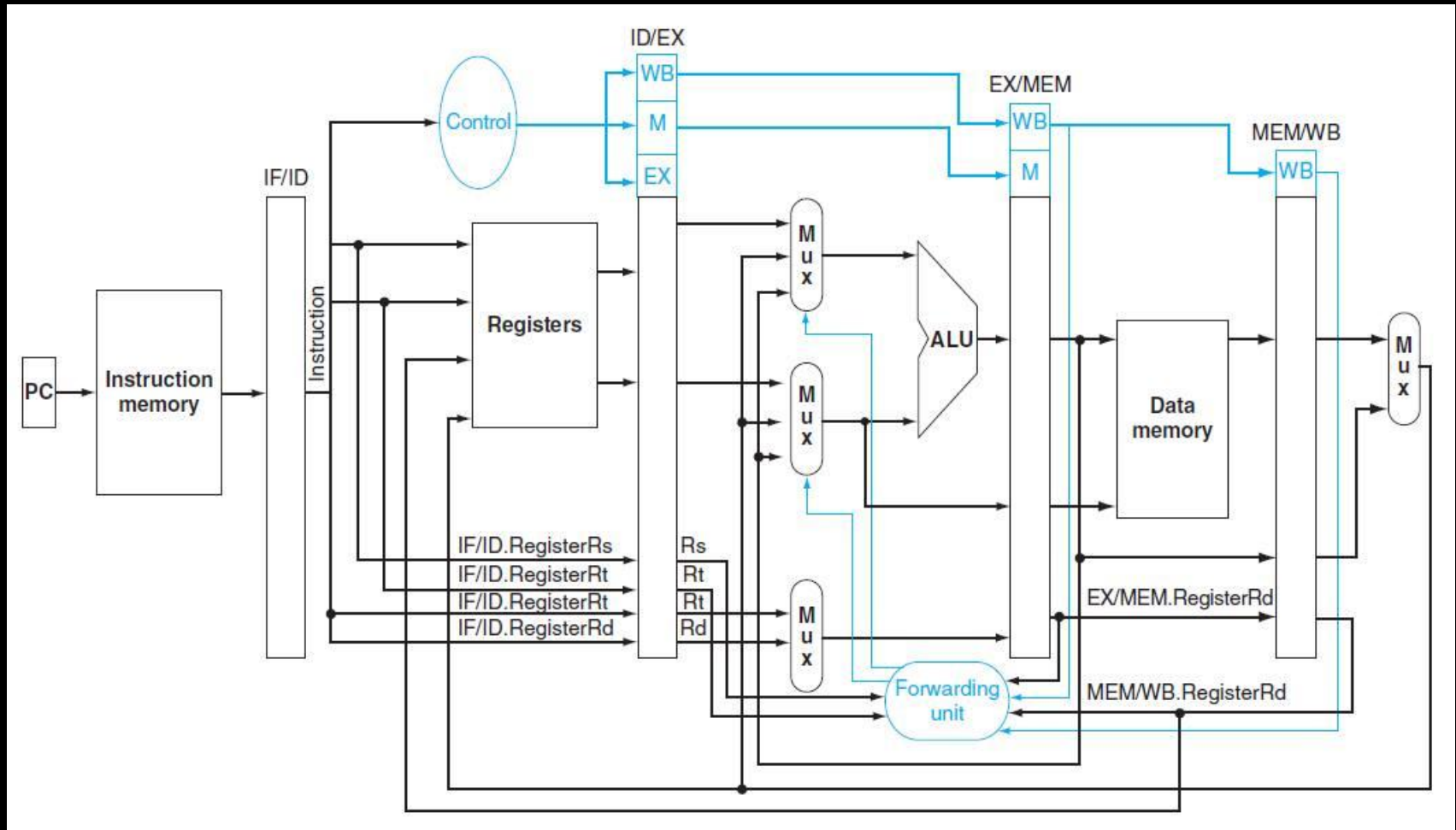
Result

[CASE 3]

```
r0 = 0
r1 = -10
r2 = 9
r3 = 1
r4 = 2
r5 = 16
r6 = 8
r7 = 0
r8 = -1
r9 = -1
r10 = 0
r11 = 8
r12 = 0
r13 = 0
r14 = 0
r15 = 0
r16 = 0
r17 = 0
r18 = 0
r19 = 0
r20 = 0
r21 = 0
r22 = 0
r23 = 0
r24 = 0
r25 = 0
r26 = 0
r27 = 0
r28 = 0
r29 = 0
r30 = 0
r31 = 0
```



Pipelined CPU



Project Structure

▣ Good coding style

- Only one module in one .v file
- Module name has to be the same as filename
 - ▣ Ex: module MUX()... endmodule in MUX.v

▣ Modules are created and connected in one top-module (Pipe_CPU.v)

Source

■ Given modules

- Pipe_CPU.v (the top-module)
- Pipe_Reg.v
- ForwardinUnit.v
- Reg_File.v
- MUX_3to1.v

■ And others modules from project 2.

■ Please modify Makefile from project 2 for you to write project. You don't have to upload it to iLMS.

Run simulation with makefile

- Put the makefile in your file directory
 1. Type “make”
 2. Makefile will do *\$ncverilog all .v file you need*

- By using the makefile, you don't have to type “ncverilog...” all the time.

Strict rule

- ❑ 1. Do not modify **Instr_Memory.v**, **Test_Bench.v**, **Reg_File.v**, **Pipe_Reg.v**. TA will replace it with our default version
- ❑ 2. Do not add/modify any clock and rst settings in your program. (ex: #delay)

Strict rule

- 3. Do not change any file name of the given .v files.
- 4. There shouldn't be \$stop in your code

※ 如果無法成功通過Script，需要助教手動更改，一律扣10分。

If your code cannot pass our script, - 10 points

Notice

- ▣ Deadline : 5/22, 23:59
 - No reason for late submission

- ▣ Please upload all the .v files to iLMS.
 - No file I/O
 - No clk and rst setting
 - You don't have to upload Test_bench.v
 - **DO NOT** compress your file

Notice

- ▣ 原本project2 deadline是5/14,23:59
- ▣ 這次有兩個繳交方案:
- ▣ 交project2(DL:5/14) & project3(DL:5/22)
 - 各自評分
- ▣ 只交project3(DL:5/22)
 - 如果project3 能通過所有測資，就能得到project2跟project3的分數
 - 如果只過8成測資，就只能得到project2跟project3的8成分數，以此類推

Notice

▣ Violation of each strict rule

-10 points

▣ Copy -----

Get 0 point

▣ 遲交 -----

Get 0 point

Hint

- You only need to modify two modules (**Pipe_CPU.v** and **ForwardinUnit.v**) to complete this project.
- You will need **MUX_3to1.v** to complete this project.
- Please make sure that the modules from your project 2 work correctly.

Hint

□ The modules you need are as follow:

- Adder.v
- ALU.v
- ALU_Ctrl.v
- Data_Memory.v
- Decoder.v
- Instr_Memory.v (do not modify)
- MUX_2to1.v
- MUX_3to1.v (we will provide)
- Makefile
- ProgramCounter.v
- Shift_Left_Two_32.v
- Reg_File.v (do not modify)
- Sign_Extend.v
- Test_bench.v (do not modify)
- Pipe_CPU.v (we will provide)
- Pipe_Reg.v (we will provide)
- ForwardinUnit.v (we will provide)