2016/5/12

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

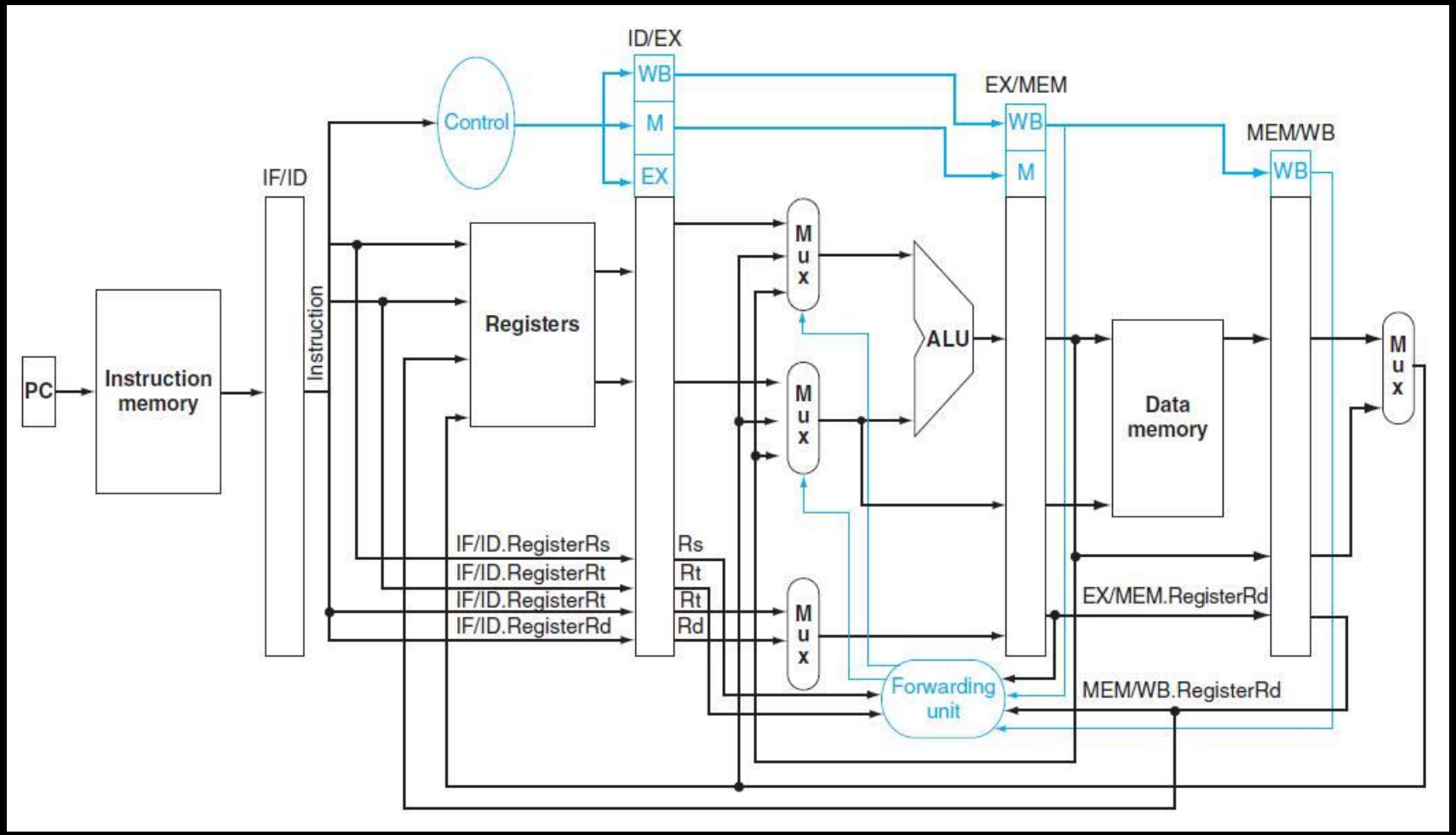
00100000000000111111111111110110
00100000000001000000000000001001
00000000010001000111000001001000
00100000000001100000000000000001
00100000000010000000000000000010
00000000110010001000000100010
00000001000100001001000001001010
00100000000010100000000000010000
00100000000011000000000000001000
00000000101001100101000000101010
00000000110010100101100000100101

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
m0 = 0
m1 = -4
m2 = 1
m3 = 0
m4 = 0
m5 = 0
m6 = 0
m7 = 0
m8 = 0
m9 = 0
m10 = 0
m11 = 0
m12 = 0
m13 = 0
m14 = 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
- 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. Put all your .v file under directory “PipeCPU_學號” and compress as “PipeCPU_學號.tar.gz” (No file I/O)

2. Do not modify Instr_Memory.v, Test_Bench.v, Reg_File.v, Pipe_Reg.v. TA will replace it with our default version.

Strict rule

3. Do not add/modify any clock and rst settings in your program. (ex: #delay)

4. Do not change any file name of the given .v files.

5. 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40分。

If your code cannot pass our script, - 40 points.

Compress Your Folder

- Your target folder name must be PipeCPU_學號.
- Command
  - tar –zcvf PipeCPU_學號.tar.gz PipeCPU_學號

Notice

- Deadline : 5/26 23:59
- Please upload “PipeCPU_學號.tar.gz” to iLMS.
- Do NOT mail your code to TA. We will use your submitted file on iLMS to grade your project.

Notice

- Violation of each strict rule — -40 points
- Program not implemented with pipeline — Get 0 point
- Copy ---------- — Get 0 point

Hint

- You only need to modify two modules (Pipe_CPU.v and ForwardinUnit.v) to complete this project.
- Remember to extend MUX_2to1 to MUX_3to1.
- 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 (extended by MUX_2to1.v)
  - Makefile
  - ProgramCounter.vShift_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)