

Lab 5

5-Stage Pipeline Processor

TA-黃威淳

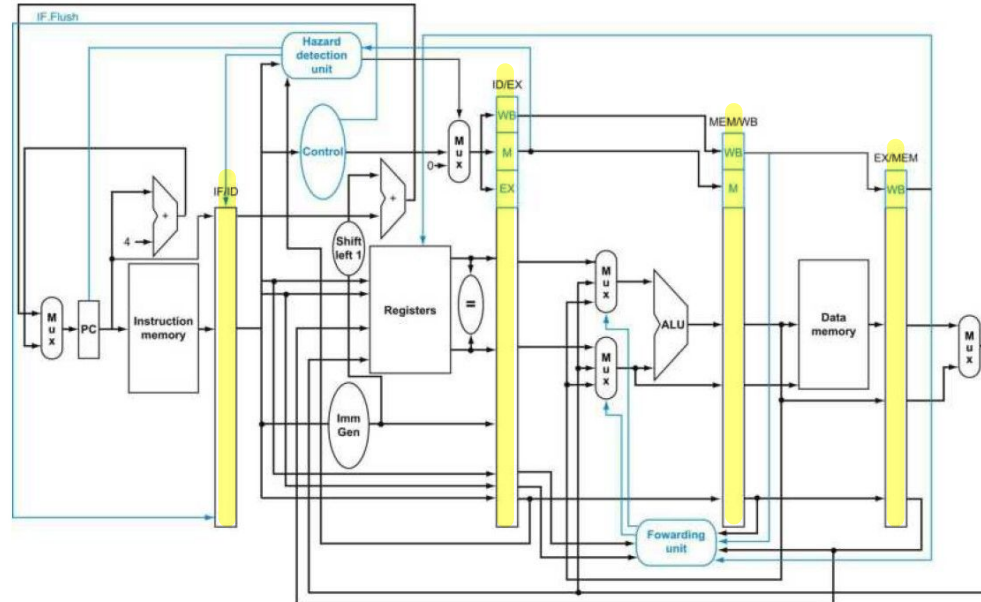
s094398.cs10@nycu.edu.tw

Notice

- [Lab 5 討論區](#)
- Lab 5 is **team work**

Overview

- Implement the datapath of 5-Stage Pipeline Processor as below
- You can refer to lab4 for additional details (mux, control signals, etc)
 - Appendix



Attached file

- **TODO**

- **Lab5Code**

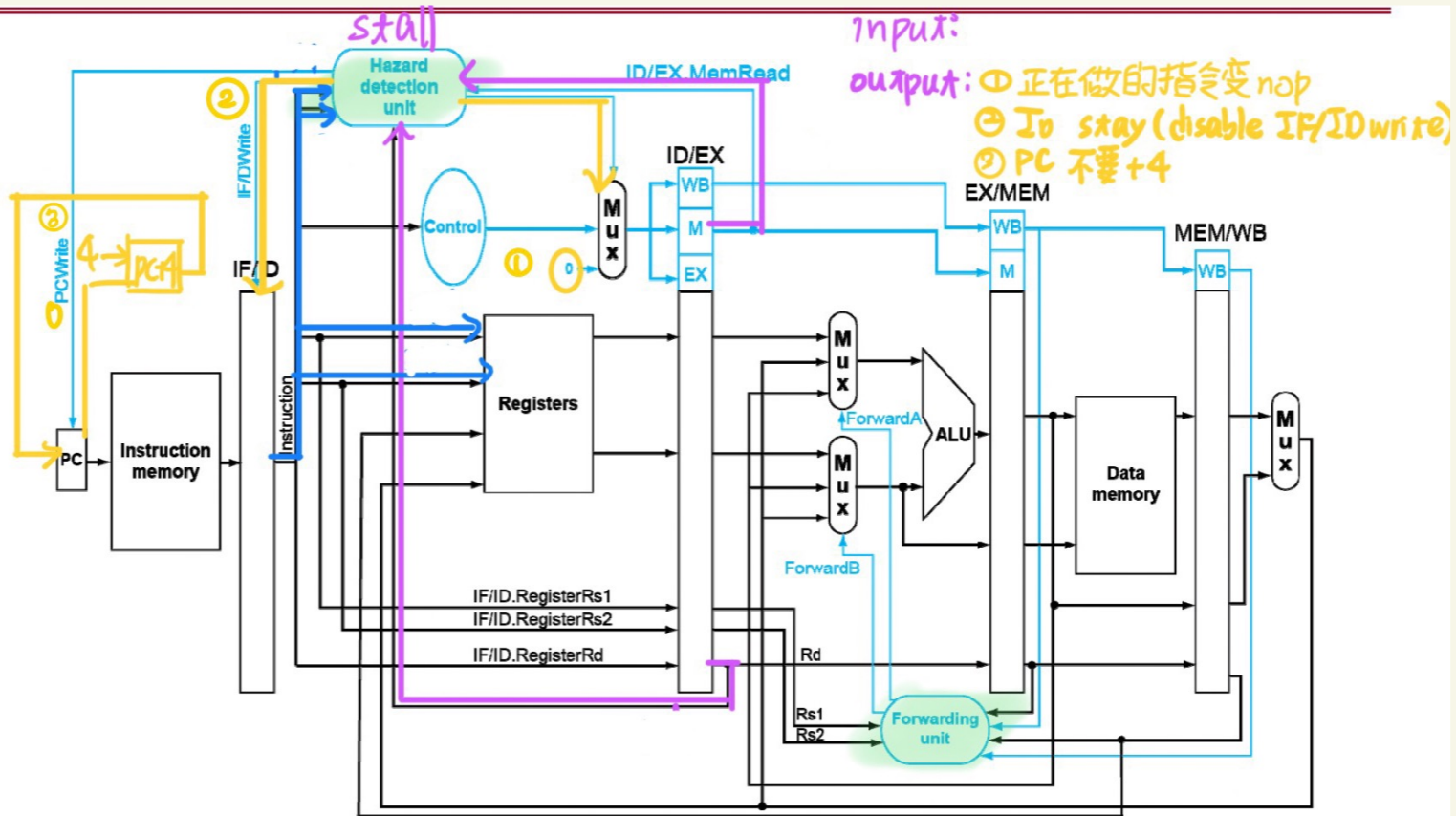
-
- Adder.v
 - ALU_Ctrl.v
 - alu.v
 - Decoder.v
 - Forwarding.v
 - Hazard_detection.v
 - Imm_Gen.v
 - MUX_2to1, 3to1.v
 - Shift_Left_1.v
 - Pipeline_CPU.v
 - IFID_register.v
 - IDEXE_register.v
 - EXEMEM_register.v
 - MEMWB_register.v

- **Validate the correction of your implementation**

- Please follow the **readme.txt** (Lab5 need to run in **UNIX-like** operating system)

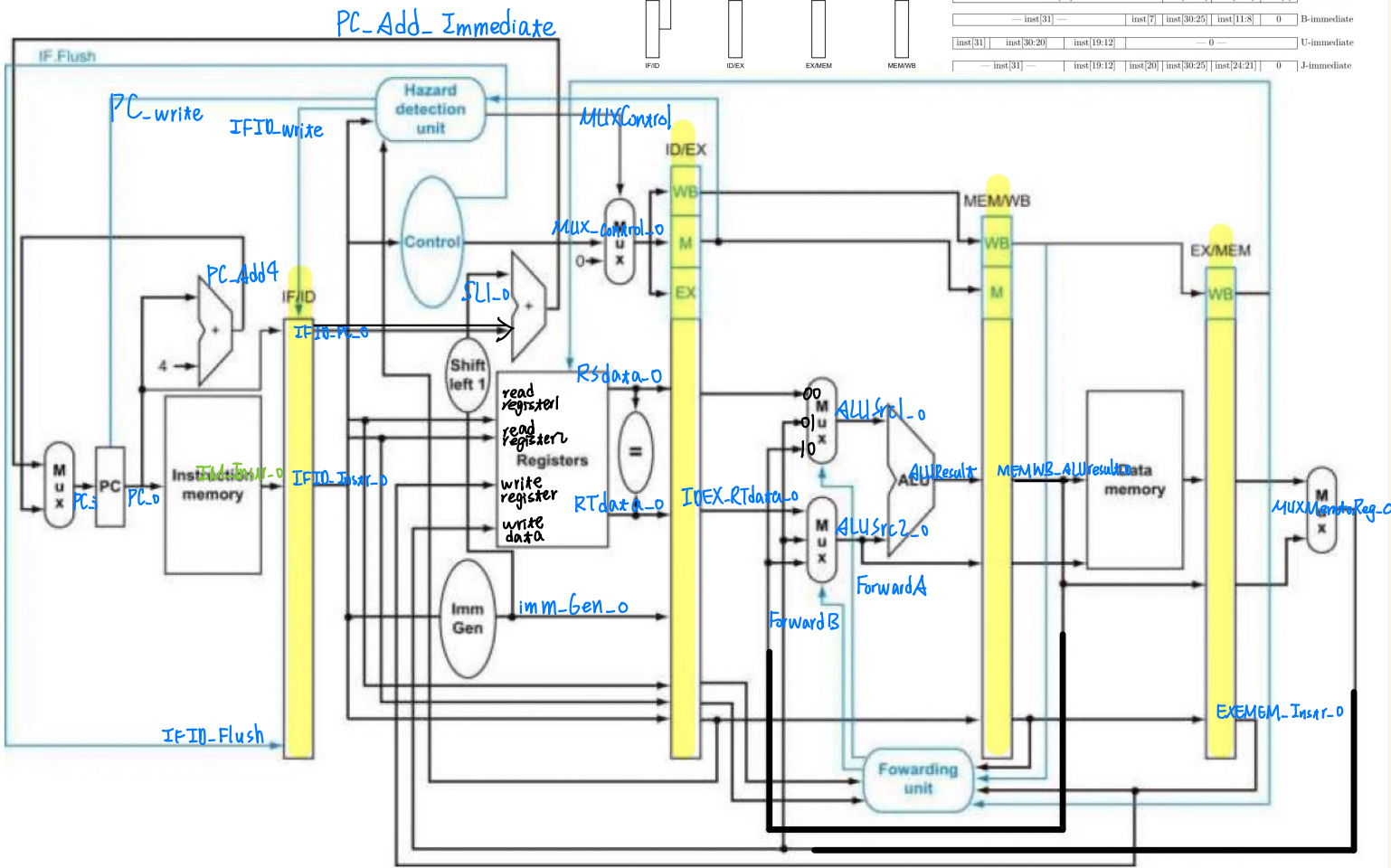
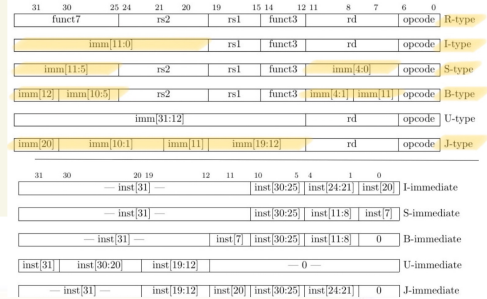
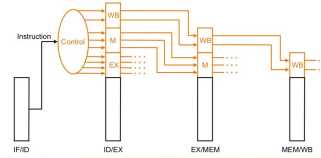
- **Testcase**

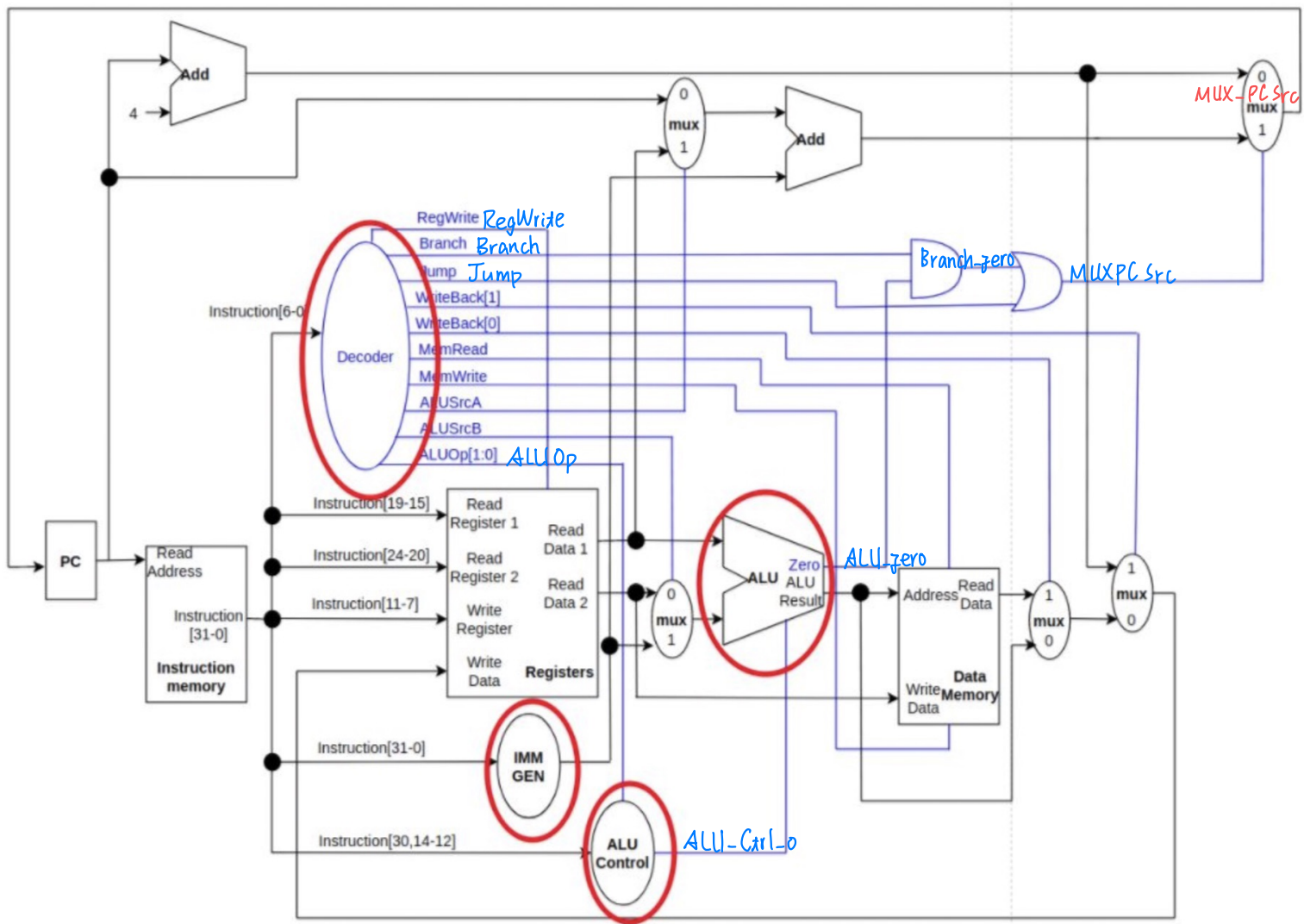
- Lab5Answer/testcase_for_compile.txt



all detection

Instruction	Execution/address calculation stage control lines				Memory access stage control lines		Write-back stage control lines		
	ALUOp	ALUSrc	Branch	Mem-Read	Mem-Write	Reg-Write	Memto-Reg		
R-format	10	0	0	0	0	1	0		
ld	00	1	0	1	0	1	1		
sd	00	1	0	0	1	0	X		
beq	01	0	1	0	0	0	X		



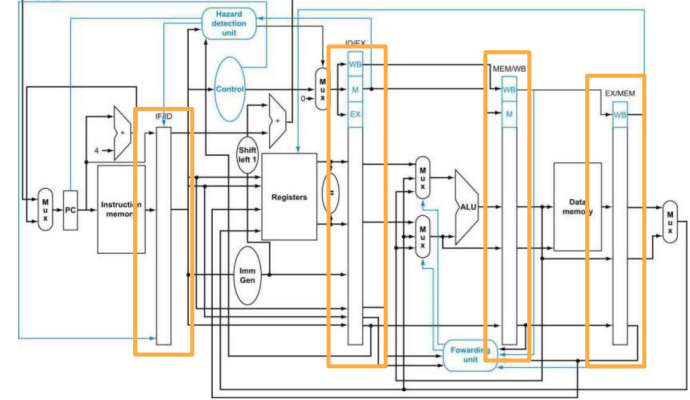


Pipeline register

- Pipeline stage

- IF: Instruction fetch from memory
- ID: Instruction decode & register read
- EX: Execute operation or calculate address
- MEM: Access memory operand
- WB: Write result back to register

- To hold information produced in previous cycle



} **IF/ID pipeline register**

} **ID/EXE pipeline register**

} **EXE/MEM pipeline register**

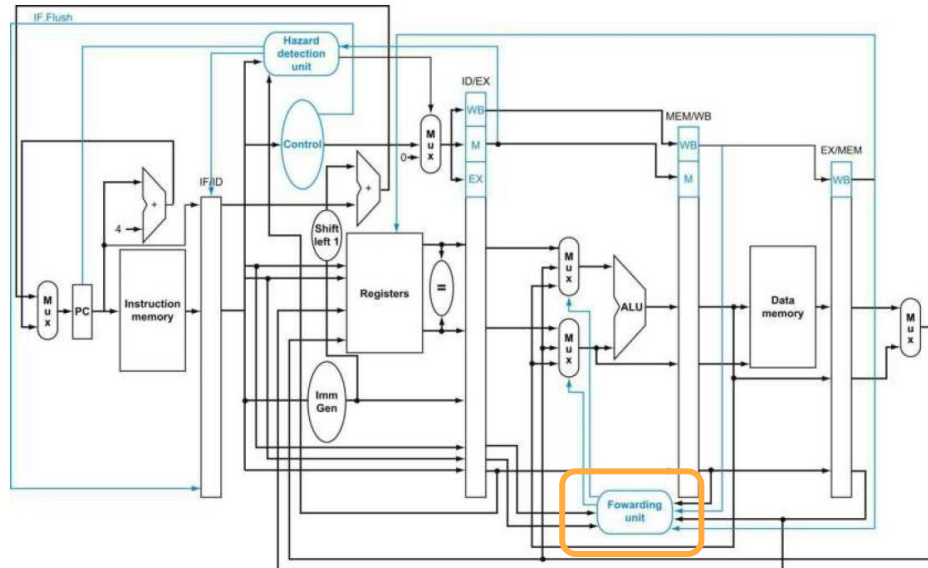
} **MEM/WB pipeline register**

Pipeline register

Hint : (Clock Positive triggered)
`always@(posedge clk_i) begin`
`·`
`· // Code HERE`
`·`
`end`

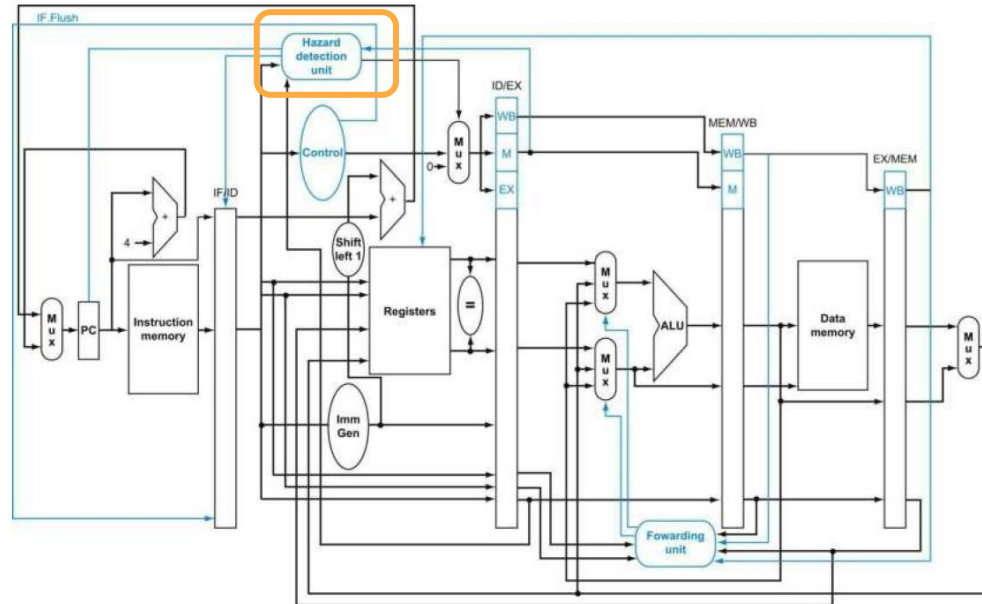
Forwarding unit

- Solving data dependency
 - Dependency detection and control
- You need to implement forwarding unit to follow the **Data Forwarding Control Conditions**



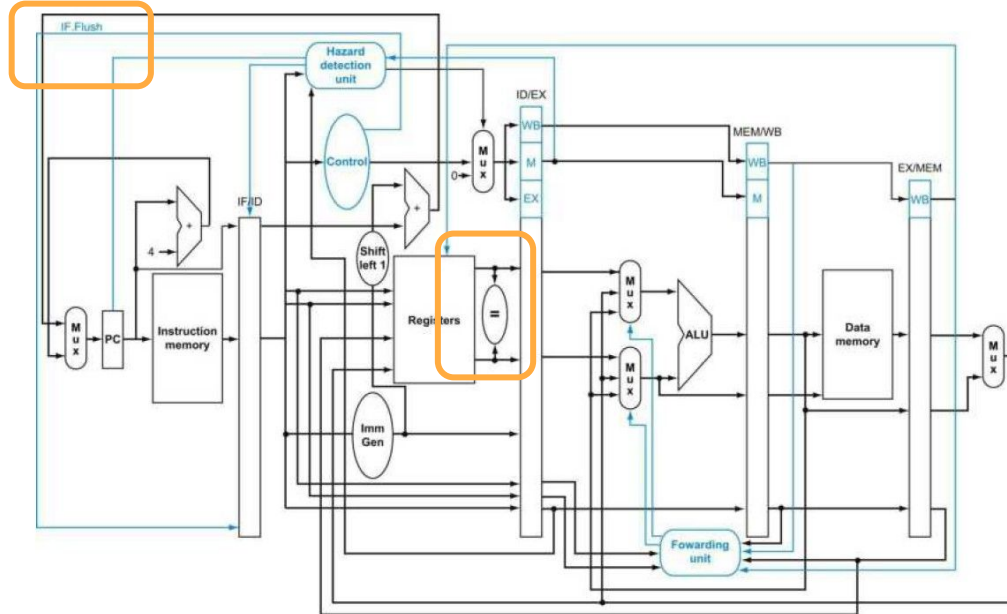
Hazard detection unit

- Solving Load-Use data hazard
- You need to implement Hazard detection unit to follow the condition



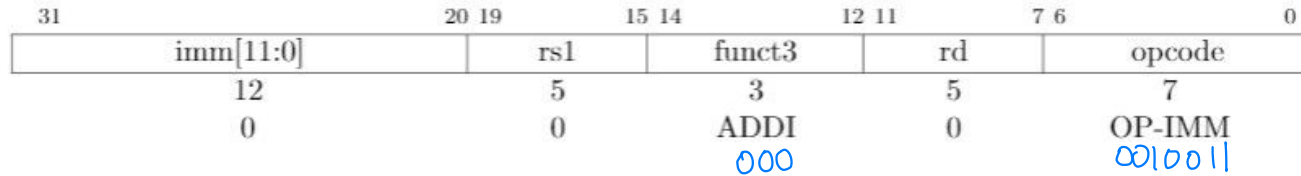
Branch Hazard

- Move branch decision from MEM to ID
- You also need to implement flush



NOP instruction

NOP Instruction



The NOP instruction does not change any architecturally visible state, except for advancing the pc and incrementing any applicable performance counters. NOP is encoded as `ADDI x0, x0, 0`.

Testcase

- Lab5Answer/testcase_for_compile.txt
- In all case, initially
 - **mem[16] = 2**
 - **All registers = 0**

Case 8 : (medium)

```
addi x2,x2,6
addi x0,x0,0
slli x2,x2,1
```

result:
x2 = 12

Case 11 : (Advanced)

```
lw x1,16(x2)
add x3,x2,x1
```

result:
x1 = 2,x3 = 2

Case 1 : (Basic)

```
addi x1, x0,50
4 nop
addi x2, x0, 18
4 nop
sub x3, x1, x2
4 nop
add x4, x1, x3
4 nop
or x5, x1, x4
4 nop
and x6, x2, x4
4 nop
```

result:
x1 = 50; x2 = 18; x3 = 32;
x4 = 82; x5 = 114; x6 = 18;

Testbench

- This script cannot run in Windows
- Put your .v file in Lab5Code
- `$ chmod +x ./lab5TestScript.sh && ./lab5TestScript.sh`

```
=====
testcase 1 pass
testcase 2 pass
testcase 3 pass
testcase 4 pass
testcase 5 pass
testcase 6 pass
testcase 7 pass
testcase 8 pass
testcase 9 pass
testcase 10 pass
testcase 11 pass
testcase 12 pass
testcase 13 pass
=====
Basic Score:30
Medium Score:40
Advanced Score:30
Total Score:100
```

```
***** CASE 13 *****
Case 13 Answer :
PC = 132
Data Memory = 0, 0, 0, 0, 2, 0, 0, 0
Data Memory = 0, 0, 0, 0, 0, 0, 0, 0
Data Memory = 0, 0, 0, 0, 0, 0, 0, 0
Data Memory = 0, 0, 0, 0, 0, 0, 0, 0
Registers
R0 = 0, R1 = 4, R2 = 0, R3 = 5, R4 = 0, R5 = 0, R6 = 0
R7 = 0
R8 = 0, R9 = 0, R10 = 0, R11 = 0, 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
-----
Your :
PC = x
Data Memory = 0, 0, 0, 0, 2, 0, 0, 0
Data Memory = 0, 0, 0, 0, 0, 0, 0, 0
Data Memory = 0, 0, 0, 0, 0, 0, 0, 0
Data Memory = 0, 0, 0, 0, 0, 0, 0, 0
Registers
R0 = x, R1 = x, R2 = x, R3 = x, R4 = x, R5 = x, R6 = x
R7 = x
R8 = x, R9 = x, R10 = x, R11 = x, R12 = x, R13 = x, R14 = x
R15 = x
R16 = x, R17 = x, R18 = x, R19 = x, R20 = x, R21 = x, R22 = x
R23 = x
R24 = x, R25 = x, R26 = x, R27 = x, R28 = x, R29 = x, R30 = x
R31 = x
Testcase 13 wrong
=====
```

Appendix

- Lab4 datapath

