Computer Architecture

Fall, 2019

Week 12

2019.11.25

[group2] **(對抗賽)**

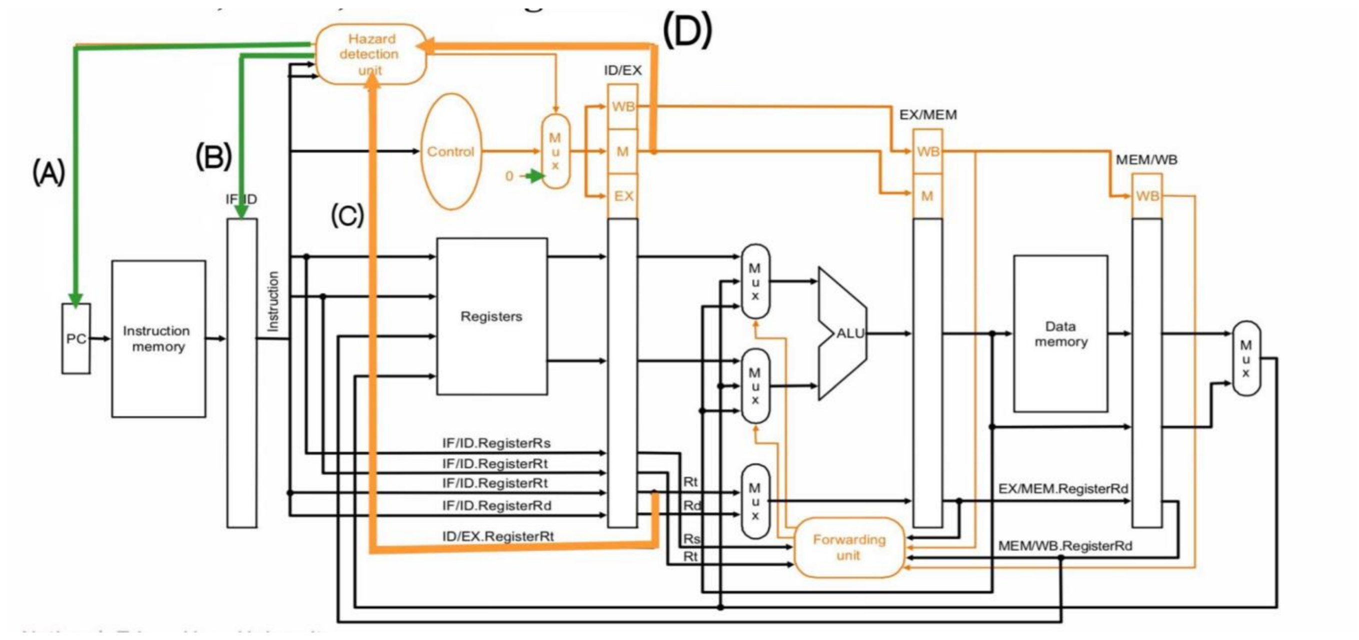
1. 下圖為 stalling 的示意圖(current stage: Decode)，請判斷選項是否正確 (是非題)

(A) 為決定 PC 是否寫新的 data 進去

(B) 為決定 IF/ID pipeline register 是否更新 data

(C) 檢查上一個 instruction 是否為 load

(D) 檢查上一個 instruction 的 rd 是否為現在 instruction 的 rs 或 rt



Ans:

(A) T

(B) T

(C) F 檢查上一個 instruction 的 rd 是否為現在 instruction 的 rs 或 rt

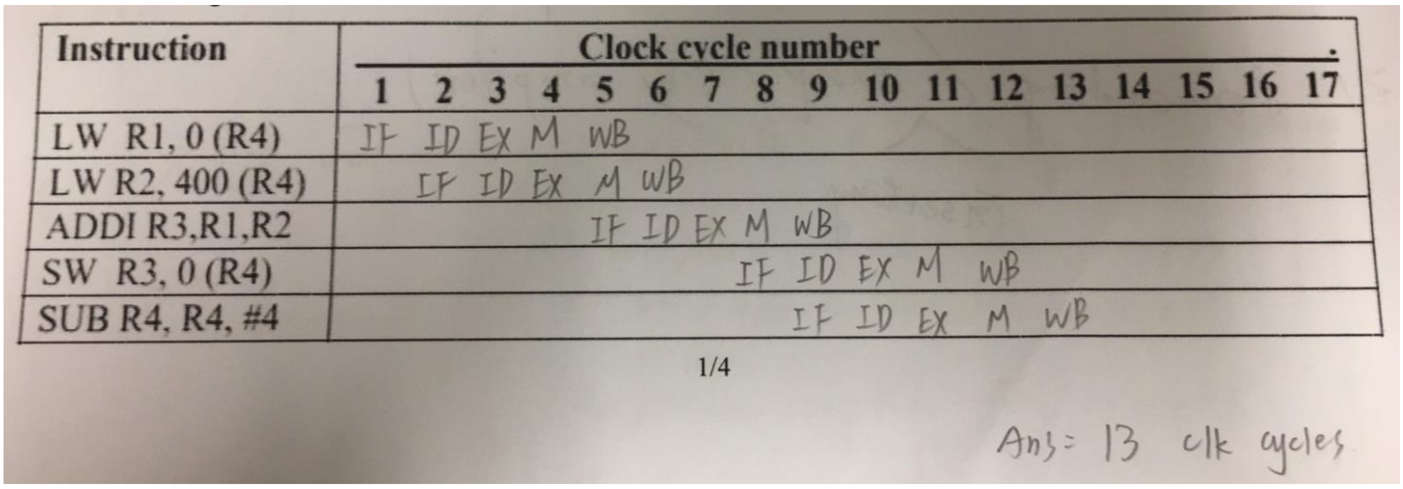
(D) F 檢查上一個 instruction 是否為 load

[group3] **(對抗賽)**

2. Calculate how many clock cycles will take execution of this segment on the simple pipeline without forwarding.

Instruction	**Clock cycle number**																
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
LW R1, 0 (R4)																	
LW R2, 400 (R4)																	
ADDI R3,R1,R2																	
SW R3, 0 (R4)																	
SUB R4, R4, #4																	
BNEZ R4, L1																	

Ans:



Instruction	**Clock cycle number**																
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
LW R1, 0 (R4)	IF	ID	EX	M	WB												
LW R2, 400 (R4)		IF	ID	EX	M	WB											
ADDI R3,R1,R2					IF	ID	EX	M	WB								
SW R3, 0 (R4)								IF	ID	EX	M	WB					
SUB R4, R4, #4									IF	ID	EX	M	WB				

1/4

Ans: 13 clk cycles

[group7] **(對抗賽)**

3. The following are some statements about hazards, indicate if they are True or False. Justify your answer if they are False.

   (a). The most efficient way to solve hazards is stalling the pipeline.

   (b). A compiler can solve all hazards without needing to insert NOPs.

   (c). Performance is hurt a lot if hazards are solved by stalling the pipeline.

   (d). All hazards can be solved forwarding data between stages.

Ans:

(a). F. Stalling the pipeline is a bad solution, since it hurts performance a lot.

(b). F. A compiler cannot solve all hazards by inserting NOPs, since sometimes it cannot find enough independent instructions to put between dependent ones.

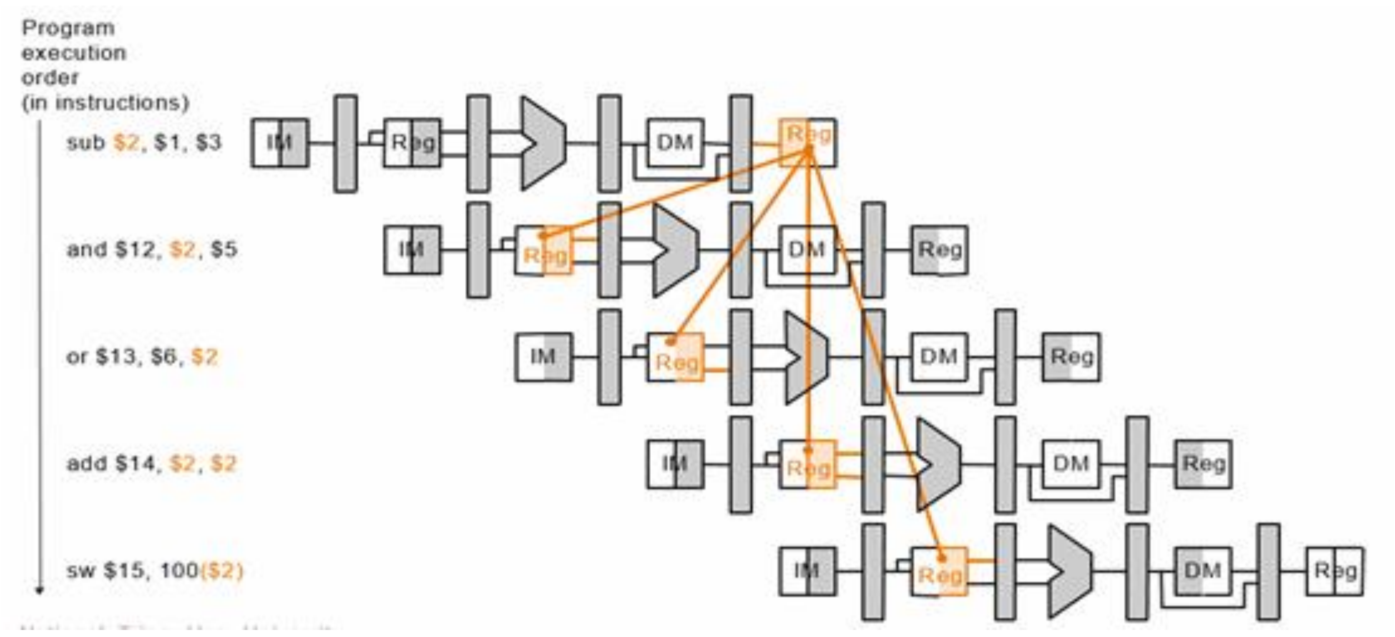
(c). T.

(d). Not all hazards can be solved by forwarding data. If the value hasn't been computed yet forwarding is not an option.

[group11] **(對抗賽)**

4. (a) In which instructions would cause data hazard and which would not? Explain why cause and why not cause. (Assume there's internal forwarding in register file)

   (b) Give two solutions to deal with the hazard.



Ans:

(a)

and or instr. cause hazard since they have to read $2 in CC3&CC4(but $2 ready in CC5)

add will not cause hazard since it read $2 in CC5. And register file is internal forwarding.

sw will not cause hazard since it read $2 in CC6.

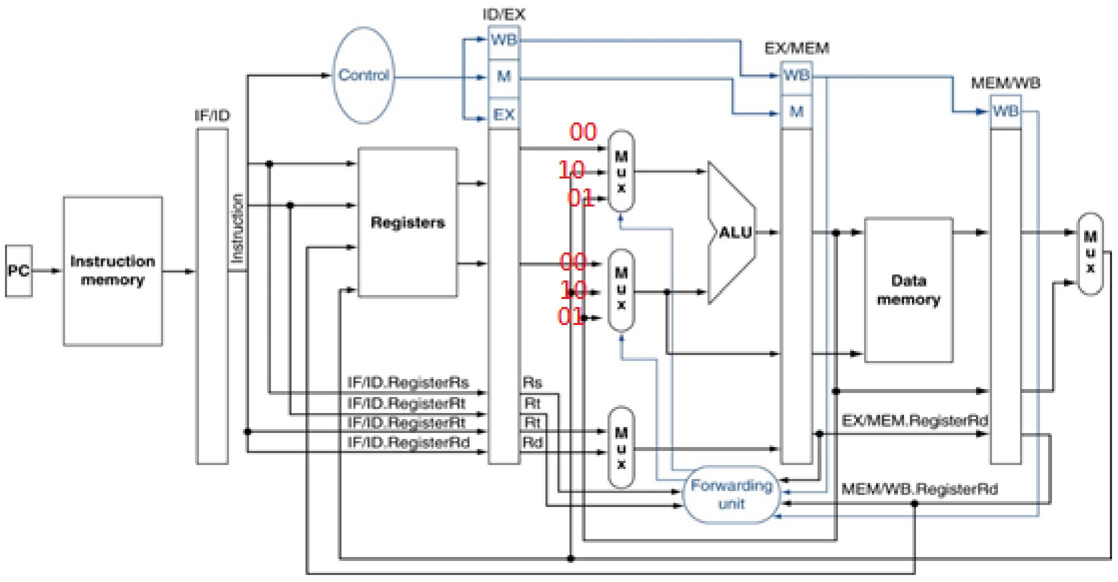
(b)

1. insert two NOPS between sub and and instruction.

2. use forwarding

[group6] **(對抗賽)**

5.



If the instruction is as below.

mul $2, $4, $5

sub $3, $6, $7

add $1, $2, $3

What should be the forward value of both input mux before the ALU according to the graph when executing third instruction, and what is both of which's hazard conditions?

Mux	registers	Forward value	Hazard condition
1	$s2		
2	$s3		

Ans:

Mux	registers	Forward value	Hazard condition
1	$s2	**10**	**MEM/WB.RegWrite && (MEM/WB.RegRd != 0) && (MEM/WB.RegRd=ID/EX.RegRs)**
2	$s3	**01**	**RegWrite && (EX/MEM.RegRd != 0) && (EX/MEM.RegRd=ID/EX.RegRt)**

[group4+8]

6. (a) What is Forwarding?

(b) 在什麼條件下需要考慮 Forwarding？什麼情況下不需要考慮 Forwarding?

Ans:

(a) [group 4]

不用等到前一個 instruction 執行到最後一個 stage，在 ALU 算出 result 後就將它送入下個 instruction 中當作 input

(b) [group 8]

Hazard conditions:

1a. EX/MEM.RegisterRd = ID/EX.RegisterRs

1b. EX/MEM.RegisterRd = ID/EX.RegisterRt

2a. MEM/WB.RegisterRd = ID/EX.RegisterRs

2b. MEM/WB.RegisterRd = ID/EX.RegisterRt

Two optimizations:

Don't forward if instruction does not write register

=> check if RegWrite is asserted

Don't forward if destination register is $0

=> check if RegisterRd = 0

[group9]

7. (多選題)選出下列正確的選項，並說明錯的選項

(A)如果有使用 internal forwarding 在 Register file，就能解決在同一個 stage 前一個 instruction 要 write 後面的要 read 的問題。

(B)解決 data hazards 時，Compiler insert NOPs 和 Foward 算是 software solution，Stall 則是 hardware solution。

(C)Insert Nops 會使 pipeline 變快。

(D)在 instruction 沒有 write register 或其 destination 為$0 時，就不需要用 forward 去解決 hazard。

(E)在製造 stall 的時候，為了讓 IF/ID 原地踏步，所以不能改變 PC 和 IF/ID，使用的方法是讓 write enable 變 disable。

Ans:

(A) (D) (E) are correct.

(B)Compiler insert NOPs (software) and Forward/Stall (hardware)

(C)NOPs 會使 pipeline 變慢

[group10] **(對抗賽)**

8. Consider a CPU with 5-stage pipeline (such as DLX) which runs the following assembly

/**/

Code：

```
     LOAD    R1, b
     LOAD    R2, c
     ADD     R3,R1,R2      /*R3=R1+R2；with a stall*/
```

```
    STORE   a, R3
    LOAD    R4, e
    LOAD    R5, f
    ADD     R6,R4,R5      /*R6=R4+R5；with a stall*/
    STORE   d, R6
/***********************************************/
```

Suppose there are two stalls in the above program, each falling between an occurrence of adjacent LOAD and ADD instructions. Show how to reschedule the above code so that all stalls can be removed.

Ans:

(1) LOAD 與 ADD 之間至少有一個 instruction 作 stall，ADD 與 STORE 之間有也一個 stall：

```
            LOAD     R1，b
            LOAD     R2，c
            LOAD     R4，e
            LOAD     R5，f
            ADD      R3，R1，R2
            ADD      R6，R4，R5
            STORE    a，R3
            STORE    d，R6
```

(2)LOAD 與 ADD 之間至少有兩個 instruction 作 stall，ADD 與 STORE 之間沒有 stall：

```
            LOAD     R1，b
            LOAD     R2，c
            LOAD     R4，e
            LOAD     R5，f
            ADD      R3，R1，R2
            STORE    a，R3
            ADD      R6，R4，R5
            STORE    d，R6
```