Computer Architecture

Fall, 2026
Week 5
2026.11.16

組別：_____ 簽名：_____

[group 1]

1. Consider a classic five-stage MIPS pipeline (IF, ID, EX, MEM, WB) with forwarding.

For each instruction sequence below, determine whether NOP instructions are required to avoid data hazards. If necessary, insert the minimum number of NOPs at the appropriate positions.

(a)

lw $t0, 0($t1)

add $t2, $t0, $t3

sub $t4, $t2, $t5

(b)

add $t0, $t1, $t2

sub $t3, $t0, $t4

and $t5, $t3, $t6

Ans:

(a) One NOP is required.
lw $t0, 0($t1)
nop
add $t2, $t0, $t3
sub $t4, $t2, $t5

Explanation:
The add instruction needs the value loaded by lw.
The loaded data is not available until the end of the MEM stage.
Forwarding alone cannot resolve this immediate load-use hazard.
Therefore, one NOP must be inserted between lw and add.

(b) No NOPs are required.
Explanation:

Both results are produced in the EX stage and can be forwarded directly to the next instruction's EX stage.

[group 2]

2. What is the possible solution of structural hazard?

Ans: add more resources (eg. Separate instruction and data memory)

[group 3]

3. Compare the three types of pipeline hazards (Structural, Data, and Control Hazards) and briefly explain their causes.

Ans:

1. Structural Hazard: Occurs when two instructions need the same hardware resource at the same time.

2. Data Hazard: Occurs when an instruction needs data from a previous instruction that is not yet available.

3. Control Hazard: Occurs when the next instruction address is uncertain due to a branch or jump instruction.

[group 5]

4. Consider the following instructions in a 5-stage MIPS pipeline:

```
lw    $t0, 0($s0)
add $t1, $t0, $t2
sub $t3, $t1, $t4
```

Assume the processor supports forwarding and stalling.

(a) Identify the data dependency between each pair of consecutive instructions.

(b) Which dependency can be solved using forwarding without a stall?

(c) Which dependency still requires a stall even with forwarding? Explain why.

Ans:

(a) lw → add: RAW dependency on $t0.

add → sub: RAW dependency on $t1.

(b) add → sub can be handled by forwarding. The result of add can be forwarded to the following sub instruction instead of waiting for it to be written back to the register file.

(c) lw → add requires a one-cycle stall. The value loaded by lw is not available until after its MEM stage, but the immediately following add needs $t0 in its EX stage. The value therefore becomes available too late for normal forwarding without delaying add.

[group 6]

5. 為何為什麼 Forwarding 無法完全消除五階段 MIPS Pipeline 中的 Load-use Data Hazard？應該如何解決這個問題？

Ans:

因為 lw 指令所載入的資料必須等到 MEM 階段結束時才會準備好，但下一條指令在 EX 階段就需要使用該資料，因此無法及時透過 Forwarding 傳遞。

為了解決這個問題，處理器會插入一個時脈週期的 Stall，也稱為 Bubble，讓下一條指令延遲執行，等待 lw 指令的資料準備完成後，再透過 Forwarding 將資料傳遞給相依的指令。

[group 7]

6. Question

1. List and briefly explain the three types of data hazards (assuming instruction i1 is followed by instruction i2).
2. Which two of these data hazards cannot happen in a standard MIPS 5-stage

pipeline, and why?

Ans:

1. Three types of data hazards:

• RAW (Read After Write): Instruction i2 tries to read an operand before instruction i1 writes it.

• WAR (Write After Read): Instruction i2 tries to write an operand before instruction i1 reads it.

• WAW (Write After Write): Instruction i2 tries to write an operand before instruction i1 writes it.

2. Hazards that cannot happen in a MIPS 5-stage pipeline:

• WAR and WAW.

• Reason: All instructions take 5 stages, operand reads are always strictly in stage 2 (ID), and writes are always strictly in stage 5 (WB).

[group 10]

7. A code sequence is:

```
add $4, $1, $3
or  $12, $5, $4
and $13, $4, $6
```

The data hazard of "or" can be checked by which of the following conditions?
(a) EX/MEM.RegisterRd = ID/EX.RegisterRs
(b) EX/MEM.RegisterRd = ID/EX.RegisterRt
(c) MEM/WB.RegisterRd = ID/EX.RegisterRs
(d) MEM/WB.RegisterRd = ID/EX.RegisterRt

Ans:

正確選項：(b) EX/MEM.RegisterRd = ID/EX.RegisterRt

過程

1. add $4, $1, $3 的目的暫存器為 $4（rd）。

2. or $12, $5, $4 的第二個來源暫存器為 $4（rt），因此 or 需要前一條 add 產生的結果。

3. 當 or 位於 EX 階段時，前一條 add 位於 MEM 階段；add 的結果存在 EX/MEM pipeline register。

4. 因此比較 EX/MEM.RegisterRd（$4）與 ID/EX.RegisterRt（$4），兩者相等，選 (b)。

對應的 Forwarding 控制訊號

EX/MEM.RegWrite = 1、EX/MEM.RegisterRd = $4 ≠ $0，且 EX/MEM.RegisterRd = ID/EX.RegisterRt，所以 ForwardB = 10（從 EX/MEM 傳送 add 的 ALU 結果）。ForwardA = 00。

補充：後面的 and 也會使用 $4，但它在 EX 階段時，add 已進入 WB，屬另一個 MEM/WB → EX forwarding；本題只問 or 的 hazard。

[group 11]

8. consider the following 5-stage MIPS pipeline instructions:

I1: add $t0, $t1, $t2

I2: sub $t3, $t0, $t4

I3: and $t5, $t0, $t6

Questions:

1. Identify all data hazards in the instruction sequence.

2. Which forwarding paths are required?

3. Assume that with a forwarding unit. Does this instruction sequence require any stall cycles? Explain why.

Ans:

1. Identify Data Hazards

- I1 writes to $t0.

- I2 reads $t0.
- I3 also reads $t0.

Therefore, there are two RAW (Read After Write) data hazards:

- I1 → I2
- I1 → I3

2. Required Forwarding Paths

- I1 → I2: EX/MEM → EX forwarding.
- I1 → I3: MEM/WB → EX forwarding.

3. Stall Cycles

No stall cycles are required.

The forwarding unit can directly forward the result to the EX stage without waiting for the WB stage to write the result back to the register file.

[group 12]

9. 在五階段 MIPS Pipeline 中，執行以下指令：

add $t0, $t1, $t2

sub $t3, $t0, $t4

問題： 這兩條指令之間會發生哪一種 Hazard？為什麼？

Ans:

會發生 RAW (Read After Write) Data Hazard。

因為第一條 add 指令會將運算結果寫入 $t0，而第二條 sub 指令需要讀取 $t0。

由於 Pipeline 會重疊執行指令，sub 可能在 add 寫入結果之前就需要使用 $t0，因此產生 Data Hazard。

[group 13]

10. Assume a 5-stage pipeline:

IF → ID → EX → MEM → WB

For each case, identify whether it is:

1. Data hazard

2. Structural hazard

3. No hazard

(a)

I1: ADD R1, R2, R3

I2: SUB R4, R1, R5

(b)

I1: ADD R1, R2, R3

I2: ADD R4, R5, R6

(c)

The processor has one memory unit. At the same time:

I1 is in MEM

I4 is in IF

Both stages need to access memory.

(d)

I1: LW R1, 0(R2)

I2: ADD R3, R1, R4

Ans:

(a) data hazard (RAW)
(b) no hazard
(c) structural hazard
(d) data hazard

[group 13]

11. Consider the MIPS 5-stage pipeline. The register file writes in the first half of a clock cycle and reads in the second half.

```
lw      $2, 20($1)
sub     $4, $2, $5
add     $6, $4, $3
```

(a) Assume there is no forwarding. How many NOPs must you insert, and where?

(b) Now assume there is forwarding and a hazard detection unit. How many stall cycles are needed, and why? For add, where does $4 come from?

Ans:

(a) You need 4 NOPs: 2 after lw and 2 after sub. Each instruction needs its source register from the instruction right before it, so it has to wait until the producer reaches WB.

(b) There is 1 stall. lw → sub is a load-use hazard: the loaded value is only ready at the end of MEM, but sub needs it at the start of EX in the same cycle, so forwarding can't fix it.

For add, $4 is forwarded from EX/MEM (ForwardA = 10), so no stall is needed there.