



Pipelined MIPS CPU Simulator

Pipelined CPU Implementation

- 請設計一個pipelined的cpu模擬器
(可以任何語言來模擬 ex:C & C++)
 - 以forwarding來解決hazard問題，換言之，需要實作forwarding的邏輯判斷
- 請以 3~4人為一組進行分組
- Input 為一名為memory.txt的文字檔
 - 裡面為MIPS的組合語言程式
- Output 請輸出此程式執行結果於一名為result.txt的文字檔
 - 請參閱後面投影片
- 專案完成後請撰寫一份專案報告，約3-4頁



Instructions

- lw
- sw
- add
- sub
- beq




Register Number

- 32 registers



Memory Size

- 32 words



Initial Values of Memory and Register

- 記憶體中的每個word都是1
- \$0暫存器的值為0，其他都是1

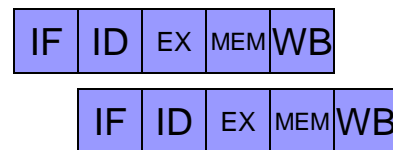
利用Forwarding解決Data Hazard

- 在指令即將進入EX階段時，實作課堂中介紹的EX與MEM Hazard判斷條件，將可以使用的指令結果盡快給ALU使用。當前指令是LW且需要它的結果時，安插stall，確保能夠讀到正確值。

■ Data Hazard

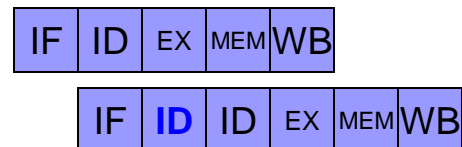
□ Case 1

add \$4, \$8, \$9
sub \$6, \$4, \$7



□ Case 2

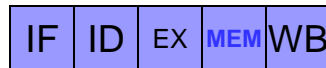
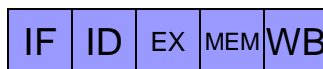
lw \$10, 20(\$9)
sub \$6, \$10, \$7



SW指令

- 在課堂中，我們並沒有提到**SW**是否需要 forwarding，它是需要的。

add \$6, \$4, \$5
sw \$6, 24(\$0)



利用Predict not taken解決 Control Hazard

- 在ID階段發現目前指令為beq時，IF直接抓取鄰近的下一個指令，等到beq指令完成ID階段後，到達EXE時，若預測錯誤，抹掉捉錯的指令，再抓取正確位置的指令；如果預測正確，則繼續執行

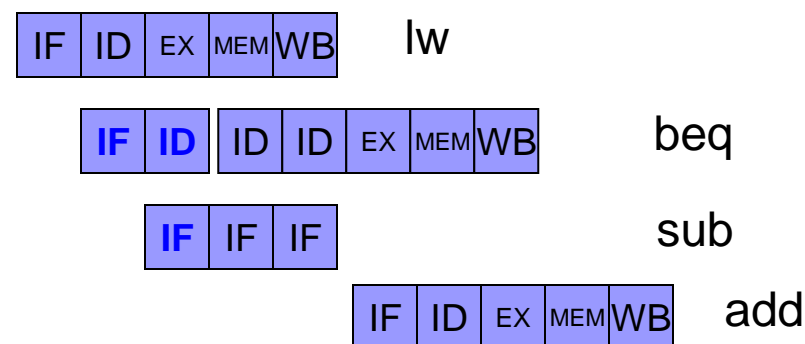
- Control Hazard

lw \$4, 8(\$5)

beq \$4, \$2, 1

sub \$7, \$8, \$9

add \$7, \$8, \$9



利用Predict not taken解決 Control Hazard

- 在ID階段發現目前指令為beq時，IF直接抓取鄰近的下一個指令，等到beq指令完成ID階段後，到達EXE時，若預測錯誤，抹掉捉錯的指令，再抓取正確位置的指令；如果預測正確，則繼續執行

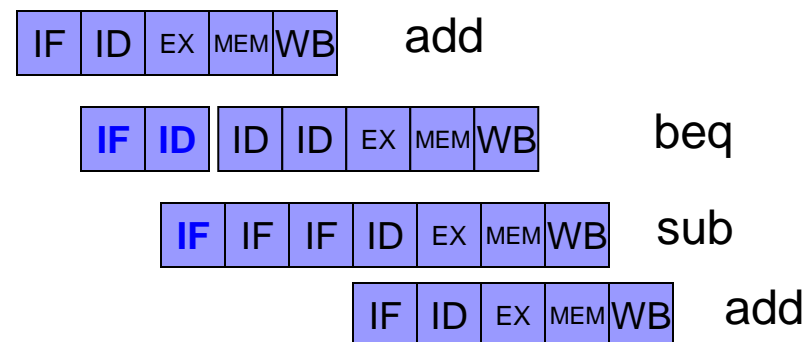
- Control Hazard

add \$4, \$5, \$3

beq \$4, \$2, 1

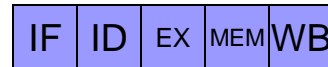
sub \$7, \$8, \$9

add \$7, \$8, \$9

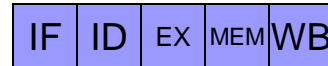


Input Example 1

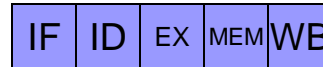
lw \$2, 8(\$0)



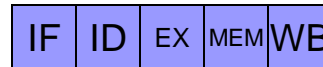
lw \$3, 16(\$0)



add \$6, \$4, \$5



sw \$6, 24(\$0)



8 cycles

\$6=2, W6=2

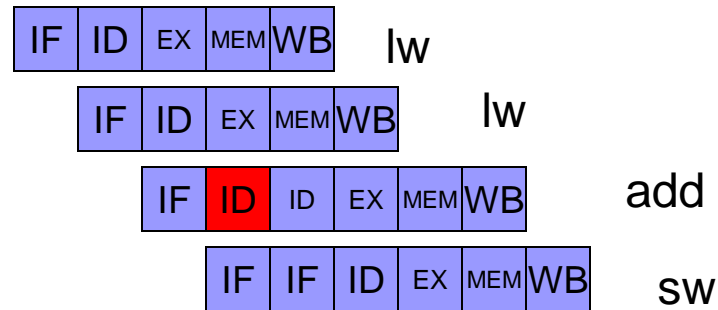
Input Example 2

lw \$2, 8(\$0)

lw \$3, 16(\$0)

add \$4, \$2, \$3

sw \$4, 24(\$0)



9 cycles

\$4=2, W6=2

Input Example 3*

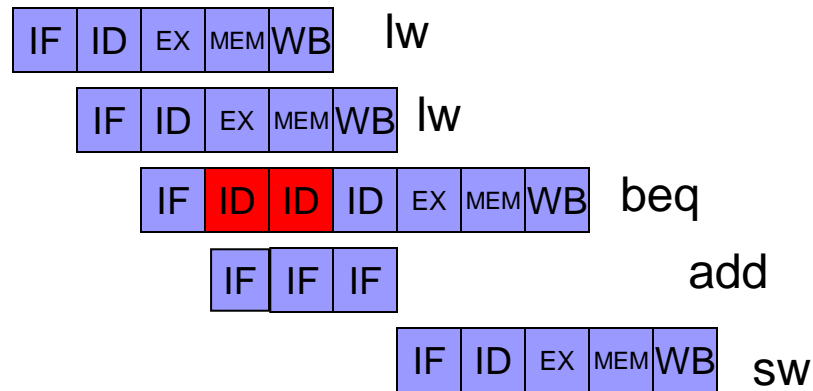
lw **\$2**, 8(\$0)

lw **\$3**, 16(\$0)

beq **\$2**, **\$3**, **1**

add \$4, \$2, \$3

sw \$4, 24(\$0)



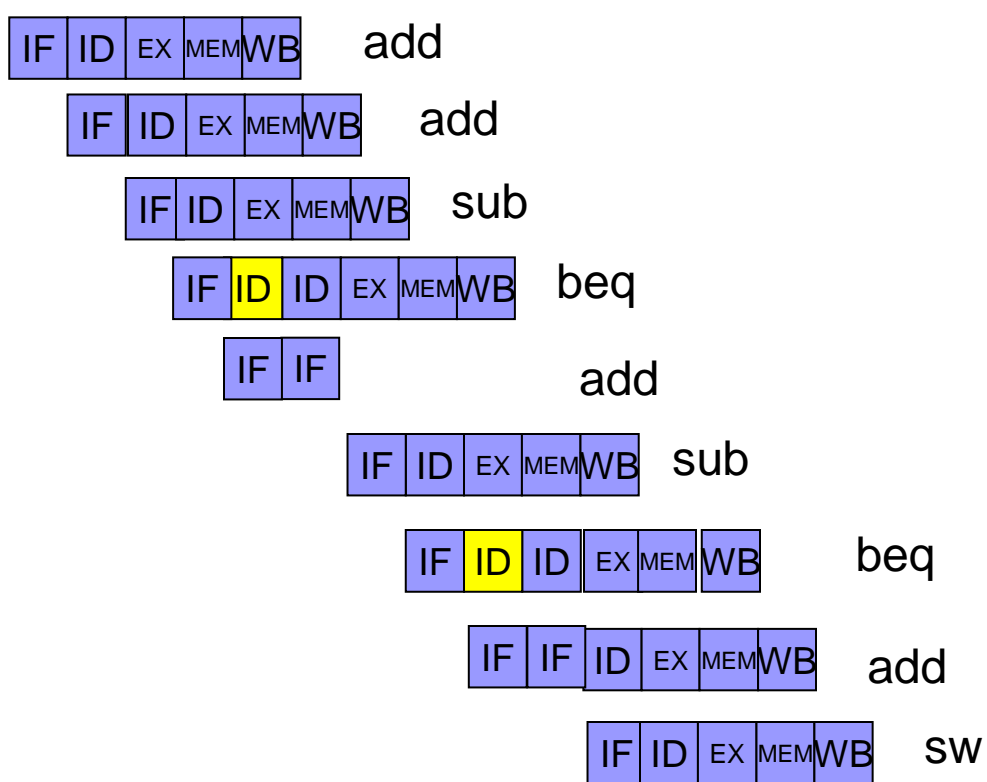
11 cycles

皆為1

Input Example 4

- add \$1, \$2, \$3
- add \$4, \$1, \$1
- sub \$4, \$4, \$1
- beq \$4, \$1, -2
- add \$4, \$1, \$4
- sw \$4, 4(\$0)

Input Example 4*



add \$1, \$2, \$3
 add \$4, \$1, \$1
 sub \$4, \$4, \$1
 beq \$4, \$1, -2
 add \$4, \$1, \$4
 sw \$4, 4(\$0)

15 cycles

\$1=2, \$4=2, W1=2

Input Example 5*

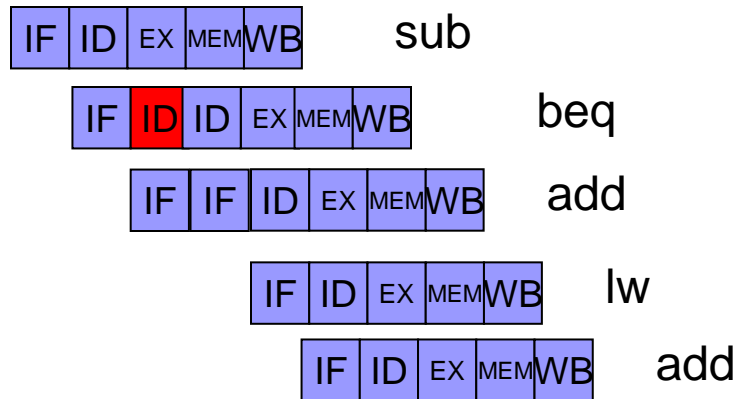
■ sub \$1, \$4, \$4

■ beq \$1, \$2, 2

■ add \$2, \$3, \$3

■ lw \$1, 4(\$0)

■ add \$4, \$5, \$6

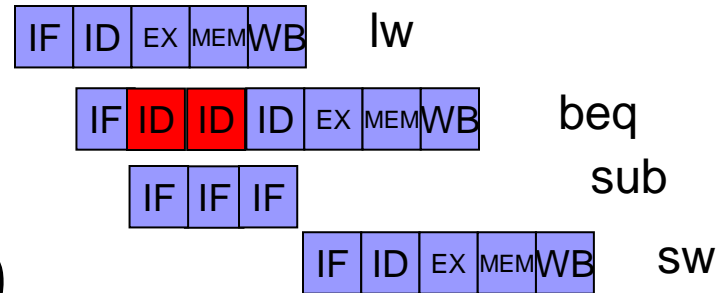


10 cycles

\$1=1, \$2=2, \$4=2, W1=1

Input Example 6*

- lw \$8, 8(\$0)
- beq \$4, \$8, 1
- sub \$2, \$7, \$9
- sw \$2, 8(\$0)



皆為1

10 cycles

Input Example 7

- add \$1, \$1, \$2
- add \$1, \$1, \$3
- add \$1, \$1, \$4
- sw \$1, 8(\$0)



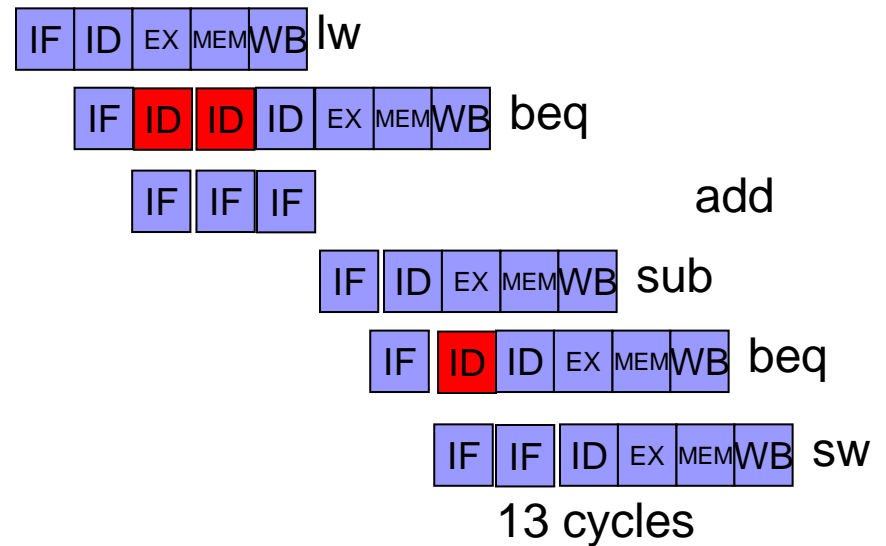
8 cycles

\$1=4, W2=4

Input Example 8

- lw \$4, 8(\$0)
- beq \$4, \$4, 1
- add \$4, \$4, \$4
- sub \$4, \$4, \$4
- beq \$4, \$1, -1
- sw \$4, 8(\$0)

\$4=0, W2=0



Output Requirements

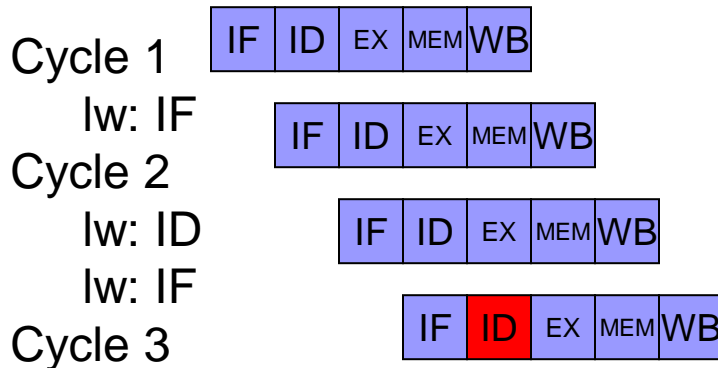
- 不同clock cycle時，在CPU中執行的指令狀態(顯示各指令在ID、EX、MEM、WB階段即將與未使用的signal值)
- 最後顯示
 - 執行該段指令需要多少cycle
 - 記憶體與暫存器執行後的結果

signal輸出順序: RegDst ALUSrc Branch MemRead MemWrite RegWrite MemtoReg

ex. lw: EX 01 010 11

OPcode	Stage	RegDst	ALUSrc	Branch	MemRead	MemWrite	RegWrite	MemtoReg
lw	EX	0	1	0	1	0	1	1

Output for Input Example 1



Cycle 6

lw: WB 11

add: MEM 000 10

sw: EX X1 001 0X

Cycle 7

add: WB 10

sw: MEM 001 0X

Cycle 8

sw: WB 0X

lw \$2, 8(\$0)

lw \$3, 16(\$0)

add \$6, \$4, \$5

sw \$6, 24(\$0)

lw: IF

Cycle 2

lw: ID

lw: IF

Cycle 3

lw: EX 01 010 11

lw: ID

add: IF

Cycle 4

lw: MEM 010 11

lw: EX 01 010 11

add: ID

sw: IF

Cycle 5

lw: WB 11

lw: MEM 010 11

add: EX 10 000 10

sw: ID

Output for Input Example 1

需要花8個cycles

\$0 \$1 \$2 \$3 \$4 \$5 \$6

0 1 1 1 1 1 2

W0 W1 W2 W3 W4 W5 W6...

1 1 1 1 1 1 2

lw \$2, 8(\$0)

lw \$3, 16(\$0)

add \$6, \$4, \$5

sw \$6, 24(\$0)

Output for Input Example 2

IF	ID	EX	MEM	WB
----	----	----	-----	----

IF	ID	EX	MEM	WB
----	----	----	-----	----

Cycle 1	IF	ID	ID	EX	MEM	WB
---------	----	----	----	----	-----	----

lw: IF

IF	IF	ID	EX	MEM	WB
----	----	----	----	-----	----

Cycle 2

lw: ID

lw: IF

Cycle 3

lw: EX 01 010 11

lw: ID

add: IF

Cycle 4

lw: MEM 010 11

lw: EX 01 010 11

add: ID

sw: IF

Cycle 5

lw: WB 11

lw: MEM 010 11

add: ID

sw: IF

Cycle 6

lw: WB 11

add: EX 10 000 10

sw: ID

Cycle 7

add: MEM 000 10

sw: EX X1 001 0X

Cycle 8

add: WB 10

sw: MEM 001 0X

lw \$2, 8(\$0)

lw \$3, 16(\$0)

add \$4, \$2, \$3

sw \$4, 24(\$0)

Output Example for Input Example 2

Cycle 9

sw: WB 0X

1 cycles →
lw \$2, 8(\$0)
lw \$3, 16(\$0)
add \$4, \$2, \$3
sw \$4, 24(\$0)

Output for Input Example 2

需要花9個cycles

\$0 \$1 \$2 \$3 \$4 \$5 \$6

0 1 1 1 2 1 1

W0 W1 W2 W3 W4 W5 W6...

1 1 1 1 1 1 2

```
lw $2, 8($0)
lw $3, 16($0)
add $4, $2, $3
sw $4, 24($0)
```

Output for Input Example 3

IF	ID	EX	MEM	WB
----	----	----	-----	----

IF	ID	EX	MEM	WB
----	----	----	-----	----

Cycle 1

IF	ID	ID	ID	EX	MEM	WB
----	----	----	----	----	-----	----

lw: IF

IF	IF	IF
----	----	----

Cycle 2

lw: ID

lw: IF

IF	ID	EX	MEM	WB
----	----	----	-----	----

Cycle 3

lw: EX 01 010 11

lw: ID

beq: IF

Cycle 4

lw: MEM 010 11

lw: EX 01 010 11

beq: ID

add: IF

lw \$2, 8(\$0)

lw \$3, 16(\$0)

beq \$2, \$3, 1

add \$4, \$2, \$3

sw \$4, 24(\$0)

Cycle 5

lw: WB 11

lw: MEM 010 11

beq: ID

add: IF

Cycle 6

lw: WB 11

beq: ID

add: IF

Cycle 7

beq: EXE X0 100 0X

sw: IF

Cycle 8

beq: MEM 100 0X

sw: ID

Output Example for Input Example 3

Cycle 9

beq: WB 0X

sw: EX X1 001 0X

Cycle 10

sw: MEM 001 0X

Cycle 11

sw: WB 0X

lw \$2, 8(\$0)

lw \$3, 16(\$0)

beq \$2, \$3, 1

add \$4, \$2, \$3

sw \$4, 24(\$0)

Output for Input Example 3

需要花11個cycles

\$0 \$1 \$2 \$3 \$4 \$5 \$6

0 1 1 1 1 1 1

W0 W1 W2 W3 W4 W5 W6...

1 1 1 1 1 1 1

lw \$2, 8(\$0)

lw \$3, 16(\$0)

beq \$2, \$3, 1

add \$4, \$2, \$3

sw \$4, 24(\$0)

專案驗收

- 測試的部份將會準備**5組**指令檔案，**4組**為上述例子，另一組驗收時再公佈
- 專案繳交期限為**2024/1/2**
- 繳交的檔案為 原始碼 / 執行檔 / 專案報告
 - 注意事項:
 - 原始碼編譯要能成功，否則將不給予分數
 - 若執行檔無法執行，將不給予分數
 - 成績取決於執行結果數據正確性與報告內容
 - 專案報告需有組員間工作分配、製作過程中所遭遇的問題、跟此專案相關之心得
- 會再訂驗收時程，每組在驗收時間講解與執行程式

Note: 有任何相關問題可Line詢問或是mail給助教

測試正確性

- 測試當天，請檢查每個測試檔
 1. 執行該程式所需的**cycle**數是否正確
 2. 任意找一個**cycle**，判斷每個指令在該**cycle**的狀態是否正確(**stage**、**signal**)
 3. 暫存器與記憶體中的值是否正確