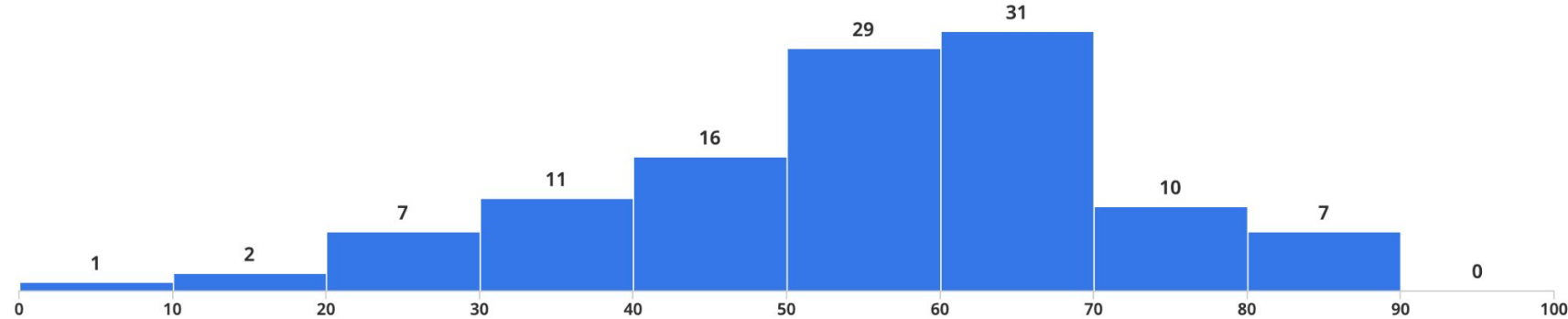


計算機結構期中考

11/19

成績分布



最小値

8.0

中位數

58.25

最大值

84.3

平均值

55.06

標準差 ?

16.44

1.(a) instruction-level parallelism

可以同時執行多個 instruction, 來提升整體 throughput。

常見作法有 pipeline, static / dynamic multiple issue.

而要注意的是, 在增加 ILP 的過程中, 可能會因為資料相依性, 分支預測錯誤等原因導致其效能增長無法達到理想情況。

評分標準:

2 分: 僅寫出定義

5 分: 定義以及和 ILP 相關的 實作 / 限制等議題, 若其中有錯誤則得 3 分

1.(b) yield

為晶片製造中，好的 die 在所有 die 之中的占比，亦即良率

公式為

$$\text{Yield} = \frac{1}{(1 + (\text{Defects per area} \times \text{Die area}/2))^2}$$

評分標準：

5 分：寫出定義 / 公式 其中之一且正確

3 分：兩者都有寫出，但有小錯誤

1.(c) dynamically linked library

在 run-time 才將相關的 object code modules link 起來。此舉為避免一樣的函式庫被重複載入，可以省空間，但速度會稍慢於 static linking。

評分標準：

5 分：有寫出 run-time 才 link 即可

3 分：有寫出 run-time 才 link 但多寫的部分有錯誤

1.(d) VLIW

very long instruction word, 在 static multiple issue 中, 把多個instruction 打包起來送入 pipeline 中。

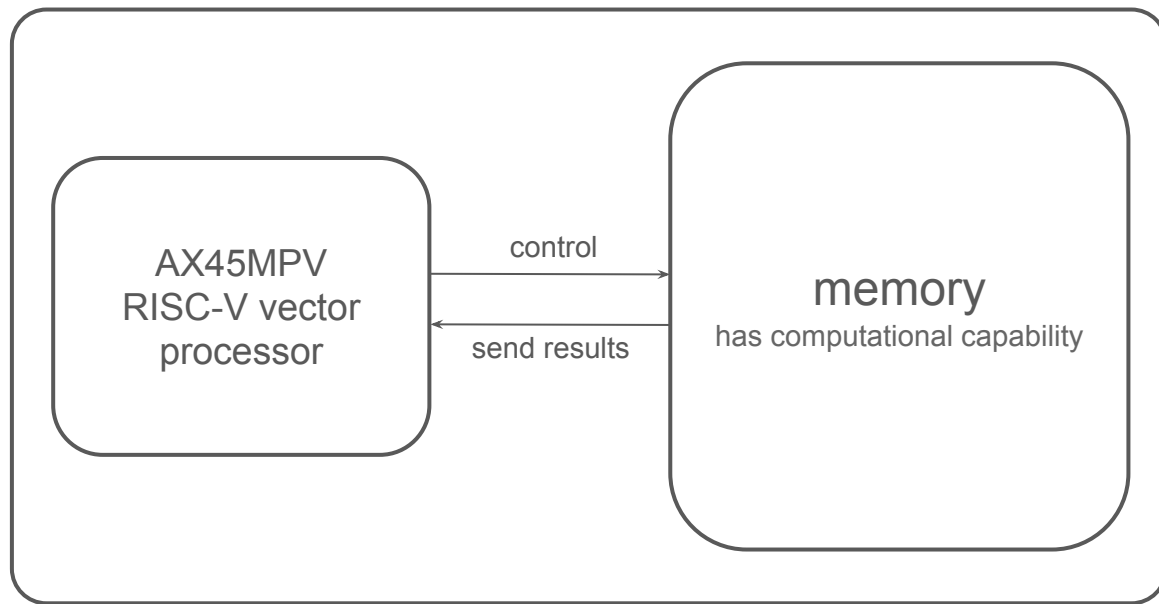
評分標準:

5 分:解釋用途

3 分:寫出全名

2. (e)

参考解答



AI inference accelerator

2. (e)

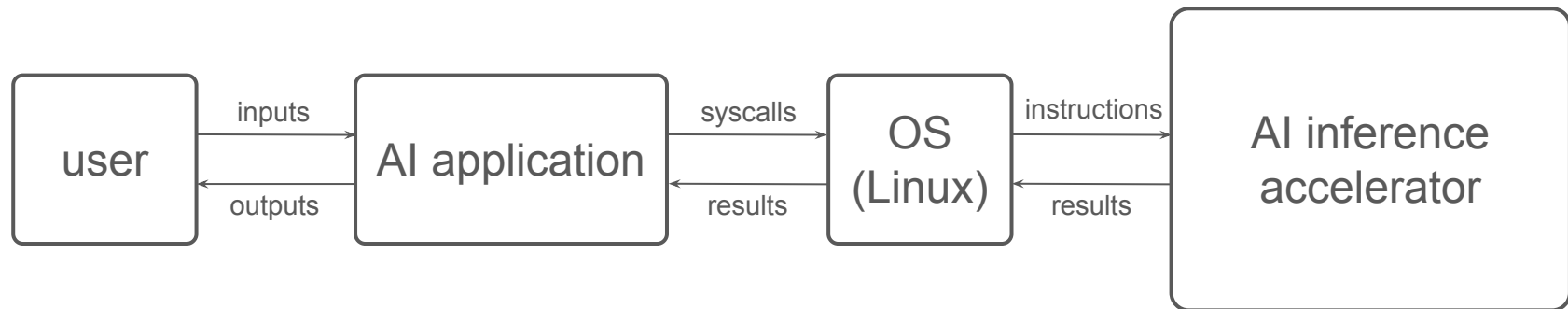
評分標準

- 5 分: 畫出一個 vector processor 跟一個 memory (並標註它有計算的能力), 而且兩者有用線連起來
- 4 分: processor 跟 memory 都有畫, 但架構圖有小錯
- 3 分: processor 跟 memory 只畫出其中一個
- 1 分: 有提到 processor 或是 memory, 但沒有畫出來, 或是有畫錯

注意: 這題的 memory 應該要放在 processor 之外, 而不是放在裡面

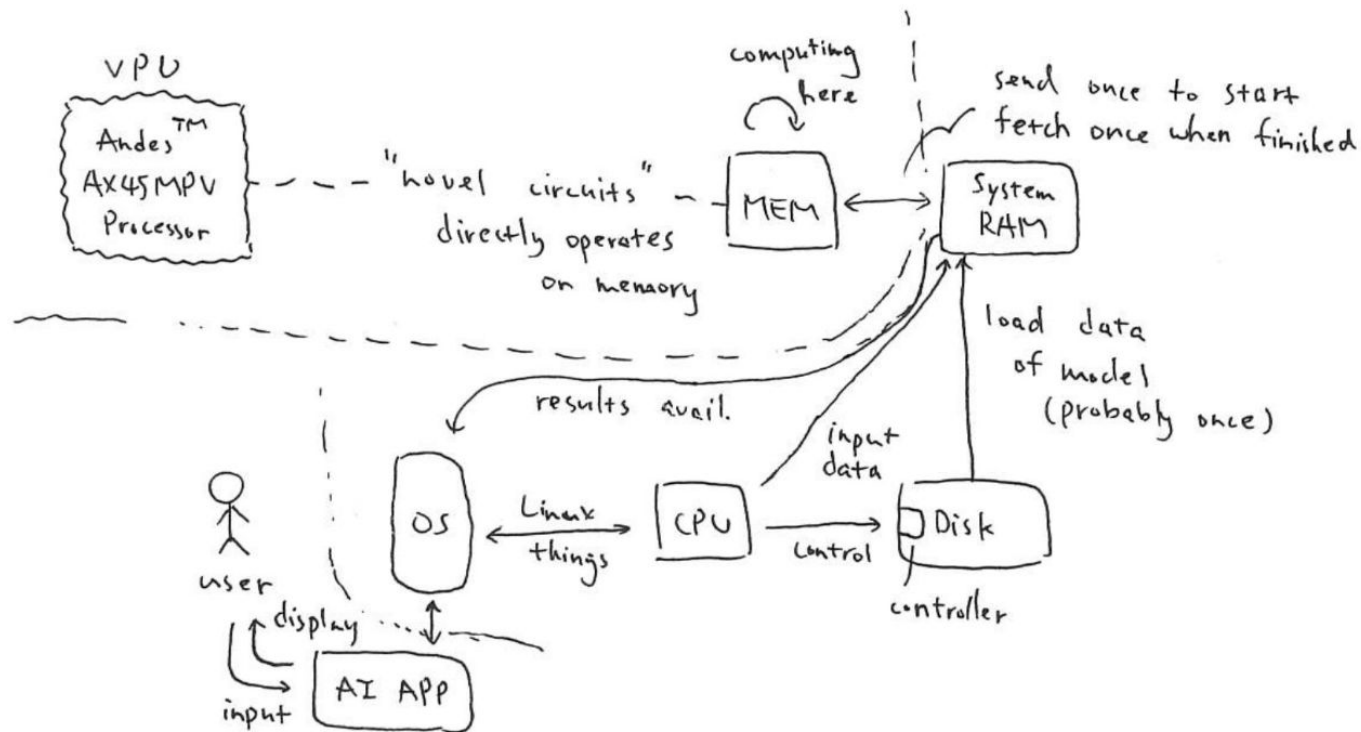
2. (f)

参考解答



2. (f)

参考解答



2. (f)

評分標準

- 5 分: 畫出 application, OS, 而且 data flow 正確
- 4 分: 只畫出 application, OS 其中一個, 或是 data flow 有小錯
- 2 分: 沒畫出軟體的 data flow, 但畫出硬體裡的, 而且正確

注意: 這題要畫出 software components

2. (g)

参考解答

- run AI inference two orders of magnitude (100x) faster than existing hardware
- a tenfold reduction in cost (10x cheaper)
- execute 99.99% of the operations needed for inference in on-chip memory

2. (g)

評分標準

- **5 分:** 以上 3 點有寫到 2 點, 而且都有使用文章提及的數字描述
- **3 分:** 以上 3 點只寫到 1 點, 而且有使用文章提及的數字描述
- **1 分:** 有提到任何優點, 但沒有使用數字描述

2. (h)

參考解答

- **Trends**

- model inference 的計算成本已超越 training 的成本
- AI model 越來越大, 所需要的計算量越來越大

- **Challenges**

- model inference 需要頻繁地存取 memory, 有嚴重的 memory bottleneck
- model 演進的速度快, 但硬體趕不上這個發展速度

- **Solutions**

- in-memory computing 讓計算可以直接在 memory 進行, 減少搬運資料的時間
- software-programmable 的 processor 可以應付快速演進的AI model
- Fractile 要研發一個 AI 加速器, 將自家的 in-memory computing 技術與 Andes 的 RISC-V vector processor 透過 ACE 整合起來

2. (h)

評分標準

- Trends (2分)
- Challenges (3分)
- Solutions (Fractile 如何與 Andes 合作佔 2 分, 如何解決 challenges 佔 3 分)

2. (h)

評分標準

- **Trends 寫到其中一個得 2 分**
 - model inference 的計算成本已超越 training 的成本
 - AI model 越來越大, 所需要的計算量越來越大
- **Challenges 寫到其中一個得 3 分**
 - model inference 需要頻繁地存取 memory, 有嚴重的 memory bottleneck
 - model 演進的速度快, 但硬體趕不上這個發展速度
- **Solutions 寫出其中一個解決 challenges 的方案得 3 分(不包含合作)**
 - in-memory computing 讓計算可以直接在 memory 進行, 減少搬運資料的時間
 - software-programmable 的 processor 可以應付快速演進的AI model
 - Fractile 要研發一個AI 加速器, 將自家的in-memory computing 技術與Andes 的 RISC-V vector processor 透過 ACE 整合起來 **寫到合作得 2 分**

3. (i) (k)

CPI: (average) cycle per instruction // SPECratio: ref t. / exec t.

定義 3 分, 每格計算 0.2 分。計算誤差 0.02內都正確

| Description | Name | Instruction Count x 10 ⁹ | CPI | Clock cycle time (seconds x 10 ⁻⁹) | Execution Time (seconds) | Reference Time (seconds) | SPECratio |
|--|-----------|-------------------------------------|------|--|--------------------------|--------------------------|-----------|
| Perl interpreter | perlbench | 2684 | 0.42 | 0.556 | 627 | 1774 | 2.83 |
| GNU C compiler | gcc | 2322 | 0.67 | 0.556 | 863 | 3976 | 4.61 |
| Route planning | mcf | 1786 | 1.22 | 0.556 | 1215 | 4721 | 3.89 |
| Discrete Event simulation - computer network | omnetpp | 1107 | 0.82 | 0.556 | 507 | 1630 | 3.21 |
| XML to HTML conversion via XSLT | xalancbmk | 1314 | 0.75 | 0.556 | 549 | 1417 | 2.58 |
| Video compression | x264 | 4488 | 0.32 | 0.556 | 813 | 1763 | 2.17 |
| Artificial Intelligence: alpha-beta tree search (Chess) | deepsjeng | 2216 | 0.57 | 0.556 | 698 | 1432 | 2.05 |
| Artificial Intelligence: Monte Carlo tree search (Go) | leela | 2236 | 0.79 | 0.556 | 987 | 1703 | 1.73 |
| Artificial Intelligence: recursive solution generator (Sudoku) | exchange2 | 6683 | 0.46 | 0.556 | 1718 | 2939 | 1.71 |
| General data compression | xz | 8533 | 1.32 | 0.556 | 6290 | 6182 | 0.98 |
| Geometric mean | — | — | — | — | — | — | 2.36 |

3.(j)

general data compression (xz)

因為資料壓縮需要大量的 讀/寫, 大量複雜的指令使得其 CPI 較高(instruction mix)

3分: 寫出最差的

2分: 寫出可能原因

3.(I)

為何使用幾何平均：老師上課提過，算術平均的話無法很好的反應整體情況，因為較大的 SPECratio 造成的影響太大。

$$GM = \sqrt[n]{\prod_{i=1}^n SPECratio_i}$$

3.(m)

pipeline stage 變兩倍(加長)使 clock frequency 變 3.6GHz(兩倍)

clock cycle time 變一半

CPI 可能會上升或不變 (dependancy)

execution time 會減少, 最多減半, 看 CPI 的漲幅

instruction count 不變

reference time 不變

SPECratio 最多增加為兩倍

評分標準:

E.T. / C.C.T / CPI / I.C. 全對 5 分

錯一項扣 1 分, 直至全錯。

沒寫出估算值一項扣 0.5 分

reference time/ SPECratio 寫錯會再扣 1 分

3.(n)

pipeline 變兩倍(加寬)

clock cycle time 不變

CPI 最多變一半, 端看 dependancy 狀況

execution time 會減少, 最多減半, 看 CPI 的漲幅

instruction count 不變

reference time 不變

SPECratio 最多增加為兩倍

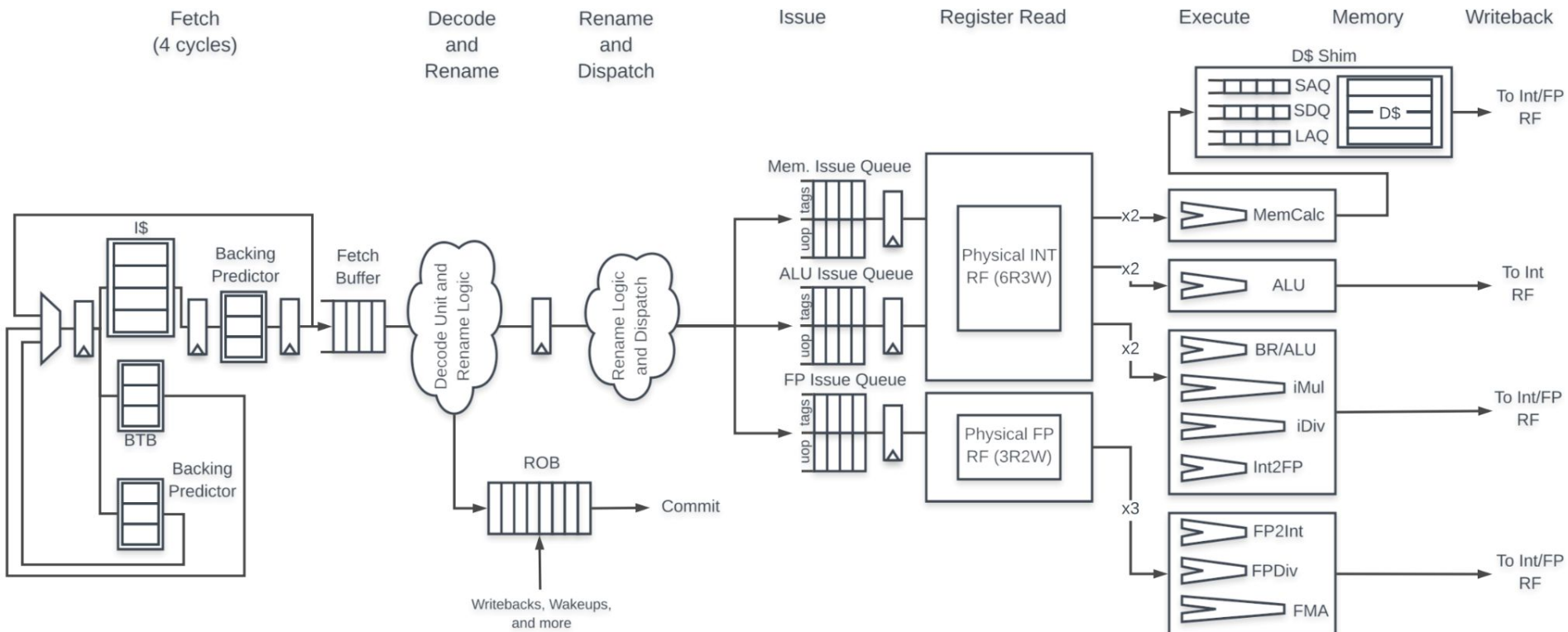
評分標準:

E.T. / C.C.T / CPI / I.C. 全對 5 分

錯一項扣 1 分, 直至全錯。

沒寫出估算值一項扣 0.5 分

reference time/ SPECratio 寫錯會再扣 1 分



4. (o)

- 同時可以有 4 個 operations 開始執行。分別是 MemCalc、ALU 、{BR/ALU, iMul, iDiv, Int2FP} 及 {FP2Int, FPDiv, FMA} 。
- 從圖中可以看到，Register Read 分別有四組線連接到這四個 block，因此在一個 cycle 內可以有 4 個 operations 開始執行。
- 補充：x2 代表最多有二個 operands 進到此 block, x3 代表最多有 3 個 operands 進到此 block。

評分標準(扣分制):

- -3: 未寫出 Operation 種類、或種類錯 2 個以上
- -2: Operation 種類錯 1 個
- -1: 解釋錯誤/未解釋原因
- -0.5: 其他小錯誤(額外扣分)
- -5: 空白/全錯/Operation 數量寫錯

4. (p)

- 分支預測在 Fetch 階段進行，可以在圖中看到 Backing Predictor 位於在 Fetch 階段。分支結果會在 Execute 階段知道，在分支預測錯誤時，分支後面錯誤執行的指令要 Flush。從 Fetch 到 Register Read 的指令都要 Flush，共會產生 8 個 bubble。 $(4+1+1+1+1)$

評分標準(扣分制):

- -0.5: 誤把 BTB 當成分支預測器
- -2: 寫錯或沒寫當預測錯誤時的處理方式 (若只是小錯誤則只 -1)
- -2: 寫錯或沒寫 bubble 數量 (只有寫 bubble 數量，或只有寫哪些階段要 flush 或產生 bubble，則只 -1)
- -5: 全錯、空白、未回答到問題

4. (q)

- Reservation Station 位於 Issue Stage 。圖中的 Issue Queue 為 Reservation Station 。Reservation Station 會讓那些 data dependency 已經被滿足的指令進到下一階段執行，而因為 data dependency 緣故還無法執行的指令會在 Reversation Station 等候。

評分標準 (加分制):

- +2.5: 有正確寫出 Reversation Station 在哪裡
- +2.5: 有正確解釋 Reversation 做什麼

(其他細部評分標準請看 Gradescope)

4. (r)

- 會有 read after write 的 data hazard 。可透過 forwarding 或指令重排、dynamic scheduling 等方式減少 bubble 。
- (因為是 out of order machine, 故也會有 WAW 或 WAR 的 data hazard , 可透過 register renaming 解決。在圖中已有此部份)

評分標準(加分制):

- +2.5: 有寫出會有 data hazard 及其種類 (若只寫有可能有 data hazard 則只 +1)
- +2.5: 有寫到怎麼減少 bubble
- 若上述全對, 但有小錯誤, 則這題得 4 分

4. (s)

- ROB 包含了所有在 pipeline 中 Decode and Rename 階段後正在執行的所有指令。
- exception 通常是在 execution 階段才知道的。當 exception 產生時，該指令對應於 ROB 的 entry 會被註記。因為指令會按原本的順序 commit，當要 commit 的指令是產生 exception 的指令時，會清除其他在 ROB 的指令及在 fetch 和 decode 階段的指令，並從 handler 所在的位置再開始 fetch 指令。也因為 ROB 是按照原本指令順序 commit，若有多個 exception 在 pipeline 中發生，也只有最早發生的會被 commit，其他的都會被捨棄而不 commit。(參考資料)
- (另外若 exception 是發生在 decode stage，例如 invalid Instruction。則就直接 flush 該 stage 之前的指令，並從 handler 所在位置開始 fetch。因為此 stage 之前都還是 inorder 執行。)

評分標準(扣分制):

- -1: 僅有些微錯誤
- -2.5: 答案部份有誤/只會答一部份/沒寫出如何處理需要捨棄的指令
- -4: 僅有回答到一小部份/一小部份正確