計算機結構 HW4 report

- What is the latency of each module in your design?

  以下是我用到的 module 的 latency

  - IF：355.81 ps
  - ID：546.84 ps
  - EX0：706.13 ps
  - EX1：648.63 ps
  - EX2：648.63 ps
  - EX3：624.43 ps

- Which path is the critical path of your cpu? And how can you decrease the latency of it?

```
ABC: Start-point = pi234 (\ex0.i_rs1_value_r [1]).  End-point = po244 ($techmap\ex0.$0\o_C[63:0] [15]).
ABC: + write_blif <abc-temp-dir>/output.blif
```

  我將 64 bit 的加法拆成 4 個 cycle 各做 16 bit 的加法，現在的 critical path 的 latency 主要來自 16 bit 的 adder，如果想要進一步減少 latency 可能需要將加法分散在更多 cycle 做，比方說每個 cycle 只做 8 bit 加法。

- How to solve data hazard?

  如果發現現在執行的指令會需要一個 已經進入 pipeline，但是還沒結束的指令 所算出來的結果時，會產生 data hazard。

  我的設計是 IF 在把從 instruction memory 讀到的 instruction 送出去之後，就把 pc + 4 送給 instruction memory，如果前一個指令還沒結束，就先將 instruction memory 送來的 instruction 存到一個 buffer 裡面，等收到上一個指令結束後發給 IF 的信號，再把 buffer 中的 instruction 送給下一個 stage (接著再送 pc + 8 給 instruction memory，以此類推)，這樣就可以避免 data hazard。這樣相當於從 instruction memory 讀到下一個 instruction 之後就開始 stall，直到前一個 instruction 結束。

- How to solve control hazard?

  承上，我都是先假設下一個要執行的指令是 pc + 4 (predict branch not taken)，然而如果位在 pc 的指令的計算結果是要 branch，而且我們在 pc 的指令結束前，就開始執行 pc + 4 的指令的話，可能會將錯誤的值寫到 register 或是 memory。

  所以我透過上題的設計，讓我的電路在收到下一個 instruction 之後就開始 stall，如果要 branch 的話就要放棄 buffer 裡的 instruction，重新送正確的 address 給 instruction memory，如果正確的話就一樣把 buffer 中的 instruction 送往下一個 stage。

- Describe 3 different workloads attributes, and which one can be improved tremendously by branch predictor ?

  - workload 1：100 個 iteration 的迴圈
  - workload 2：會在 4 個 label 之間跳 (J1 -> J3 -> J2 -> J4 -> 回到 J1)，共 500 個 iteration (所以會有大約 2000 個 branch)
  - workload 3：1000 個 iteration 的迴圈

  如果有 branch predictor 的話，可以減少因為 fetch 到錯誤的 instruction 而導致的多餘的 stall。

  - workload 1 的 iteration 次數不多，並且執行 branch 指令的次數相較於其他指令的比例並不高。
  - workload 2 的一個 iteration 就有 4 個 branch 指令，並且通常執行完兩個指令就會遇到一個 branch 指令，所以執行 branch 指令的次數相較於其他指令的比例較高。
  - workload 3 雖然 iteration 次數最多，但是他是在迴圈中執行一些其他指令之後在最後才 branch

  所以總體而言， workload 2 使用 branch predictor 效果會最好，因為它需要執行 branch 數量最多，需執行的 branch 指令的比例也最高。

- Is it always beneficial to insert multiple stage of pipeline in designs? How does it affect the latency?

pipeline 有很多個 stage 的話，就可以將工作量較大的 stage 的工作分散到其他 stage 中，減少 stage 的 latency。越多 stage 在最佳情況下也可以容納更多指令在 pipeline 中，提升 throughput。然而，因為同時執行的指令變多了，data hazard 以及 control hazard 可能會更容易發生，因而可能產生更多的 stall。此外 hazard 的處理也會變得更複雜，可能會需要因此設計更多電路處理 hazard，反而使 latency 上升。