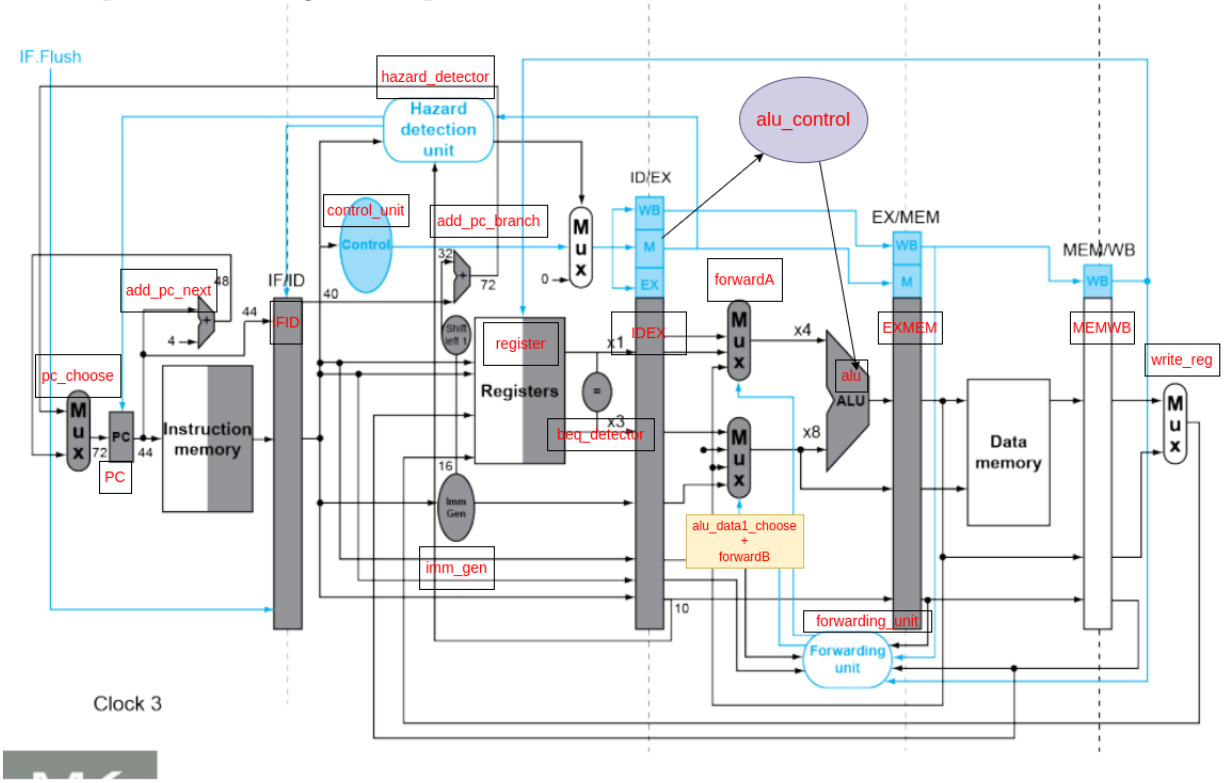
Computer Architecture report

b08902004 資工三 陳詩淇

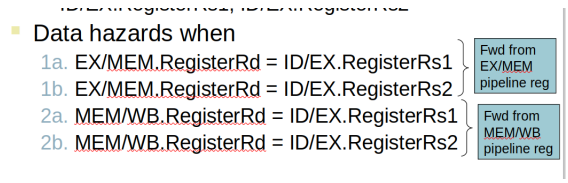
December 28, 2021

實做解釋

我大致是照著上課的 PPT 這張圖進行實做，紅字為我所有 module 的名稱，稍微不一樣的只有 alu 前那個下方的 mux，我是把他分開成兩個 mux，也就是 alu_data1_choose 跟 forwardB，資料會先進到 forwardB，他的 output 會再跟 imm_gen 的 output 一起進到 alu_data1_choose 去做選擇，然後才輸入給 alu。



關於其他細節，為了避免 data hazard，我根據 PPT 的下面兩條 rules 做了 forwarding_unit：



forwarding_unit 會發出 select 的訊號給 forwardA 跟 forwardB 這兩個 mux 去決定哪個資料會被餵進 alu 跟 EXMEM。

然後根據下面 PPT 的規則，我也嘗試寫了 hazard_detector 去避免 load hazard：

- Load-use hazard when
  - ID/EX.MemRead and
    ((ID/EX.RegisterRd = IF/ID.RegisterRs1) or
    (ID/EX.RegisterRd = IF/ID.RegisterRs1))
- If detected, stall and insert bubble

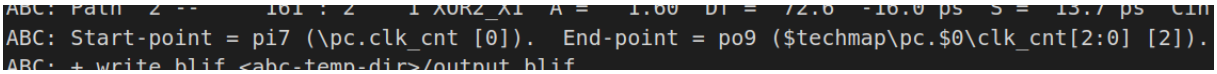
hazard_detector 的訊號會傳給 pc 跟 control_unit 去做 nop。

儘管我盡量做成 pipeline 的樣子，但我的 program counter 是寫成每 4 個 cycle 才能換下一個新的 instruction，如果調更少 cycle 的話會出事。 PC 每輸出一次 instruction address 給 instruction memory 後就會馬上讓 o_i_valid_addr 變成 0，確保後面幾個 cycle 是空的，以防之後同個 register 一直被改到。

而因為 data memory 需要兩個 cycle 才能完成讀取，所以我讓我的 MEMWB 也會要等兩個 cycle 才會把其他東西從 output 吐出來，而剛從 data memory 得到的資料就可以很剛好地跟其他東西一起出來，這樣餵給 register 去改資料時就會正確。

回答問題

1. Critical Path



這是我拿來計算 pc 讀了幾個 cycle 的 register。原本我是用 integer 下去計算，然後 critical path 顯示出來有 32 位元（如下圖），跟用 3 位元的 register 相比較長，所以我才改成用 register。

2. Data Hazard

用 forwarding unit 去檢查 EX\MEM 跟 MEM\WB 的 rd 是否跟 ID\EX 的 rs1 或 rs2 一樣，如果有就代表產生 hazard，需要將比較晚改到的資料 forward 到 EX 的階段。

3. Control Hazard

提早在 ID 的階段在 register 將兩個 register 的值讀出來後馬上判斷是否要執行這個指令的 branch，如果要的話就立刻把目前 ID 讀進來的指令給 flush 掉並讓下一個 IF 要抓的指令位置改成 branch 要跳的位置。

4. Workload Attributes

Workload 1 是一個 iterate 100 次、迴圈內部較多指令的程式。Workload 2 則是個雖然指令看起來較少，但實際上 iterate 次數較多且會不停在 4 個 segment （J1, J2, J3, J4） 中跳來跳去（workload 2 大部分的 branch 都是設定成一定會跳的）。Workload 3 則是個會 iterate 更多次的迴圈，雖然內部程式碼少，但 iterate 次數比 workload 1 多大概十倍。

Branch predictor 可以大幅提昇跑 workload 2 的效能。

5. Multiple Stage

否。插入太多 stage 在 pipeline 中可能會使接線過於複雜，並使計算複雜化，造成 latency 增加。