計算機結構 HW4 report　　B07902116　資工三　陳富春

Pipeline design :

大致架構和作業提供下圖參考的 CPU 架構相同，也是將 CPU 分成 IF、ID、EX、MEM、WB 五個階段做 pipeline，由於讀寫 data memory 最多需要 2 個 cycle，而讀取 instruction memory 需要 3 個 cycle 的時間，我設定讓每個 stage 有 4 個 cycle 的時間完成每個 stage 的工作，並將結果存到 stages 之間的 registers。

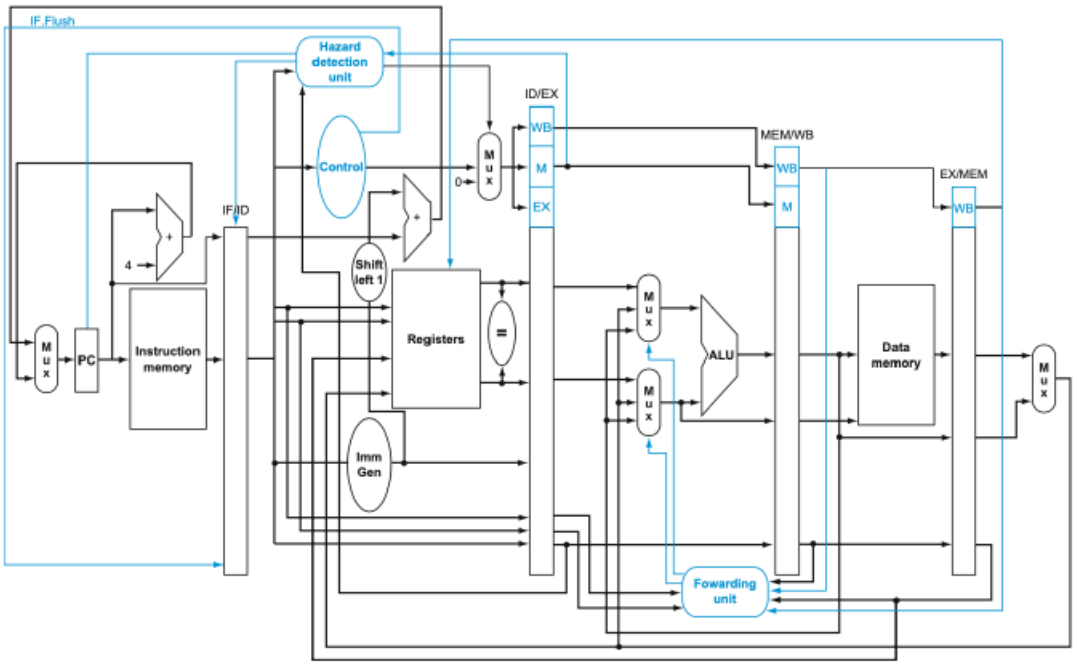

FIGURE 4.62 The final datapath and control for this chapter. Note that this is a stylized figure rather than a detailed datapath, so it's missing the ALUsrc Mux from Figure 4.55 and the multiplexor controls from Figure 4.49.

Critical path :

將 cpu.yslog 中的內容截圖得到

```
ABC: Start-point = pi17 (\i_i_inst [16]).  End-point = po0 ($abc$47967$auto$rtlil.cc:1832:Not$6624).
ABC: + write_blif <abc-temp-dir>/output.blif
```

如果要減少 latency 的話，使用 single cycle CPU 可以達到這件事，這麼做雖然能減少單一指令的 latency，但會使整體的 throughput 下降，因為一次只有一個 structure 在運作，相當於捨棄了 pipeline 能帶來的好處。

Data hazard :

對於一般的 data hazard，我做出一個 Forwarding Unit 來避免前一個指令的結果來不及寫回 register file 的情況。
具體來說 Forwarding Unit 會檢查現在進到 EX stage 的指令的 rs1 及 rs2，假如他們和 EX/MEM 或 MEM/WB 中的 rd 相同，則表示會發生 data hazard，此時則 Forwarding Unit 會輸出 EX_Forward1 EX_Foward2 來控制 multiplexor 選擇正確的資料來源。

而對於 load-use hazard，我做出 Hazard Detection Unit 來檢查是否發生 load-use 的情況。
具體來說 Hazard Detection Unit 會藉由檢查 ID/EX control signal 中的 MemRead 來確認現在進到 EX stage 的指令是否為 ld 指令，如果是的話，繼續檢查這個 ld 指令和現在進到 ID stage 的 rs1 rs2 是否相同，萬一相同的話，則表示發生 load-use hazard，此時必須不讓在 ID stage 內的指令繼續往後執行，因此先讓 control unit 輸出全為 0 的訊號，然後再避免 PC 和 IF/ID registers 被覆寫，達到相當於 insert bubble 的效果，這樣便解決了 load-use hazard 的問題。

Control hazard :

當 branch 指令進入 ID stage 時，PCsource 會檢查是否滿足 branch 的條件，如果滿足的話，便會調整輸出的控制訊號，讓 PC_source_mux 選擇輸出的下一個位置是 branch 後的位置，而不是原本的 PC+4。

Workload attributes :

T8 的迴圈次數是三者裡面最小的，但同時每次迴圈要執行的指令數也是最多的
T9 的迴圈次數和每次迴圈執行的指令數都居中，但每次迴圈都要在 4 個區域中做 branch
T10 的迴圈次數是最多的，但每次迴圈要執行的指令數是最少的

如果能做出正確的 branch predict，那麼 T9 的執行時間應該會有顯著的下降，因為在每次的迴圈中，T9 的指令還需要額外判斷 4 次是否要 branch 到下一個區域，即使總迴圈次數比 T10 少，但處理的 branch 指令數目卻比 T10 多，因此若有 branch predictor 的話，T9 的執行時間會下降最多

```
C:\Users\wryyy\Downloads\hw4>iverilog -D T8_small -f cpu.f

C:\Users\wryyy\Downloads\hw4>vvp ./a.out
VCD info: dumpfile cpu.vcd opened for output.
Small Workload1 test
Check memory
Correct!
Cycle count:        1615

C:\Users\wryyy\Downloads\hw4>iverilog -D T9_small -f cpu.f

C:\Users\wryyy\Downloads\hw4>vvp ./a.out
VCD info: dumpfile cpu.vcd opened for output.
Small Workload2 test
Check memory
Correct!
Cycle count:        1343

C:\Users\wryyy\Downloads\hw4>iverilog -D T10_small -f cpu.f

C:\Users\wryyy\Downloads\hw4>vvp ./a.out
VCD info: dumpfile cpu.vcd opened for output.
Small Workload3 test
Check memory
Correct!
Cycle count:         675
```

Insert multiple stage of pipeline :

當越來越多的 stage 被加入 pipeline 的設計當中，雖然可能能提升整體的 throughput，但對單一指令的 latency 有可能會越來越久，因為 cycle time 是根據需要最久執行時間的 stage 決定的，所以如果要增加 stage，最好是能將原本執行最久的 stage 拆成多個較短執行時間的 stage，才能提高效率；反之，如果 pipeline 中的 stage 增加，cycle time 卻維持相同的話，同樣的指令反而要花更久時間才能執行完，對 throughput 及 latency 都有負面影響。