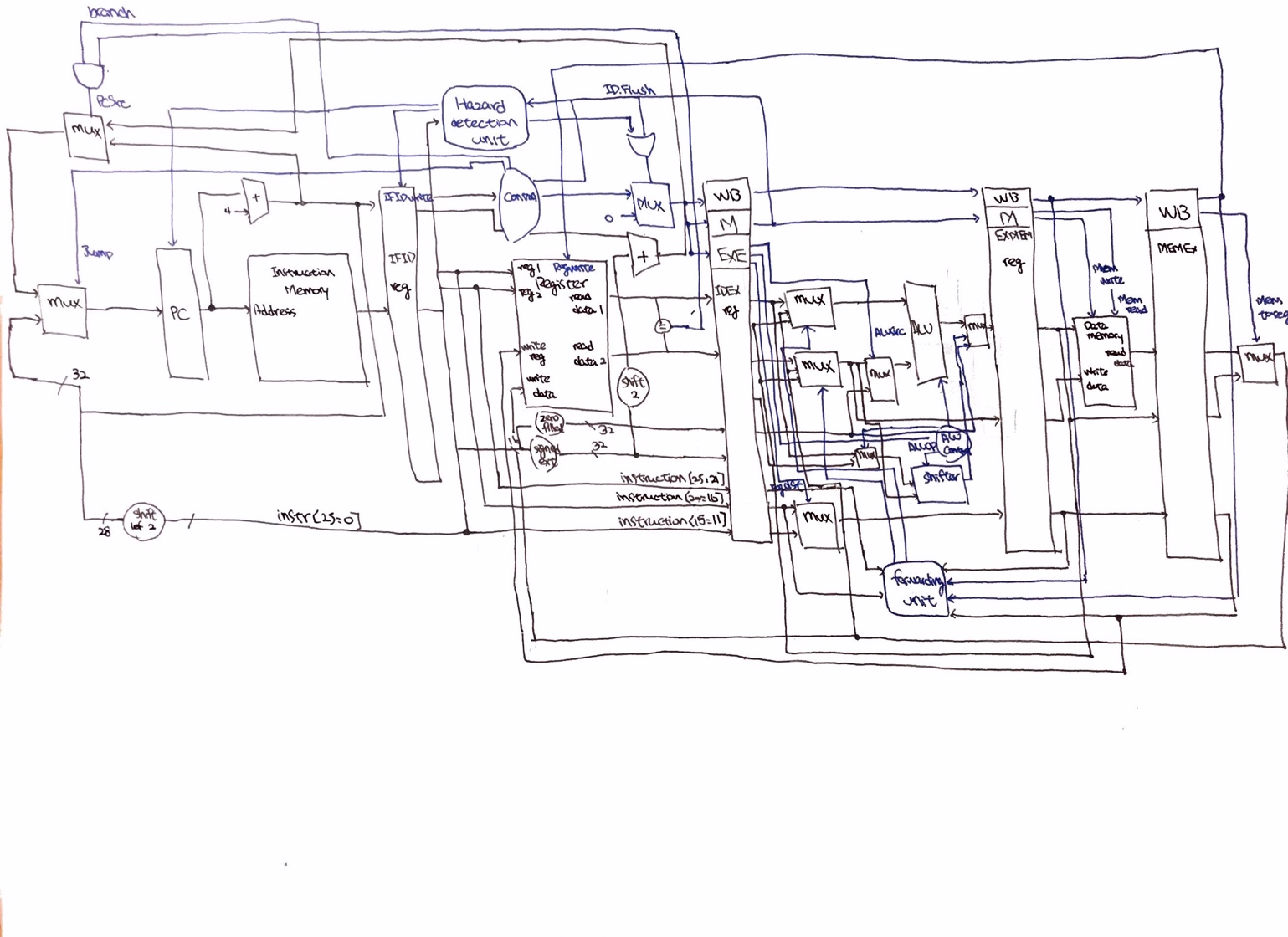
**Computer Organization**

**HDL simulator you used:**

**ModelSim**

**Finished part:**

**Adder.v , ALU.v , ALU\_Ctrl.v , Decoder.v , Mux3to1.v , Mux2to1.v , Pipeline\_CPU.v , Shifter.v , Sign\_Extend.v , Zero\_Filled.v, Data\_Memory.v, Reg\_File.v, Forwarding.v, HazardDetectionUnit.v, Pipeline\_Reg.v**

**Architecture diagrams:**

**Hardware module analysis:**

**PC: Program counter, 一開始先從initial值開始跑，接著其值是由其前面的mux來決定是否為pc+4, branch address還是jump address。**

**Instruction Memory: 用來記憶cpu指令，測資會從此讀入cpu。**

**IFID reg: 進行存放pipleline cpu這一輪stage產生的結果，並且將其結果等待下一輪傳給下一個stage，之後的IDEXE, EXEMEM, MEMWB也是一樣的做法。**

**Register File:模擬register的讀出跟寫入，從IFID拿兩個reg的位置，並且接受從後面傳回來的write reg 和write data。**

**Hazard Detection Unit: 從IDIF IDEXE的M拿值並且進行判斷是否有需要讓此pipeline cpu 執行stall的工作。**

**Forward Unit: 從MEM, WB stage拿值與EXE的RS RT做比對，如果有dependent的狀況，就要做forwarding，把已經算好的新的RS RT即時回傳回去給EXE做運算。**

**Problems you met and solutions:**

**一開始對於Hazard detection感到十分害怕，因為前面的設計沒有考慮到這一部分的電路，因次再放入Hazard detection unit的時候，很多地方都需要進行些微的小調動，然而在前面做了非常精密的電路之下，牽一髮則動全身。因此在改動的期間遇到非常多小BUG，所幸最後在一個一個module check之後，把每一條電路都確認好，就成功地把Hazard detection unit與Forward Unit實作出來。**

**Summary:**

**在這個最後一個project中，稍微回顧自己之前做的cpu，從原本最簡單麼simple cycle cpu，到現在如此精密的pipeline架構，真的感覺十分有成就感。在這一次一次的作業中，把講義上的cpu架構概念釐清，一步一步的把功能改善、優化，都是一個里程碑，最後在期末把最棘手的hazard detection也放上去了，實在倍感欣慰，感謝助教這學期認真地出作業辛苦了，暑假愉快！**