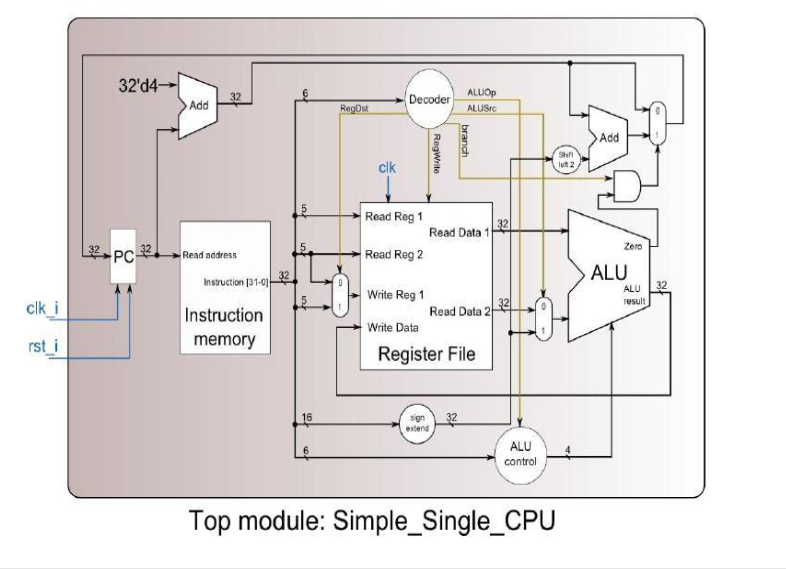
Computer Organization

Architecture diagrams:



Hardware module analysis:

PC :

控制本次輸入的instruction的位置，一種情況是上一次的instruction+4，另一種是根據branch所跳的位置來讀取insruction。

Adder unit:

本次作業中有兩個Adder，第一個是PC+4，第二個是PC+4 + Shift left 的output。

* PC+4:將目前的instruction位置加上4，成為下個instruction位置的選擇之一。
* PC+4 + Shift left :將PC+4 加上的Shift left 的output，成為下個instruction位置的選擇之一。

IM:

根據PC傳送的位置尋找instruction code，並將instruction code輸出到後續的Unit中

Mux Unit:

本次作業中有三個Mux unit，第一個是Reg\_dst，第二個是Reg\_src，第三個是PC\_source。

* Reg\_dst:控制register file RDaddr的輸入，控制指令是Decoder的RegDst，如果RegDst=0，output為instr\_out[20:16]

如果RegDst=1，output為instr\_out[15:11]

* Reg\_src:控制ALU src2的輸入，控制指令是Decoder的RegSrc。

如果RegSrc=0，output為RT data output

如果RegSrc=1，output為Sign extention output

* PC\_source:控制下一個instruction的位置，控制指令是and(ALU zero,Branch)，PC+4 + Shift left

如果and=0，output是PC+4

如果and=1，output是 PC+4 + Shift left output

Decoder:

根據讀入的Op code，判斷出各指令，並根據目前指令控制RF、ALU control unit、Mux unit以及branch control。

* R-type:

RegWrite=1

ALU\_op =010

ALUSrc =0

RegDst =1

Branch =0

* Addi:

RegWrite=1

ALU\_op =011

ALUSrc =1

RegDst =0

Branch =0

* Slti:

RegWrite=1

ALU\_op =100

ALUSrc =1

RegDst =0

Branch =0

* Beq

RegWrite=0

ALU\_op =001

ALUSrc =0

RegDst =0

Branch =1

RF:

暫存instruction中的各個register的值，輸出RS和RT的data，並根據Reg\_write來判斷是否寫入ALU result到 RD。

SE unit:

將instr[15:0]共16bit的data extend成32bit的data，而其數值並未改變。

Shift left unit

將SE output 往左shift 2bit，作為往後adder的input之一。

ALU control unit:

根據Decoder的 ALU\_op及funct\_code來判斷該指令，並輸入相對應的ALU\_control到ALU進行運算。

* ALU\_op=010(R-type)

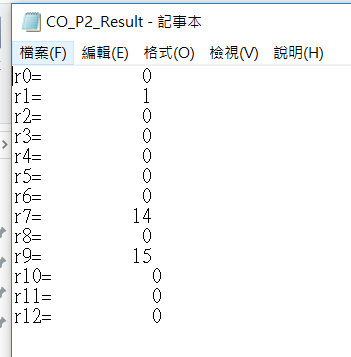
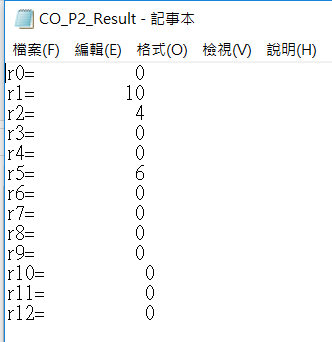
1. Funct\_code=100000(add): ALU\_control=0010
2. Funct\_code=100010(sub): ALU\_control=0110
3. Funct\_code=100100(and): ALU\_control=0000
4. Funct\_code=100001(or): ALU\_control=0001
5. Funct\_code=101010(slt): ALU\_control=0111

* ALU\_op=011(addi): ALU\_control=0010
* ALU\_op=100(slti): ALU\_control=0111
* ALU\_op=001(beq):ALU\_control=0110

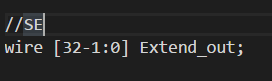
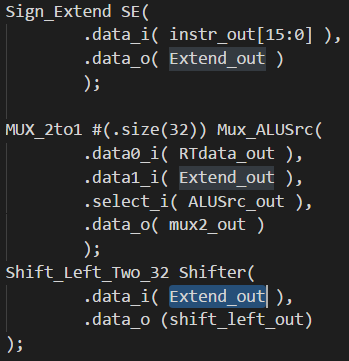
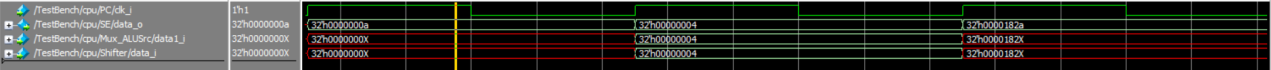
ALU:

根據從ALU control unit輸入的ALU\_control，將src1和src2進行運算，並輸出zero及result。

Finished part:



Problems you met and solutions:



確定 SE data\_o 、Mux\_ALUSrc data1\_i 和Shifter data\_i使用同一個wire的情況下，SE data\_o的資料傳送到 Mux 和 Shifter 後發生錯誤。

在Shifter的檔案中發現變數assign的位置相反了，進而改變Extend\_out的數值，因此造成這種錯誤

Summary:

這次的lab相較於lab1而言困難了許多，要編輯很多.v檔，因此debug時十分困難，要循著datapath一路尋找錯誤，因此我學會了使用波形圖來同時檢視多筆資料，在各筆資料的比較中，更容易找到錯誤地的方。