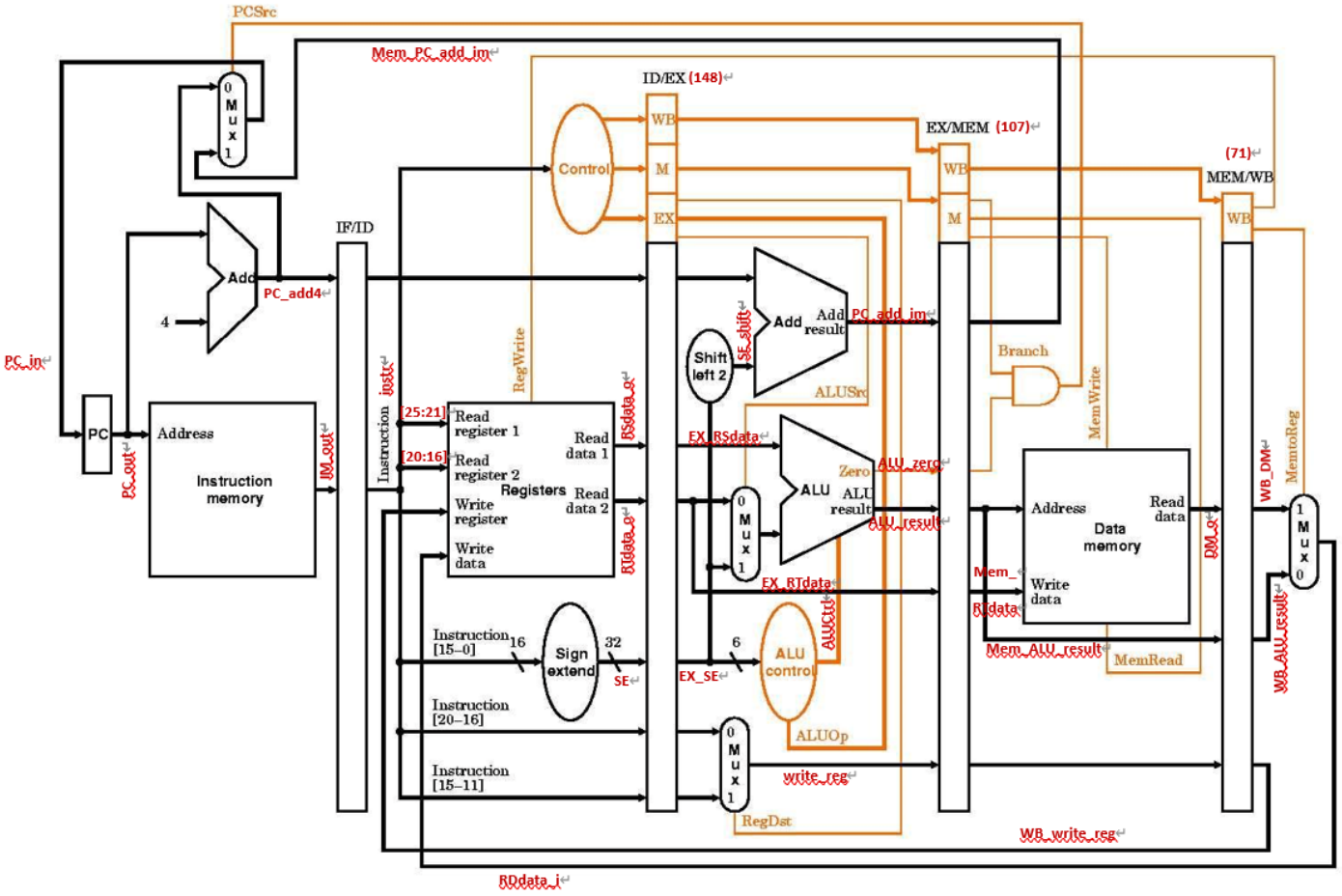
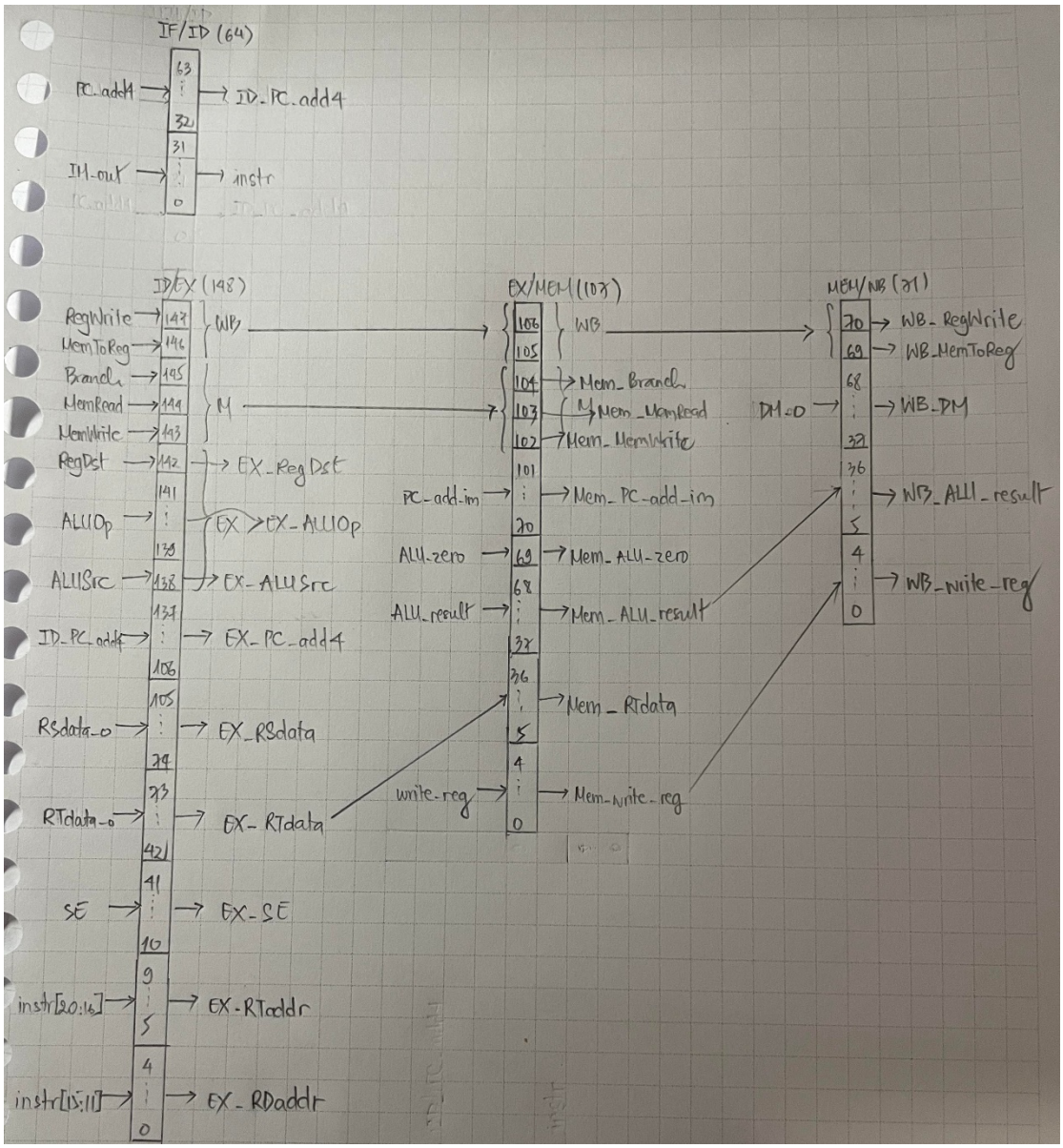
Computer Organization Lab4

Name: 郭大寧

ID: 109550184

Architecture diagrams:





Hardware module analysis:

(explain how the design work and its pros and cons)

一個指令需要實作 5 個步驟：IF、ID、EX、MEM、WB。使用 4 個 stall 夾在 5 步驟之間，stall 的任務需要儲存前步驟的值給下一個步驟使用。利用 stall，可以同時實作多指令。

使用 Instruction memory 來取得來取得 instruction。使用 Registers 來決定 rs 和 rt。使用 decoder 來取得 control signals 來控制系統。control signals 分成 3 組，對應在 3 個步驟需使用。

EX：RegDst、ALUOp、ALUSrc

Mem：Branch、MemRead、MemWrite

WB：RegWrite、MemToReg。

ALU 部分，針對 add、addi、sub、and、or、slt、slti、lw、sw、beq、mult 取得

operation。分成 6 種：and，or，add、addi、lw、sw，sub、beq，slt、slti，mult 來計算。

Data Memory 部分來處理 lw 和 sw，利用對應的 stall 來取得 MemWrite 和 MemRead，來分別需要執行的指令。

Pros:

跟 single cycle cpu 相比，pipelined cpu 可以同時實作多指令，不需要等前指令結束。增加處理速度。

Cons:

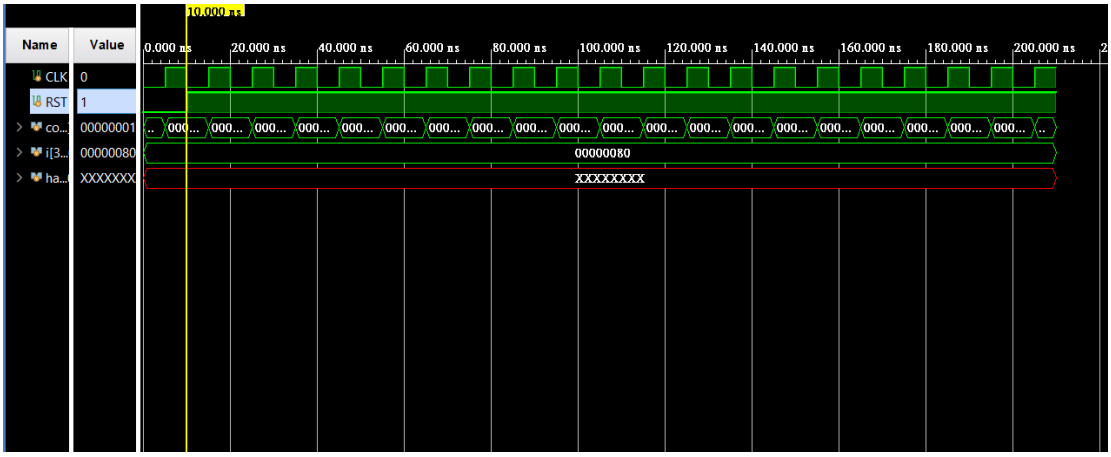
需要 stall 來儲存每個步驟，需要開報比較多，比較大的空間來儲存。

沒處理 data hazards，forwarding 部分，變數值來不及計算導致連著的指令需要使用的值是錯誤，導致答案錯誤。

Finished part:

> **(show the screenshot of the simulation result and waveform, and explain it)**

```
Register=============================================================

r0=         0, r1=         3, r2=         4, r3=         1, r4=         6, r5=         2, r6=         7, r7=         1

r8=         1, r9=         0, r10=        3, r11=        0, r12=        0, r13=        0, r14=        0, r15=        0

r16=        0, r17=        0, r18=        0, r19=        0, r20=        0, r21=        0, r22=        0, r23=        0

r24=        0, r25=        0, r26=        0, r27=        0, r28=        0, r29=        0, r30=        0, r31=        0


Memory=============================================================

m0=         0, m1=         3, m2=         0, m3=         0, m4=         0, m5=         0, m6=         0, m7=         0

m8=         0, m9=         0, m10=        0, m11=        0, m12=        0, m13=        0, m14=        0, m15=        0

r16=        0, m17=        0, m18=        0, m19=        0, m20=        0, m21=        0, m22=        0, m23=        0

m24=        0, m25=        0, m26=        0, m27=        0, m28=        0, m29=        0, m30=        0, m31=        0
```



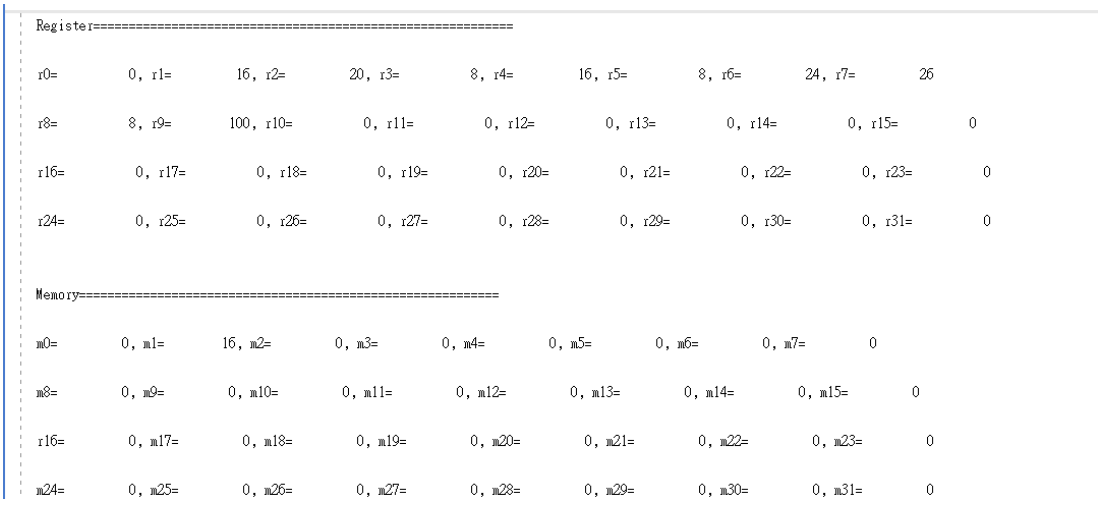
每一個指令實作完成 5 個步驟會儲存到值，pipelined 可以接著實作，不需要等前指令完成。

```
addi $1,$0,3;       6
addi $2,$0,4;       7
addi $3,$0,1;       8
sw $1,4($0);        9
add $4,$1,$1;       10
or $6,$1,$2;        11
and $7,$1,$3;       12
sub $5,$4,$2;       13
slt $8,$1,$2;       14
beq $1,$2,begin     15
lw $10,4($0);       16
```

Problems you met and solutions:

Lab4 使用 Lab3 的資料但有些地方細節不統一，導致程式無法運作，花費時間去找。

Bonus (optional):



如下指令需要前指令的變數需要增加 2 次 nop 來給前指令完成計算變數才能傳給下指令使用。在 Test2 有 3 組，I1/I2、I5/I6、I8/I9，需要在各組中間增加 2 個 nop。總共需要增加 6 個。

```
I1: addi $1,$0,16       6
I2: addi $2,$1,4        9
I3: addi $3,$0,8        10
I4: sw $1,4($0)         11
I5: lw $4,4($0)         12
```

I6: sub $5,$4,$3 15
I7: add $6,$3,$1 16
I8: addi $7,$1,10 17
I9: and $8,$7,$3 20
I10: addi $9,$0,100 21

Summary:

實作 pipelined cpu，利用 single cycle cpu 加上使用 stall 來運作，讓指令可以同時做，減少等待執行指令的時間，讓工作可以早完成。這次 pipedlined cpu 針對 add、addi、sub、and、or、slt、slti、lw、sw、beq、mult 來計算。沒有處理 data hazards 和 forwarding 問題，需手動增加 nop 前指令計算變數，下個指令才能使用。