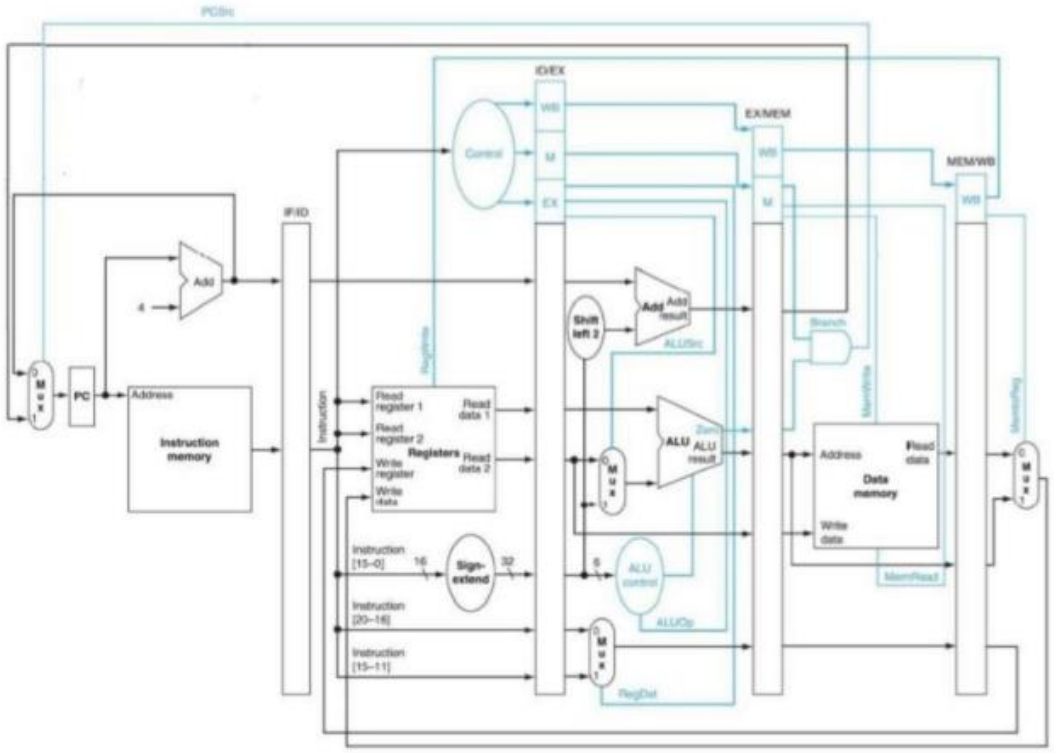
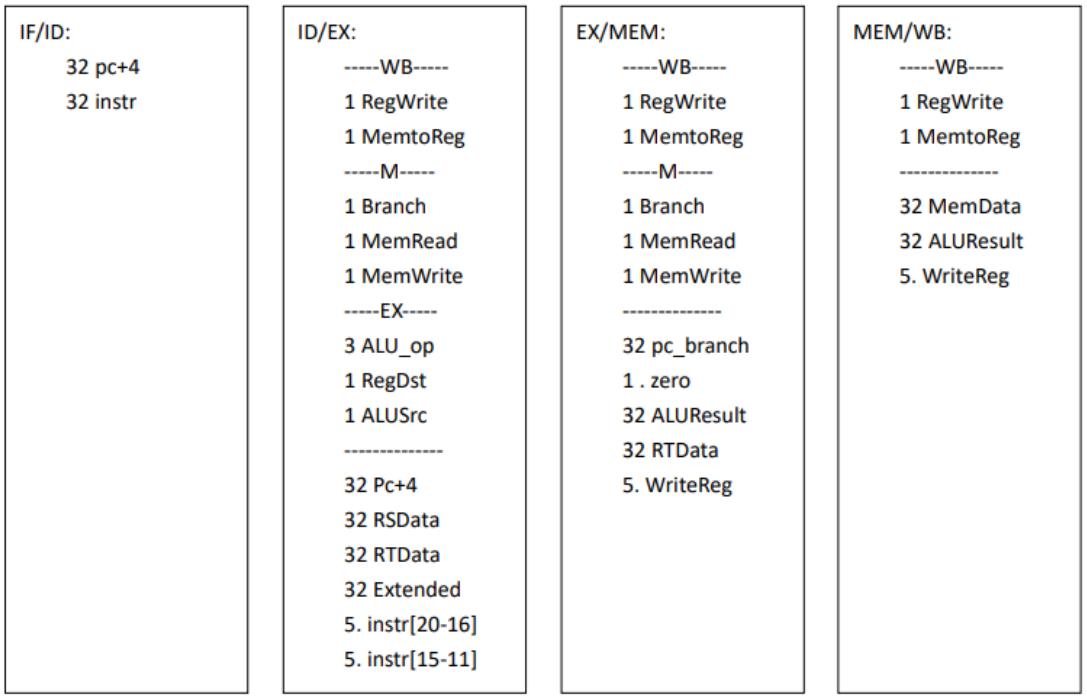
Computer Organization Lab4

Architecture diagrams:

CPU: 取自 CO_Lab_4.pdf



Pipeline Register: 數字為 bit 數



Hardware module analysis:

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

和 single-cycle CPU 相比，因為多加了 pipeline register，每個 stage 可以被分配給不同的指令，所以同時可以有多個指令在進行，不會浪費空閒的 hardware。
每個 pipeline register 中，儲存該指令在下一個 stage 所需要的 control signal 和 data

IF: 讀取當前 PC Addr 的指令，因為指令有可能是 branch 所以需要儲存 pc+4
ID: 利用 IF/ID 儲存的指令，得到 RSData、RTData 和 control signal 等
EX: 利用 ID/EX 中的 control signal 和 Data 做 ALU 的計算，一併算出 Branch addr、得到 RD Addr (WriteReg)
MEM: 利用 EX/MEM 儲存的 RSData (Read Addr)和讀取出的 RTData，再根據 control signal 決定是否要讀/修改 memory。
且在此時確定 branch 的結果，並將計算的結果連結至 Program Counter
WB: 根據 MEM/WB 儲存的 control signal 決定是否需要 Write to Reg (Memory data / ALU result)

每個 stage 都各自做不同的指令，有效利用資源，但有可能不同指令之間有 data dependency，就會造成 hazard，則需要更複雜的機制(forwarding、stall 等)去控制，但整體而言效率是比 single-cycle CPU 好很多的。

Finished part:

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

Test1: 和助教給的參考答案是相同的

```
begin:
addi        $1,$0,3;          // a = 3
addi        $2,$0,4;          // b = 4
addi        $3,$0,1;          // c = 1
sw          $1,4($0);         // A[1] = 3
add         $4,$1,$1;         // $4 = 2*a
or          $6,$1,$2;         // e = a |  b
and         $7,$1,$3;         // f = a &  c
sub         $5,$4,$2;         // d = 2*a  - b
slt         $8,$1,$2;         // g = a <  b
beq         $1,$2,begin
lw          $10,4($0);        // i = A[1]
```

```
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
```

Problems you met and solutions:

除了在 Pipeline Reg 的設計上想了比較久的時間，基本上都是沿用 single-cycle CPU 的設計，沒有做太多更動，因此沒有遇到太多困難。

Bonus (optional):

Modified machine code:

00100000000000010000000000010000	I1: addi $1, $0, 16
00100000000010010000000001100100	I10: addi $9, $0, 100
00100000000000110000000000001000	I3: addi $3, $0, 8
00100000001000100000000000000100	I2: addi $2, $1, 4
10101100000000010000000000000100	I4: sw $1,4($0)
10001100000001000000000000000100	I5: lw $4, 4($0)
00100000001001110000000000001010	I8: addi $7, $1, 10
00000000011000010011000000100000	I7: add $6, $3, $1
00000000100000110010100000100010	I6: sub $5, $4, $3
00000000111000110100000000100100	I9: and $8, $7, $3

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

r0=         0, r1=         16, r2=         20, r3=         8, r4=         16, r5=         8, r6=         24, r7=         26

r8=         8, r9=         100, r10=         0, 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=         16, 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
```

Summary:

這次的 lab 讓我對 pipeline CPU 有更多的理解，bonus 的部分也讓我發現自己思考的盲點。