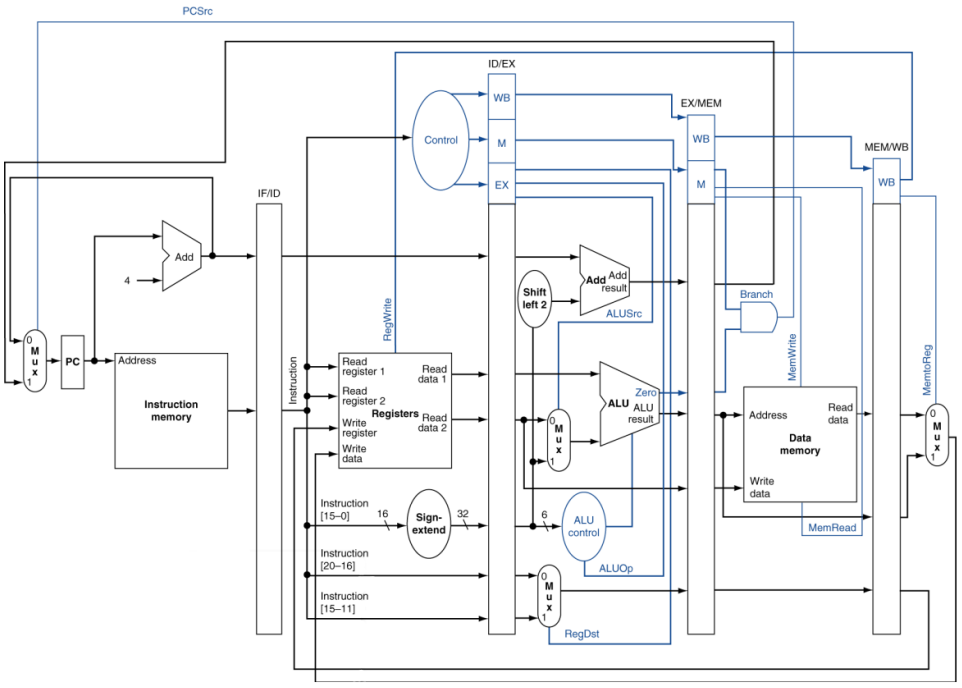
Computer Organization Lab4

Name: 吳文心

ID: 109550022

Architecture diagrams:



Hardware module analysis:

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

在 ALU 和 ALU_Ctrl 中多加了關於 MULT 的運算。

因為實做 pipeline CPU 須在各個 stage 之間加上 register 來存取各 stage 的 instruction 的 state，由 Lab_3 的 Simple CPU 出發，照電路圖將各 stage 之間的 register 定好，再接上所有元件之間相連的電路即可完成這次作業。

使用 IF_ID, ID_EX,EX_MEM 及 MEM_WB 四個 register 分別作為 stage 之間的暫存器。

除上所述，本次作業的架構基本都和 Lab3 一樣。

Finished part:

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

test 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
```

begin: addi $1,$0,3; // ($1) = 3

addi $2,$0,4; // ($2) = 4

addi $3,$0,1; // ($3) = 1

sw $1,4($0); // m1 = 3

add $4,$1,$1; // ($4) = 2*($1) = 2*3 = 6

or $6,$1,$2; // ($6)= ($1) | ($2) = 011_2 or 100_2 = 111_2 = 7

and $7,$1,$3; // ($7) = ($1) & ($3) = 011_2 and 001_2 = 001_2 = 1

sub $5,$4,$2; // ($5) = ($4) – ($2) = 6 – 4 = 2

slt $8,$1,$2; // ($8)= ($1) < ($2) = True = 1

beq $1,$2,begin // if (($1) == ($2)) go to “begin”，因為兩 register 中的值分別為 3 和 4 => ($1) != ($2)，所以並沒有 branch

lw $10,4($0); // ($10) = m1 = 3

Problems you met and solutions:

經過前幾次 Lab 的訓練，這次 Lab 很快就完成了，只是接線的部分要特別小心，時間主要花在檢查哪條線接錯(就是檢查應相連的電路元件是否有一樣的變數)。

Bonus (optional):

```
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
```

I1: 00100000000000010000000000010000
I3: 00100000000000110000000000001000
I10: 00100000000010010000000001100100
I2: 00100000001000100000000000000100
I4: 10101100000000010000000000000100
I8: 00100000010011100000000000001010
I5: 10001100000001000000000000000100
I7: 00000000011000010011000000100000
I9: 00000000111000110100000000100100
I6: 00000000100000110010100000100010

For I1/I2: 將 I3 和 I10 移到 I1、I2 之間 => 相差兩個 instructions
For I5/I6: 將 I7 和 I9 移到 I5、I6 之間 => 相差兩個 instructions
For I8/I9: 將 I8 移到 I5 之前 => 相差兩個 instructions

Summary:

這四次作業下來，只能說由衷敬佩寫硬體語言的工程師們，他們真的太強了，邏輯超清晰！