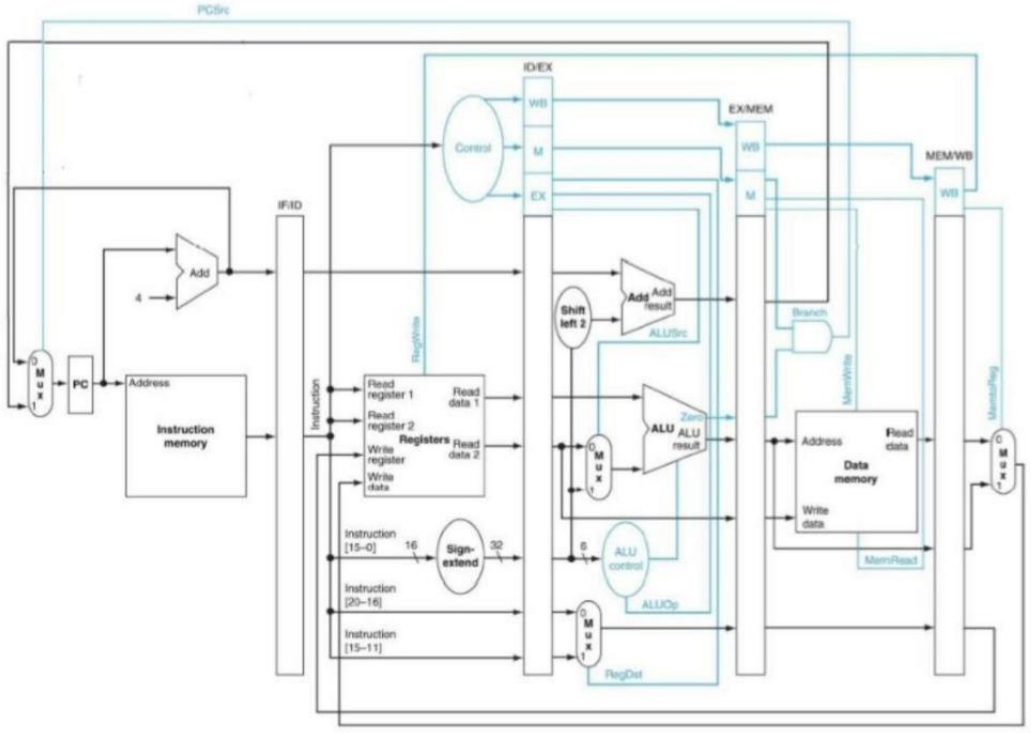
Computer Organization Lab4

Name: 陳宥安

ID: 109550073

Architecture diagrams:



基本上我就是按照pdf上的電路圖照著刻，在下個部份會稍微敘述一下每個元件部份在幹嘛。

Hardware module analysis:

以下針對各個檔案（元件）做簡單說明：

Adder1, Adder2: 就是兩個加法器（a+b）。

MUX_2to1*4: 就是個選擇器。

Program_Counter: 存答案的地方counter。

Instruction_Memory: 讀取並解讀MIPS指令。

Data_Memory: 暫時存一些資料的地方。

Decoder: 判斷MIPS是哪種format、判定要做何種instruction，並decode出一些訊號供續使用。

Reg_File: 靠Decoder傳入判斷instruction是哪種format來決定ALU的輸入值。

Sign_Extend: 如果instruction為i-format要做extension。

ALU: 就是個ALU， 加減和邏輯運算， 跟上個Lab一樣。

ALU_Ctrl: 控制ALU的控制訊號。

Shift_Left_Two: 就是個shifter， 向右shift兩位。

Pipe_Reg: Register for儲存在不同階段的指令。

Finished part:

```
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
$stop called at time : 210 ns : File "/home/iammrchen/Desktop/CO_LAB3/CO_LAB3.srcs/sim_1/imports/lab4_code/TestBench.v" Line 55
INFO: [USF-XSim-96] XSim completed. Design snapshot 'TestBench_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:11 ; elapsed = 00:00:06 . Memory (MB): peak = 7804.684 ; gain = 71.996 ; free physical = 6698 ; free virtual = 19153
```

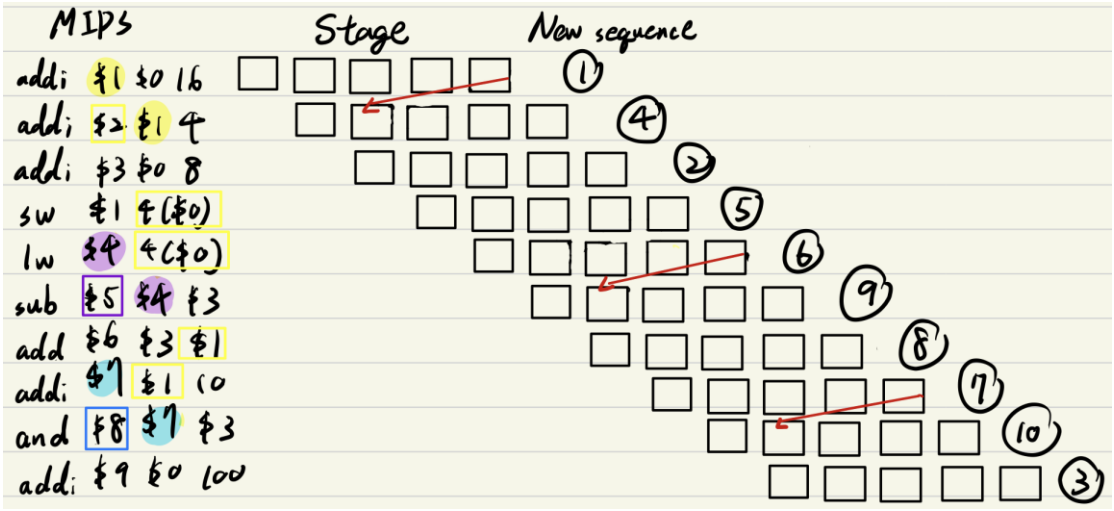
如圖， 輸出正確， MIPS的部份我會在summary作解釋。

Problems you met and solutions:

接線似乎是我的罩門， 在Pipe_CPU_1裡面， 我第一次完成時跑simulation竟然沒有結果， 結果是線的大小寫用錯（手殘）， 害我找了快一個小時==。另外這次import machine code變成在testbench裡， 不知為何那個path一定要用絕對路徑我才讀的到， 或許是系統在搞吧.....。

Bonus (optional):

此次bouns是要透過修改Machine code的部份來解決hazard的問題， 以下我以「發生hazard， 依序把後面不受影響的指令往前搬來執行」這項原則提出我的解法：



除了依照原則以外，由於原本I5/I6、I8/I9都受到I1/I2 hazard的影響，所以才先把I10拉上來做，才跑I4的sw跟後續的指令。再來，I5之後緊接著跑I8是因為如果先照順序跑I7的話，最後I8/I9的data hazard會需要加上nop才能解決hazard，所以I5後才先跑I8。最後，修改完畢後的machine code如下（順序的話上面的圖有寫）：

```
 1 00100000000000010000000000010000
 2 00100000000000110000000000001000
 3 00100000000010010000000001100100
 4 00100000001000100000000000000100
 5 10101100000000010000000000000100
 6 10001100000001000000000000000100
 7 00100000001001110000000000001010
 8 00000000011000010011000000100000
 9 00000000100000110010100000100010
10 00000000111000110100000000100100
```

接下來的兩張截圖是修改前後執行的結果：

a. 修改前：

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

r0=          0, r1=         16, r2=          4, r3=          8, r4=         16, r5=         -8, r6=         24, r7=         26

r8=          0, 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
$stop called at time : 210 ns : File "/home/iammrchen/Desktop/CO_LAB3/CO_LAB3.srcs/sim_1/imports/lab4_code/TestBench.v" Line 55
INFO: [USF-XSim-96] XSim completed. Design snapshot 'TestBench_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
```

b. 修改後：

```
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
$stop called at time : 210 ns : File "/home/iammrchen/Desktop/CO_LAB3/CO_LAB3.srcs/sim_1/imports/lab4_code/TestBench.v" Line 55
INFO: [USF-XSim-96] XSim completed. Design snapshot 'TestBench_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
```

修改後的結果才符合原本code的運算結果（CO_P4_test_2_result.txt），故我們成功的解決了hazard。

Summary:

我們來分析兩份Machine code的MIPS:

a. CO_P4_test_1.txt

```
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]
```

基本上就是把spec提到的所有功能都試跑一次並存在相對應的register和memory裡，每一行指令的意思一開始註解就附了，所以不再多贅述，而beq那行由於s1!=s2所以就沒有跳到begin，因為沒有hazard的問題所以不需要調換指令順序，最後執行的結果跟CO_P4_test_1_result.txt 一樣，所以所有功能都正常。

b. CO_P4_test_2.txt

```
I1:    addi    $1,$0,16
I2:    addi    $2,$1,4
I3:    addi    $3,$0,8
I4:    sw      $1,4($0)
I5:    lw      $4,4($0)
I6:    sub     $5,$4,$3
I7:    add     $6,$3,$1
I8:    addi    $7,$1,10
I9:    and     $8,$7,$3
I10:   addi    $9,$0,100
```

這個測資加入了hazard的部份，其他指令都是第1份測資就測試過的。所以如果第1份測資答案對的話就代表功能沒有問題，所以只要處理hazard的部份（bouns有解釋）即可，如果直接照原本的順序執行會有錯誤的結果，而改變順序後執行的結果跟CO_P4_test_2_result.txt 一樣，所以成功解決hazard。

最後，這個lab讓我們實作了pipeline CPU和解決hazard，其實蠻好玩的，把整個第四章做完了的感覺，不過還是希望這是最後一個lab啦，我想放假.....。