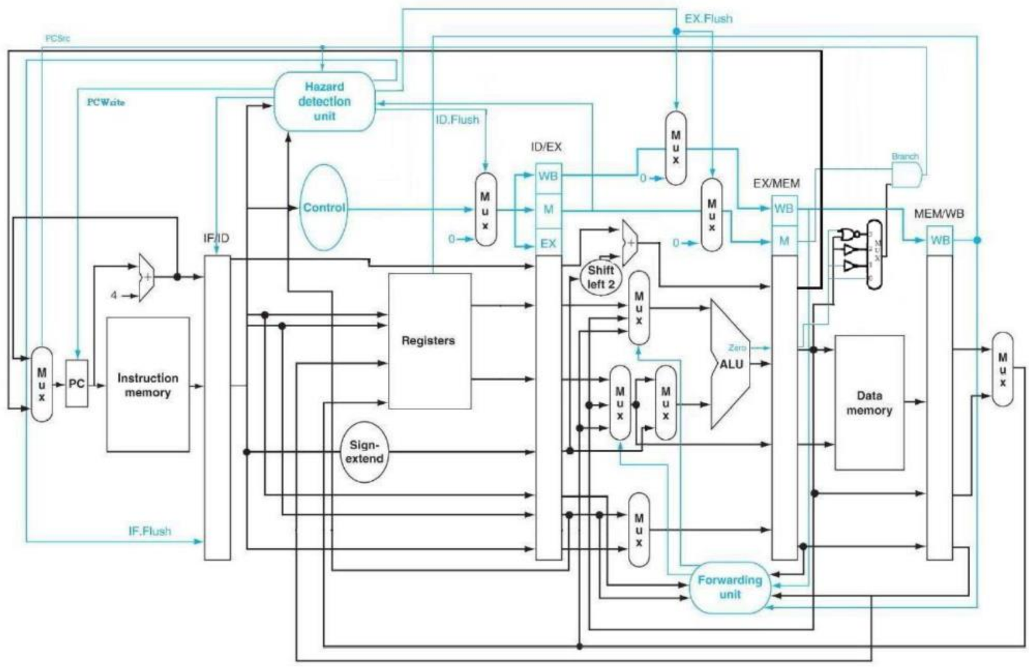
Computer Organization Lab5

Name: 吳文心

ID: 109550022

Architecture diagrams:



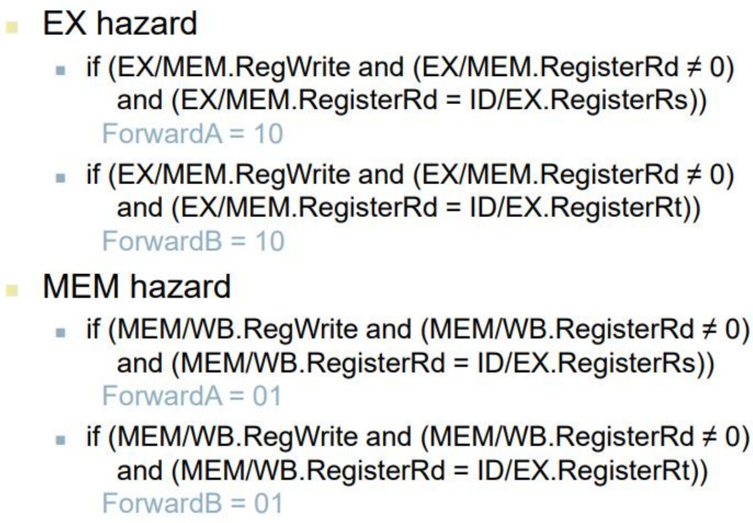
Hardware module analysis:

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

Lab 5 新的變化(相較於 Lab 4)：

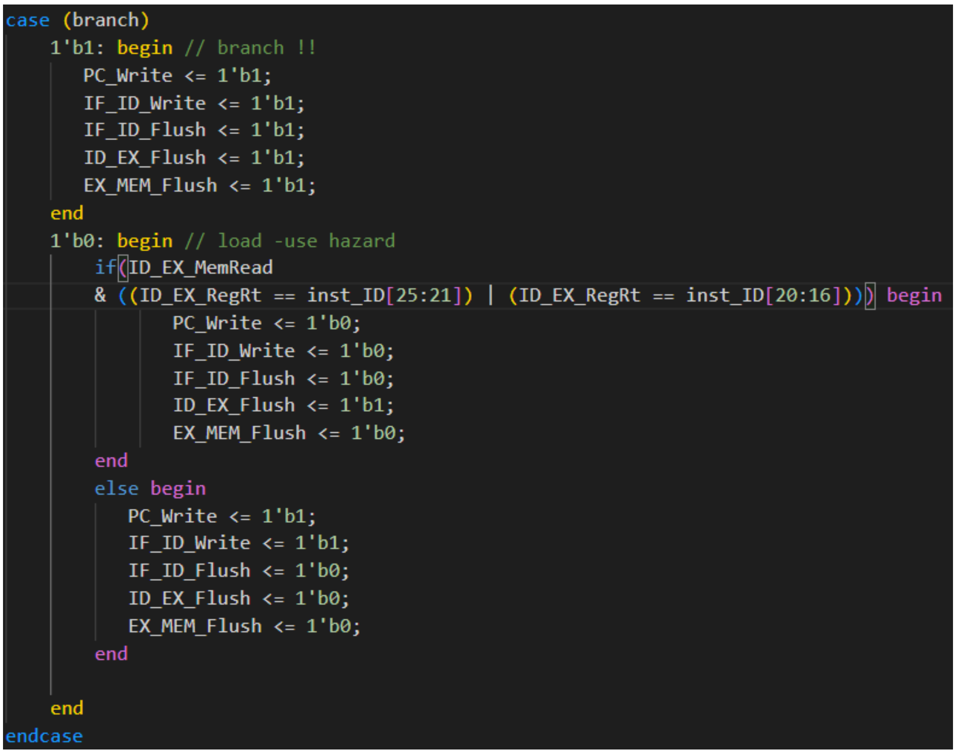
新加了 Forwarding.v 和 Hazard.v，Forwarding.v 處理 Forwarding 之後不需要加 bubble 的 data hazard，如 R-type 計算完值之後下個指令立即需要用到該值，而 Hazard.v 處理 Forwarding 之後仍要 stall 的 data hazard 或是 branch 指令要跳(原本是照順序執行，如果要 branch 的話要清掉已經執行下去的、卻不該執行的指令)，如 load-use hazard 和 beq、bne、bge、bgt 等指令

Forwarding.v 依照講義內容實作：



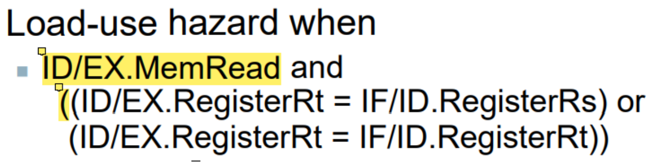
- EX hazard
  - if (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
    ForwardA = 10
  - if (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRt))
    ForwardB = 10
- MEM hazard
  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
    ForwardA = 01
  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and (MEM/WB.RegisterRd = ID/EX.RegisterRt))
    ForwardB = 01

Hazard.v 實作如下：



```
case (branch)
    1'b1: begin // branch !!
        PC_Write <= 1'b1;
        IF_ID_Write <= 1'b1;
        IF_ID_Flush <= 1'b1;
        ID_EX_Flush <= 1'b1;
        EX_MEM_Flush <= 1'b1;
    end
    1'b0: begin // load -use hazard
        if(ID_EX_MemRead
        & ((ID_EX_RegRt == inst_ID[25:21]) | (ID_EX_RegRt == inst_ID[20:16]))) begin
                PC_Write <= 1'b0;
                IF_ID_Write <= 1'b0;
                IF_ID_Flush <= 1'b0;
                ID_EX_Flush <= 1'b1;
                EX_MEM_Flush <= 1'b0;
        end
        else begin
            PC_Write <= 1'b1;
            IF_ID_Write <= 1'b1;
            IF_ID_Flush <= 1'b0;
            ID_EX_Flush <= 1'b0;
            EX_MEM_Flush <= 1'b0;
        end

    end
endcase
```

branch 的值代表當前指令有沒有要 branch，沒有要 branch 的話，會判斷是否有 load-use hazard，除 branch 和 load-use hazard 之外其餘情況保持和 Lab 4 相同的運作模式，load-use hazard 的判斷同樣是依據講義內容：



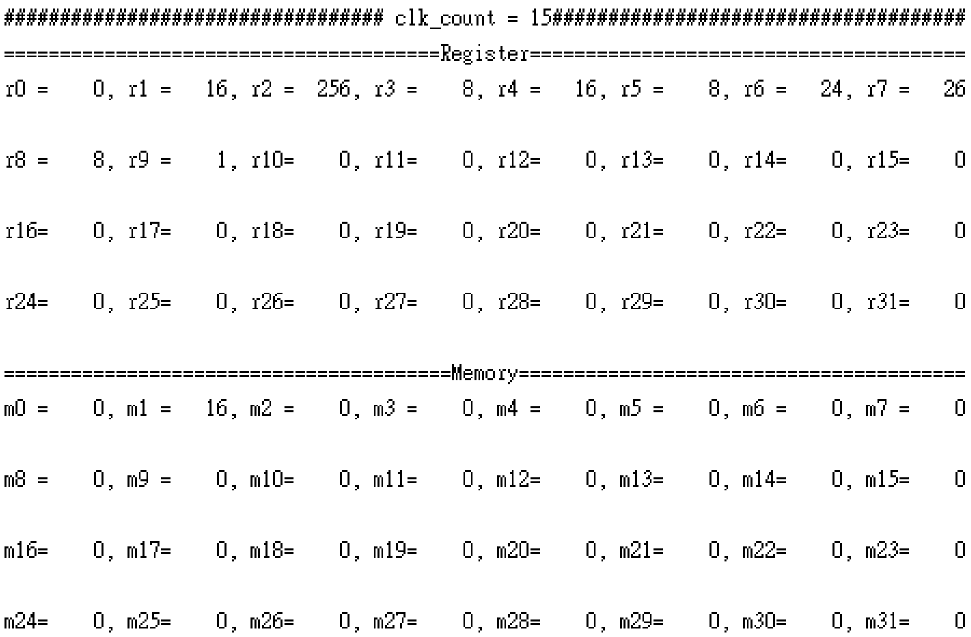
Load-use hazard when

- ID/EX.MemRead and
  ((ID/EX.RegisterRt = IF/ID.RegisterRs) or
  (ID/EX.RegisterRt = IF/ID.RegisterRt))

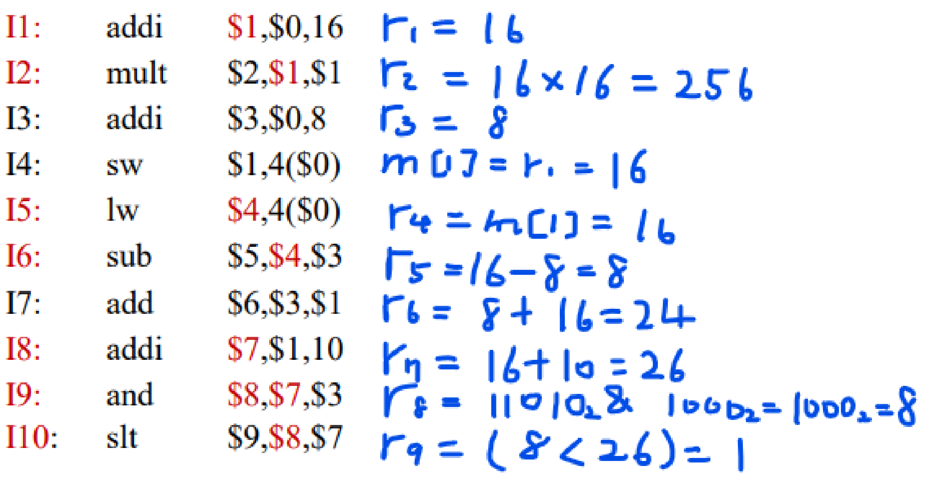
Finished part:

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

test 1：

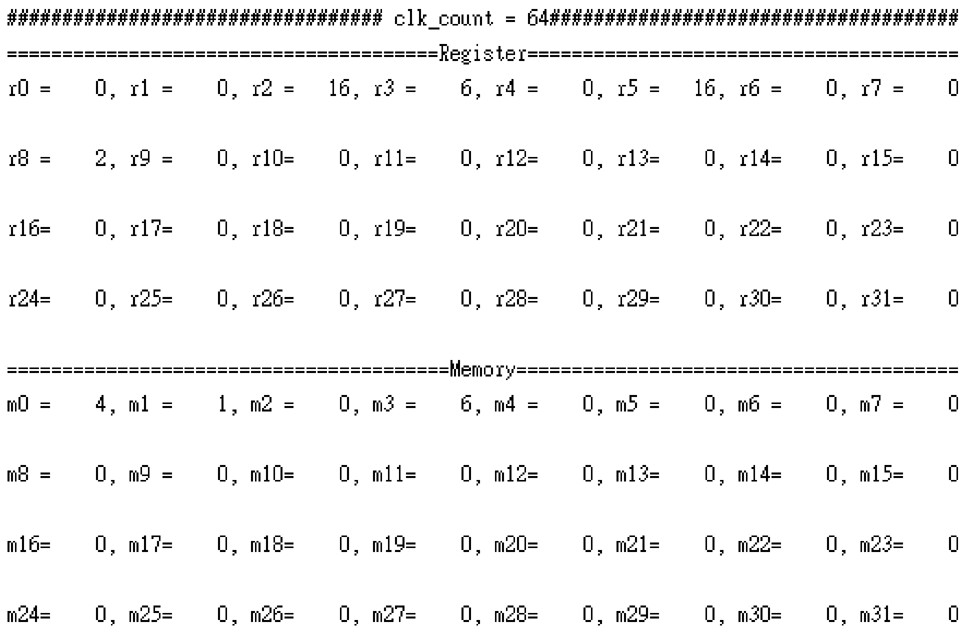


```
################################### clk_count = 15#####################################
========================================Register========================================
r0 =     0, r1 =    16, r2 =   256, r3 =     8, r4 =    16, r5 =     8, r6 =    24, r7 =    26

r8 =     8, r9 =     1, 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

m16=     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:	addi	$1,$0,16	$r_1 = 16$
I2:	mult	$2,$1,$1	$r_2 = 16 \times 16 = 256$
I3:	addi	$3,$0,8	$r_3 = 8$
I4:	sw	$1,4($0)	$m[1] = r_1 = 16$
I5:	lw	$4,4($0)	$r_4 = m[1] = 16$
I6:	sub	$5,$4,$3	$r_5 = 16 - 8 = 8$
I7:	add	$6,$3,$1	$r_6 = 8 + 16 = 24$
I8:	addi	$7,$1,10	$r_7 = 16 + 10 = 26$
I9:	and	$8,$7,$3	$r_8 = 11010_2 \& 1000_2 = 1000_2 = 8$
I10:	slt	$9,$8,$7	$r_9 = (8 < 26) = 1$

test 2：



```
################################# clk_count = 64######################################
======================================Register=======================================
r0 =     0, r1 =     0, r2 =    16, r3 =     6, r4 =     0, r5 =    16, r6 =     0, r7 =     0

r8 =     2, r9 =     0, 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 =     4, m1 =     1, m2 =     0, m3 =     6, m4 =     0, m5 =     0, m6 =     0, m7 =     0

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

m16=     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:     addi    $2, $0, 3
I2:     sw      $2, 0($0)
I3:     addi    $2, $0, 1
I4:     sw      $2, 4($0)
I5:     sw      $0, 8($0)
I6:     addi    $2, $0, 5
I7:     sw      $2, 12($0)
I8:     addi    $2, $0, 0
I9:     addi    $5, $0, 16
I10:    addi    $8, $0, 2
I11:    beq     $0, $0, 2
I12:    addi    $2, $2, 4
I13:    bge     $2, $5, 6
I14:    lw      $3, 0($2)
I15:    bgt     $3, $8, 1
I16:    beq     $0, $0, -5
I17:    addi    $3, $3, 1
I18:    sw      $3, 0($2)
I19:    beq     $0, $0, -8
```

end

I1: r2 = 3

I2: m[0] = 3

I3: r2 = 1

I4:m[1] = 1

I5: m[2] = 0

I6: r2 = 5

I7: m[3] = 5

I8: r2 = 0

I9: r5 = 16

I10: r8 = 2

I11: (0==0)，branch to I14

I14: r3 = 3

I15: (3>2)，branch to I17

I17: r3 = 4

I18: m[0] = 4

I19: (0==0)，branch to I13

I13: r2 = 4

I14: (16>6)，branch to I20

end

Problems you met and solutions:

load-use hazard 有偵測到但一直沒有用心的資料去計算，後來發現是接線接反了，應該有兩筆資料用作 Hazard，分別來自 EX stage 和 MEM stage，我把兩個 input 接反了，接回來就過了，好感動

Summary:

終於計組最後一個作業也結束了！接線生活也可以告一段落了~完結灑花