Computer Organization Lab4

Source code and the note:

這次作業中我認為最重要的module就是Pipe_Reg，

藉由這個module我們才得以實現Pipeline CPU，

在其中可以注意到我們使用<=而並非=是因為我

們需要我們的訊號能夠在register中維持一個

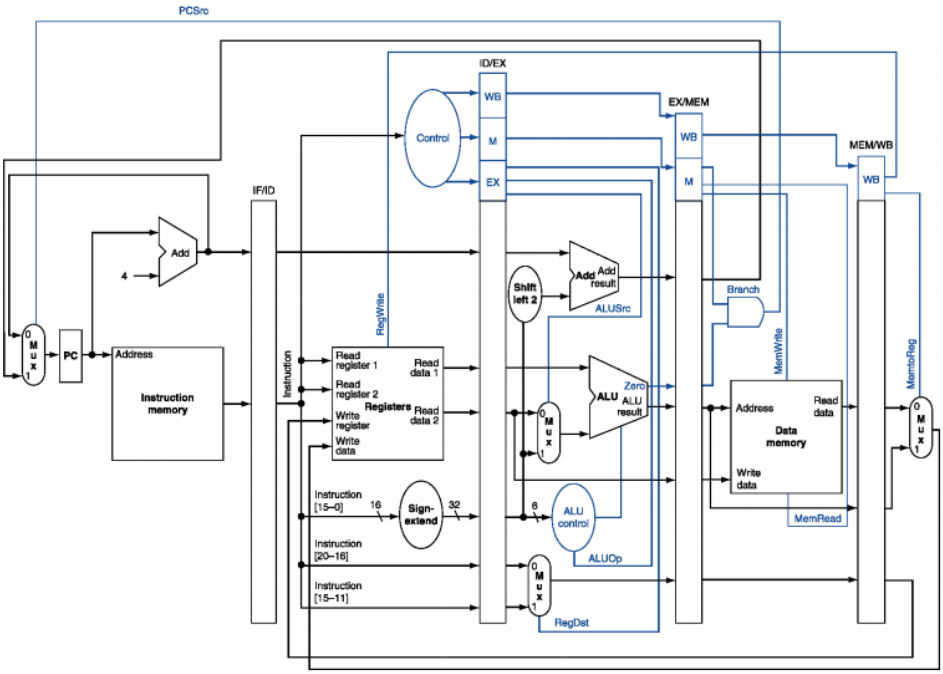
clock cycle的時間，而這是<=才能做到的，若使用

=的話，data_i一變data_o就會馬上改變，也不會

使我們希望的結果。

```
module Pipe_Reg(
    clk_i,
    rst_i,
    data_i,
    data_o
    );

parameter size = 0;

input   clk_i;
input   rst_i;
input   [size-1:0] data_i;
output reg  [size-1:0] data_o;

always@(posedge clk_i) begin
    if(~rst_i) data_o <= 0;
    else data_o <= data_i;
end

endmodule
```

Architecture diagrams:



Hardware module analysis:

這次的電路圖與 Lab2 的大致相同，最大的差別是為了要做出 Pipeline CPU，我們在不同 stage 之中新增了許多 register，讓 CPU 可以一次處理多個指令，同時控制的訊號也是與 single cycle 一樣正確無誤。

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
```

Bonus:

Machine Code:

```
00100000000000010000000000010000 addi $1, $0, 16
00100000000000110000000000001000 addi $3, $0, 8
00000000000000000000000000000000 NOP
00000000000000000000000000000000 NOP
10101100000000010000000000000100 sw     $1, 4($0)
00100000001000100000000000000100 addi $2, $1, 4
00000000011000010011000000100000 add    $6, $3, $1
10001100000001000000000000000100 lw     $4, 4($0)
00100000001001110000000000001010 addi $7, $1, 10
00100000000010010000000000001100100 addi $9, $0, 100
00000000100000110010100000100010 sub    $5, $4, $3
00000000111000110100000000100100 and    $8, $7, $3
```

Finish Part:

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