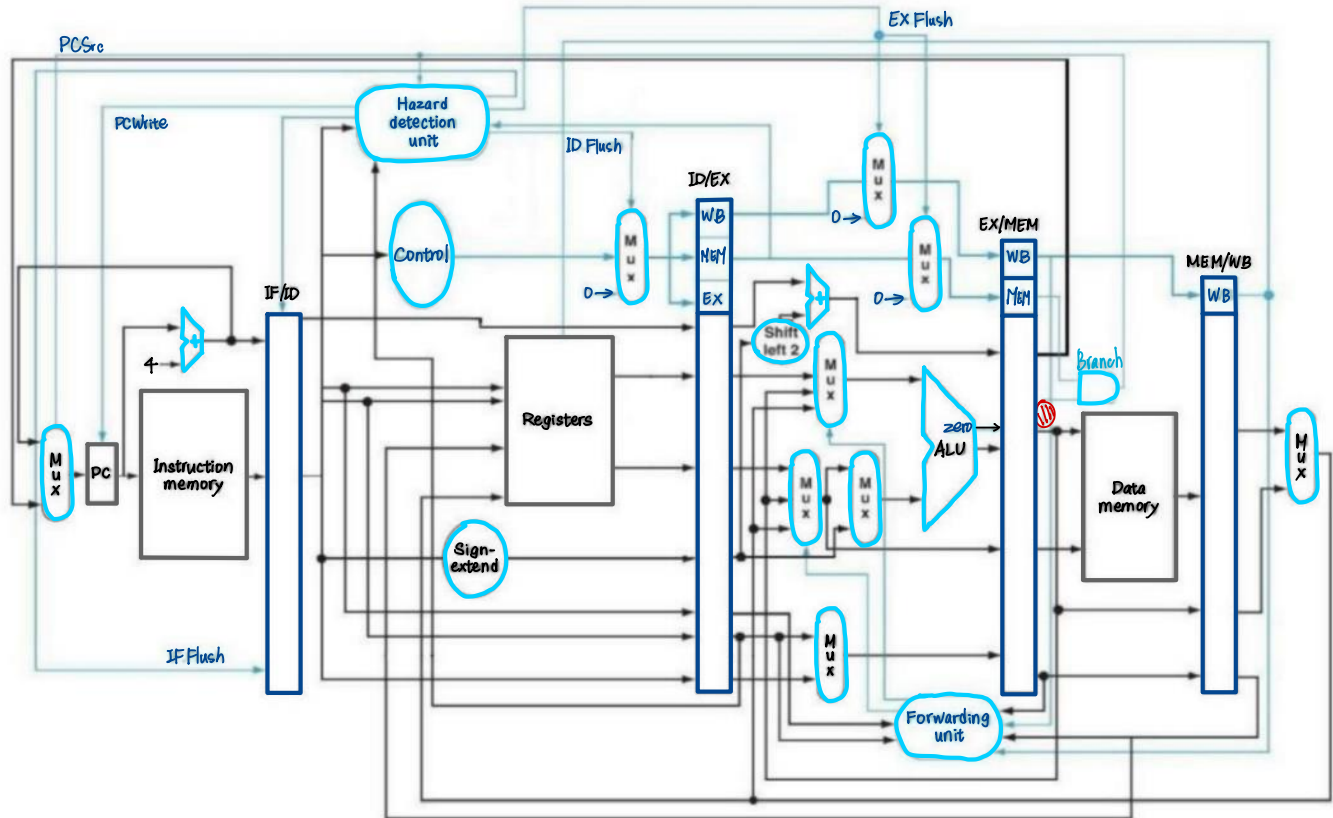


# Computer Organization Lab5

Name: 陳晶

ID: 109700045

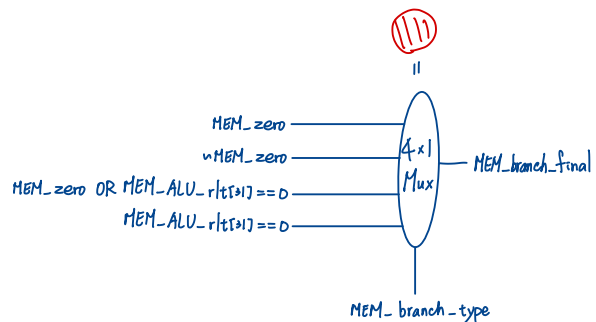
Architecture diagram:



Hardware module analysis:

優:

1. 可以同時執行許多指令，善用硬體資源
2. 可增加 throughput
3. 相較於 single cycle CPU，儘管比單個指令有可能速度較慢，但絕大部分時候都比較省時
4. 相較於上次實作的 pipelined CPU，可支援更多指令，包含 bge、bgt。
5. 相較於上次實作的 pipelined CPU，可用硬體的方法解決 data hazard

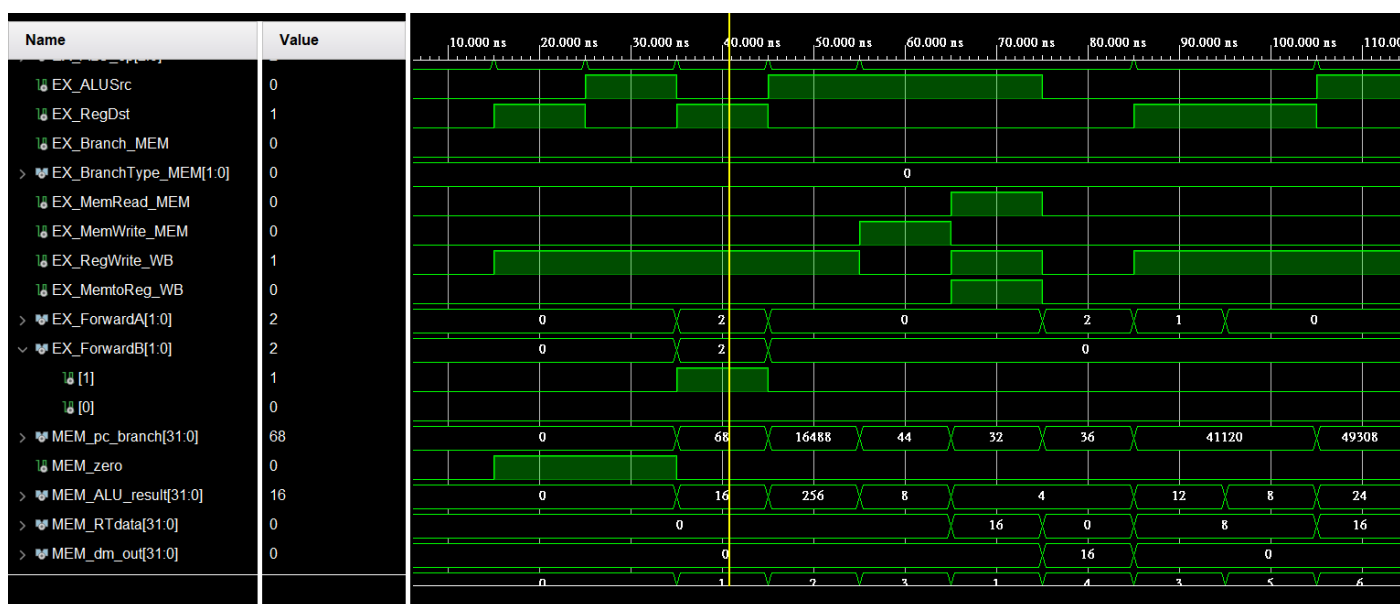


缺:

1. 需要額外的 flip flops 儲存每個 stage 的結果。
2. 實作時較複雜。

## Finished part:

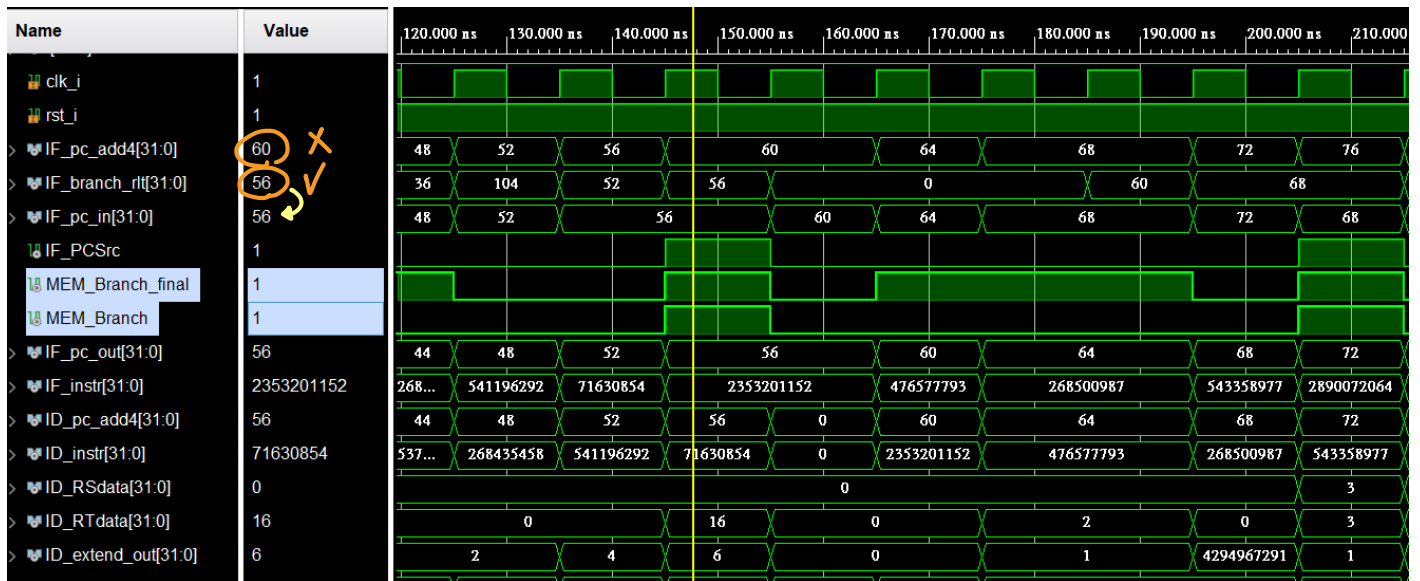
Test data 1:



在第四個 clock，因為「addi 的目的暫存器」==「mult 的來源暫存器」，所以 forwardA 和 forwardB 都變 10。

```
##### clk_count = 17#####  
-----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
```

## Test data 2:



在黃線指的這個 clock 為第 11 個 instruction (beq \$0, \$0, 2) 位於 mem stage 的 clock，此時將 branch mux 運算出的 MEM\_branch\_final 以及原有的 MEM\_branch 作 AND 運算，能夠確定要跳到 branch 的位址，所以在 IF\_pc\_add4 和 IF\_branch\_rlt 中，MUX 選擇 IF\_branch\_rlt 作為 PC 的 input。

test data 2 的完整運算過程:

### i. Testbench ("CO\_P5\_test\_2.txt"):

4	I1:	addi	\$2, \$0, 3	C6	$r_2 \rightarrow 3$
8	I2:	sw	\$2, 0(\$0)	C7	$m_0 \rightarrow 3$
12	I3:	addi	\$2, \$0, 1	C8	$r_2 \rightarrow 1$
16	I4:	sw	\$2, 4(\$0)	C9	$m_1 \rightarrow 1$
20	I5:	sw	\$0, 8(\$0)	C10	$m_2 \rightarrow 0$
24	I6:	addi	\$2, \$0, 5	C11	$r_2 \rightarrow 5$
28	I7:	sw	\$2, 12(\$0)	C12	$m_3 \rightarrow 5$
32	I8:	addi	\$2, \$0, 0	C13	$r_2 \rightarrow 0$
36	I9:	addi	\$5, \$0, 16	C14	$r_5 \rightarrow 16$
40	I10:	addi	\$8, \$0, 2	C15	$r_8 \rightarrow 2$
44	I11:	beq	\$0, \$0, 2	C16	$0 = 0$ 分支
48	I12:	addi	\$2, \$2, 4	C32	$r_2 \rightarrow 4$
52	I13:	bge	\$2, \$5, 6	C33	$r_2 < r_5$
56	I14:	lw	\$3, 0(\$2)	C20	$r_3 = 3$
60	I15:	bgt	\$3, \$8, 1	C21	$r_3 > r_8$ 分支
64	I16:	beq	\$0, \$0, -5	C35	$r_3 < r_8$
68	I17:	addi	\$3, \$3, 1	C26	$r_3 \rightarrow 4$
72	I18:	sw	\$3, 0(\$2)	C27	$m_0 \rightarrow 4$
76	I19:	beq	\$0, \$0, -8	C28	$0 = 0$ 分支
80					

Handwritten notes and arrows indicate the state of registers and memory locations during the execution of the instructions. For example, at clock cycle 48,  $r_2 = 4$ ,  $r_5 = 16$ ,  $r_3 = 3$ , and  $m_0 = 4$ . The diagram also shows the state of the branch mux and the final PC value.

```
##### 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
```

## Problems you met and solutions:

1. 無法處理 addi 接 sw 時的 data hazard。

Solution: 檢查所有相關參數並更正。

2. 執行 bge、bgt 指令時，會用到 ALU result 是否為負數來判斷要不要 branch，但預設的資料型別為 unsigned，無法用大於零或小於零來判斷 ALU result 的正負。

Solution: 用 MEM\_ALU\_result[31]來判斷正負，是 1 則為負，是 0 則為正，如圖：

```
MUX_4to1 #(.size(1)) MuxBranch(
    .data0_i(MEM_zero),
    .data1_i(~MEM_zero),
    .data2_i(MEM_zero | (MEM_ALU_result[31] == 0)),
    .data3_i(MEM_ALU_result[31] == 0),
    .select_i(MEM_BranchType),
    .data_o(MEM_Branch_final)
);
```

3. load use hazard 和 branch hazard 的條件同時滿足時會出錯，因為處理 load use hazard 的部分把 PCwrite 設為 0，導致 PC out 無法更新到 branch 後的指令。

Solution: 把 load use hazard 的條件加上「& (~Branch\_i)」，如圖：

```
if((ID_EX_MemRead_i &((ID_EX_RegisterRt_i == IF_ID_RegisterRs_i)|
  (ID_EX_RegisterRt_i==IF_ID_RegisterRt_i))) & (~Branch_i))
begin
  PCWrite_o <= 0;
  IF_IDWrite_o <= 0;
  Control_o <= 0;
end
```

## Summary:

進階的管線化處理器是個聰明的設計，不但提高 Instruction-Level Parallelism，大幅提升處理器的效能，也能解決隨之而來的 data hazard。

心得：不知不覺已經寫完最後一份計組作業了，覺得挺有成就感的，因為在修計組之前沒有修數位電路設計，所以一開始看到波形圖覺得密密麻麻黑黑綠綠的很可怕，甚至連什麼是 sequential circuit 都沒任何概念 XD，不過經過一次次的摸索，慢慢能從寫 c/c++ 的思考模式轉換到寫 verilog 的模式了，也比較會看波形圖來 debug 了，看得懂的感覺真好 ☺。

謝謝助教們辛苦地改作業！