期末project

108學年度第2學期

老師：朱守禮老師

學生：

資訊二乙35組

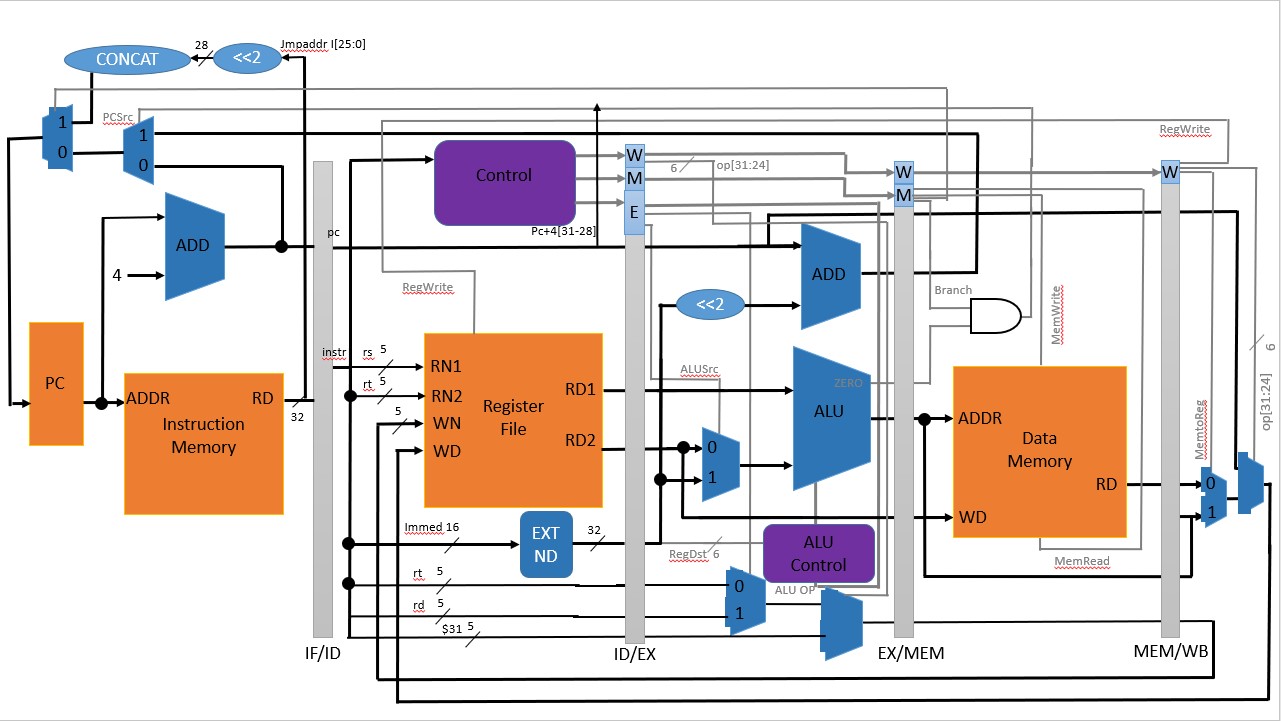
10627233 蔡允齊

10627232蔡旻樺

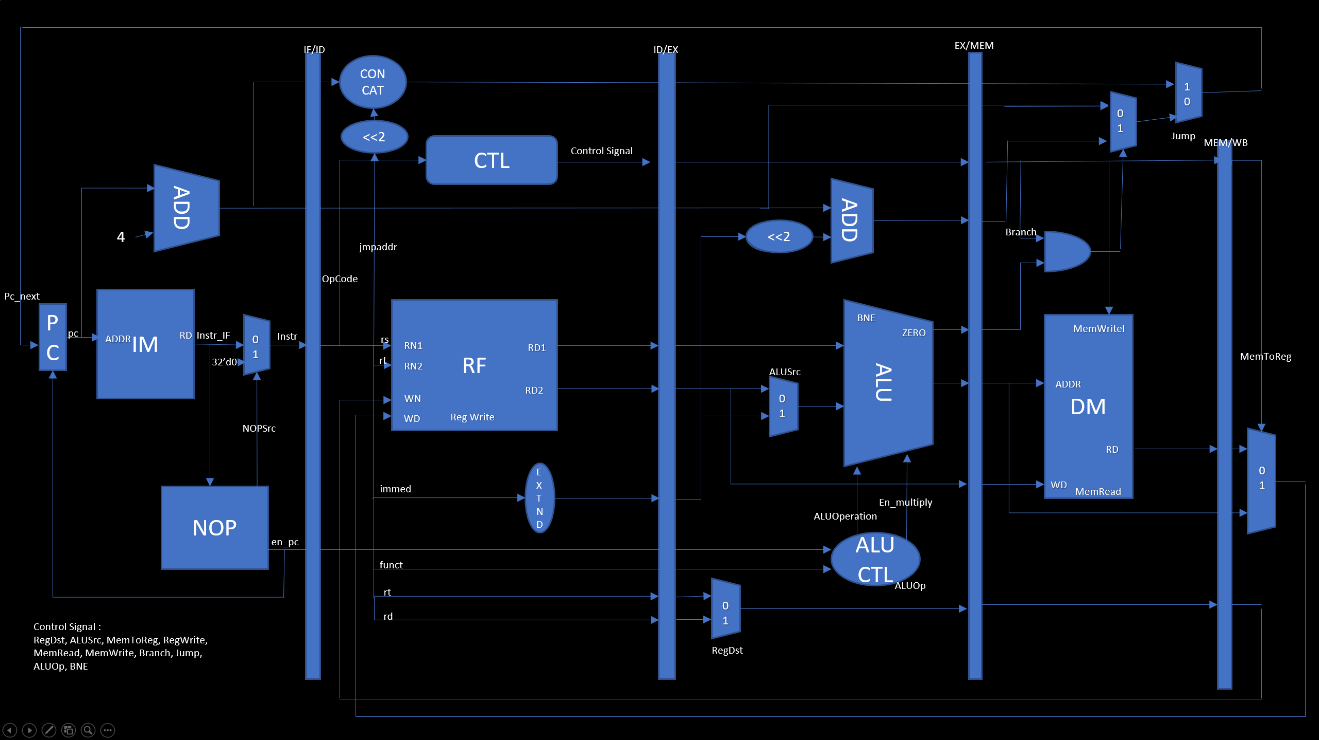
10627229 陳寬祐

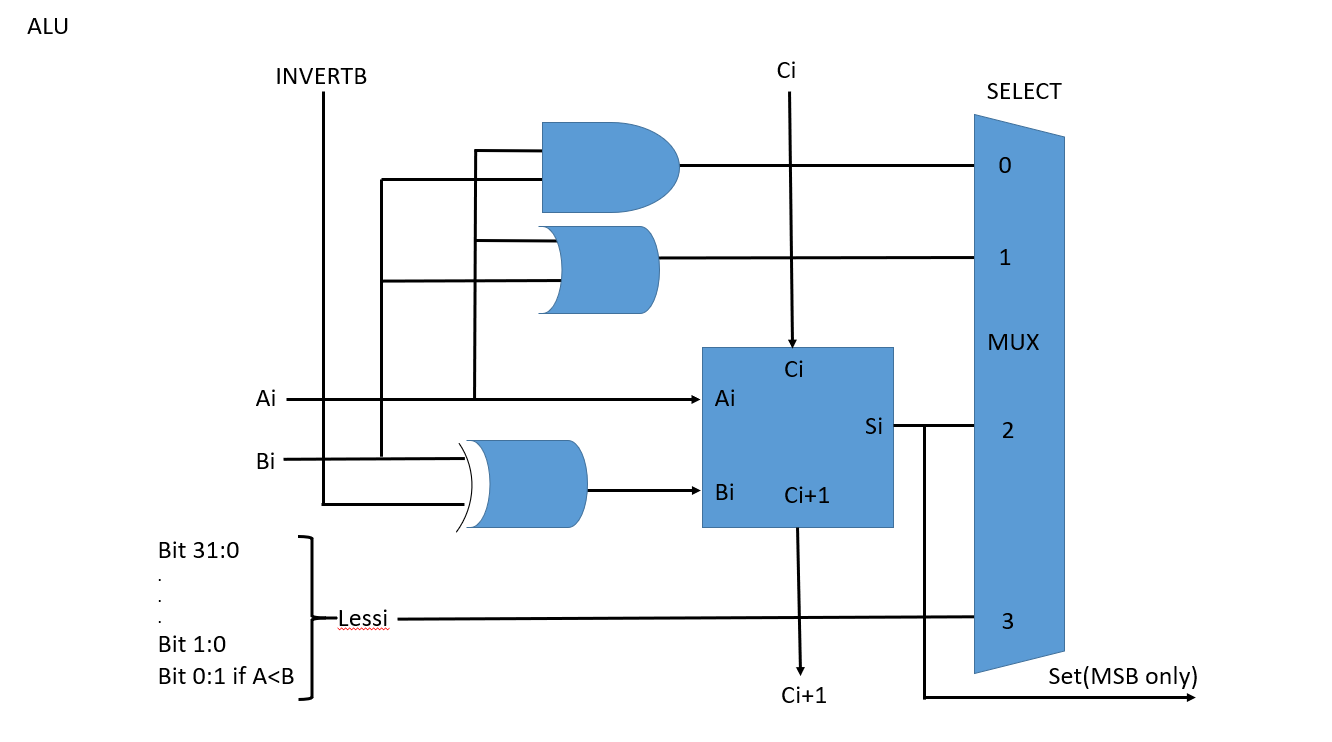
1. Datapath與詳細架構圖

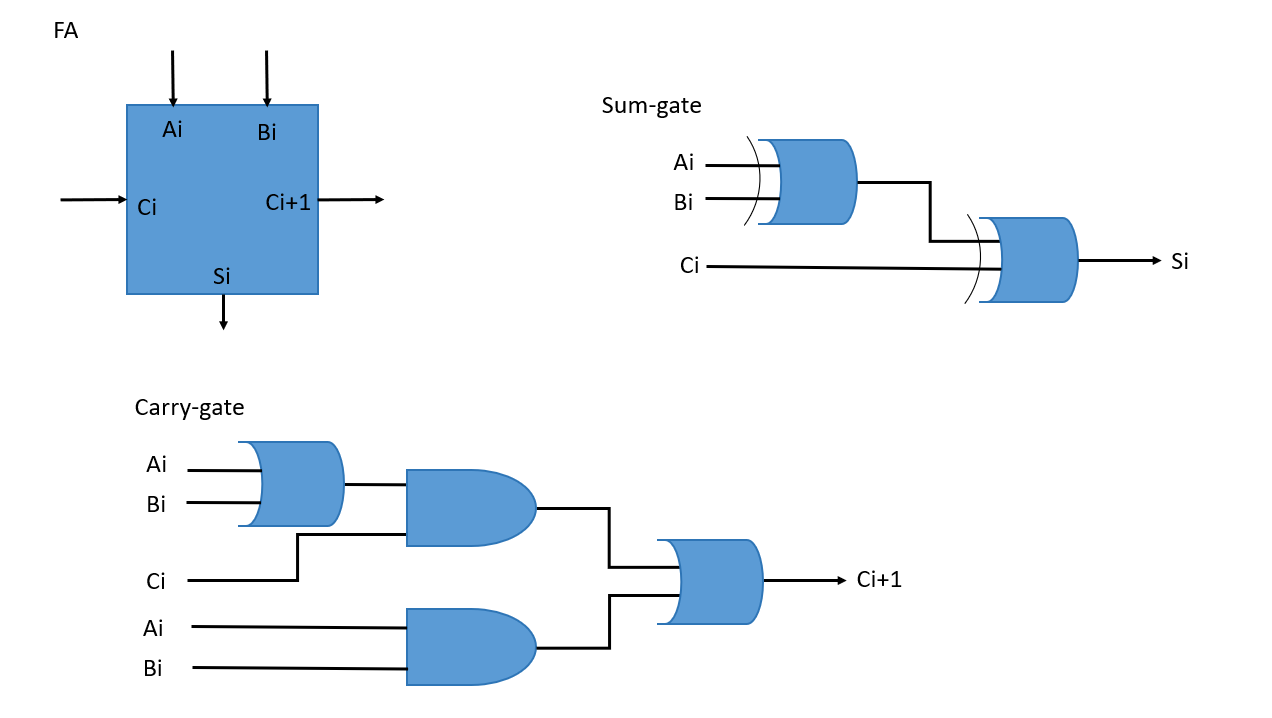
原始 pipeline

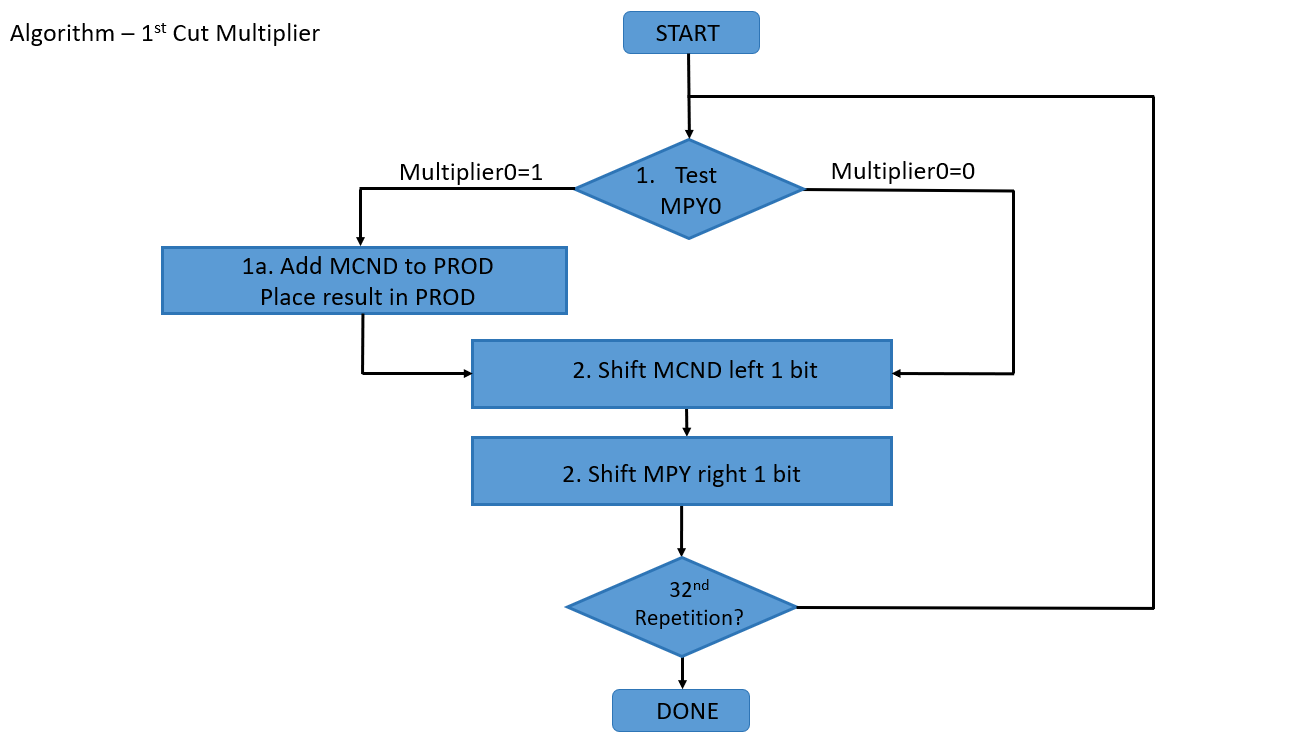


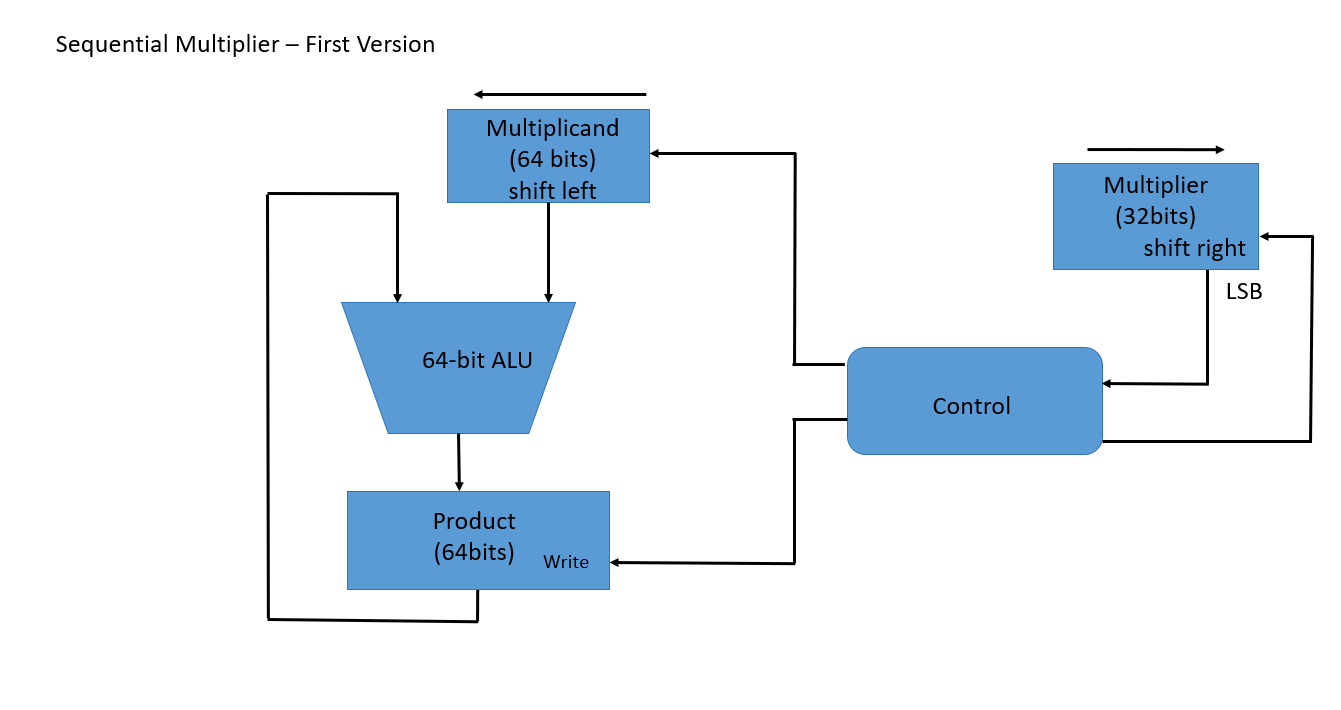
加入 nop 控制後

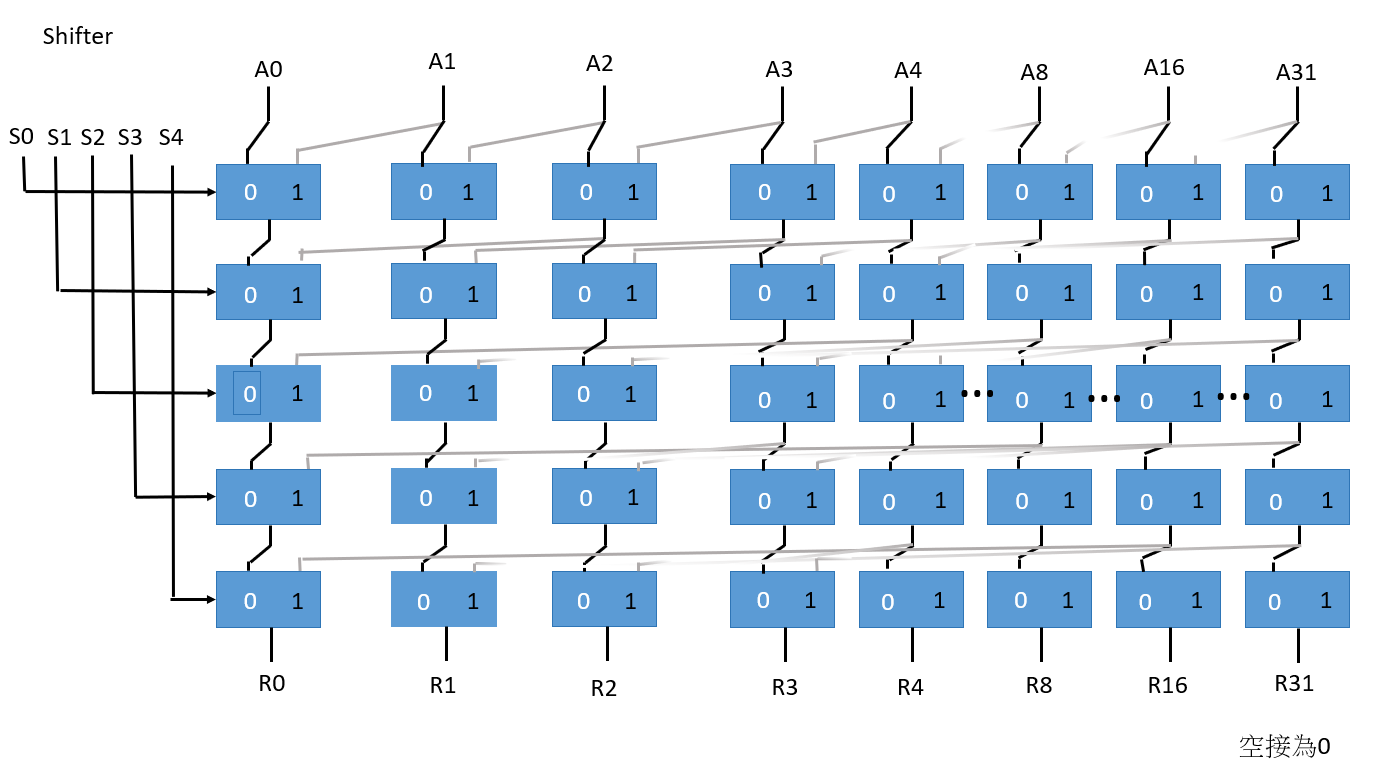












二、設計重點說明

**add32**: 把PC = PC + 4

**tb\_PipeLine:** 使用者可自行設定指令及參數以進行測試

**FA:** 全加器

**ALU\_1bit:** 做bitwise AND/OR/ADD/SUB/SLT運算

**ALU\_msb:**  ALU的最高位元，跟ALU\_1bit是相同的，只差在會處理set訊號

**ALU\_32bits:**  創立31個ALU\_1bit加1個ALU\_msb並將他們接起來表示一個32bit的ALU，裡面處理了ALU/shifter/multiplier/HiLo的訊號

**alu\_ctl:** 先判斷ALUOp來決定是做加還是減或是其他指令。其他指令是根據Funct選擇要做哪一個運算(add,sub,and,or,slt)並將對應的訊號傳入對應的電路中

**Multiplier:** 乘法器做法就是乘數的第0格如果是1的話，product加上被乘數，接著乘數 右移看下一位元，被乘數左移後面補一個0，執行時間為ALU/Shifter的32倍 (32個週期)

**HiLo:** 會將Multiplier算出的答案的前32bit放進HiOut，後32bit放進LoOut

**Shifter:** 將輸入的數字向左偏移，由160個mux組成 (32\*5)

**mux2:** 根據sel選擇要輸出哪一個訊號(二擇一)

**sign extend:** 把有號數從16bit擴充到32bit

**reg\_PC:** 設定pc

**reg\_IF2ID:** 備份PC及Instruction的訊號

**reg\_ID2EX:** 備份RegisterFileRD1，RegisterFileRD2，signed extend，rt及rd的訊號

**reg\_Control\_IDEX:** 備份Control傳出的訊號

**reg\_EX2MEM:** 備份PC+4+offset，alu\_out，Zero，RegisterFileRD2及RFMUX的訊號

**reg\_Control\_EXMEM:** 備份reg\_Control\_IDEX傳出的訊號

**reg\_MEM2WB:** 備份DataMemoryRD，alu\_out及RFMUX的訊號

**reg\_Control\_MEMWB:** 備份reg\_Control\_EXMEM傳出的訊號

**reg\_file:** 根據各指令解析出RD1及RD2的值，若RegWrite為1則將WD寫入WN的位置

**nop\_detector:** 處理做branch系列與乘法指令時要控制PC的模組與處理nop的模組

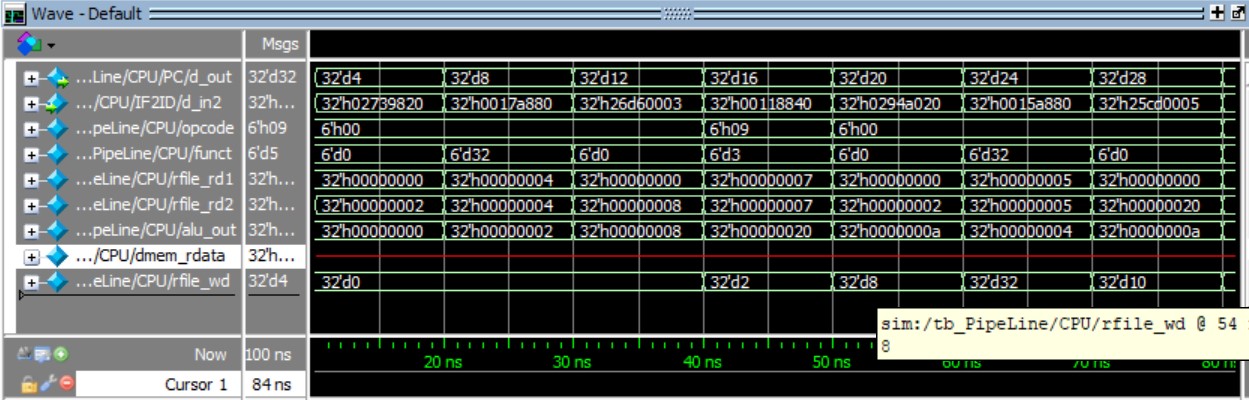
**mips\_pipeline:** 實行pipeline的模組

**memory:** 根據MemWrite或MemRead決定是否從記憶體中進行讀或寫的動作

**control:** 根據不同指令控制RegWrite, Branch, PCSrc, RegDst, MemtoReg, MemRead, MemWrite, ALUSrc, Jump, en\_pc\_ID, BNE 的訊號

三、Modelsim 驗證結果與 Waveform 輸出圖形，並加以說明。

1: // add $s3, $s3, $s3 (4+4 = 8)



4. 不用記憶體讀/寫

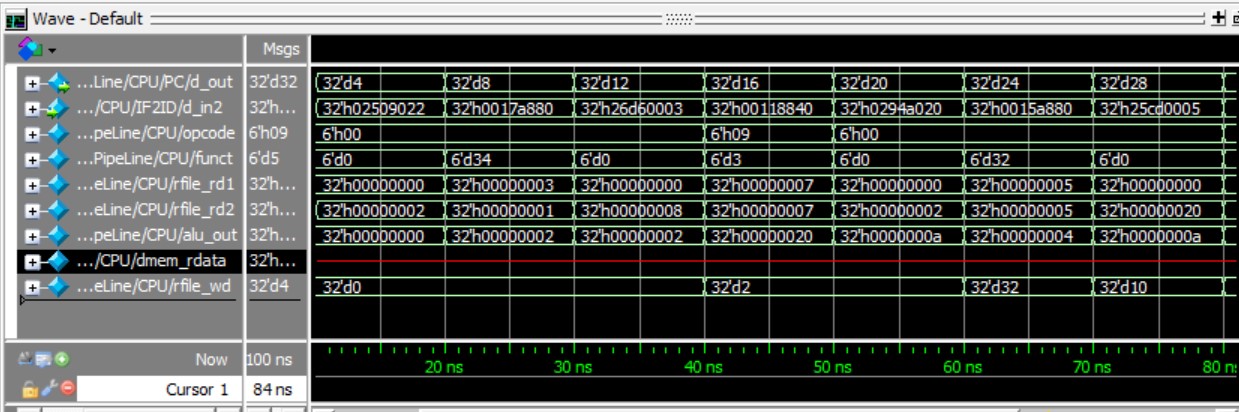
3.執行

1.讀取指令

5.回傳

2.分析指令

2: // sub $s2, $s0, $s2 (3-1=2)



4. 不用記憶體讀/寫

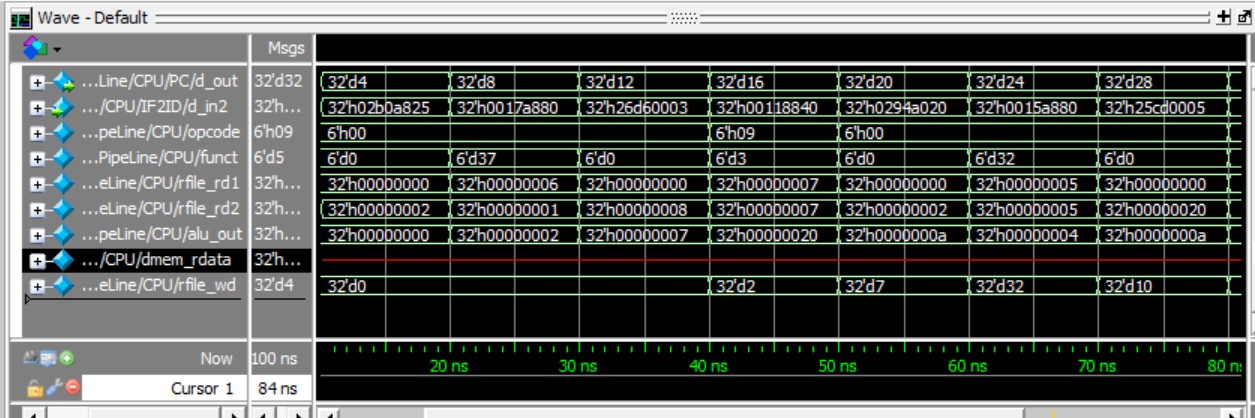
1.讀取指令

2.分析指令

3.執行

5.回傳

3. // OR 21, 16, 21 (6 | 1 = 7 )



4. 不用記憶體讀/寫

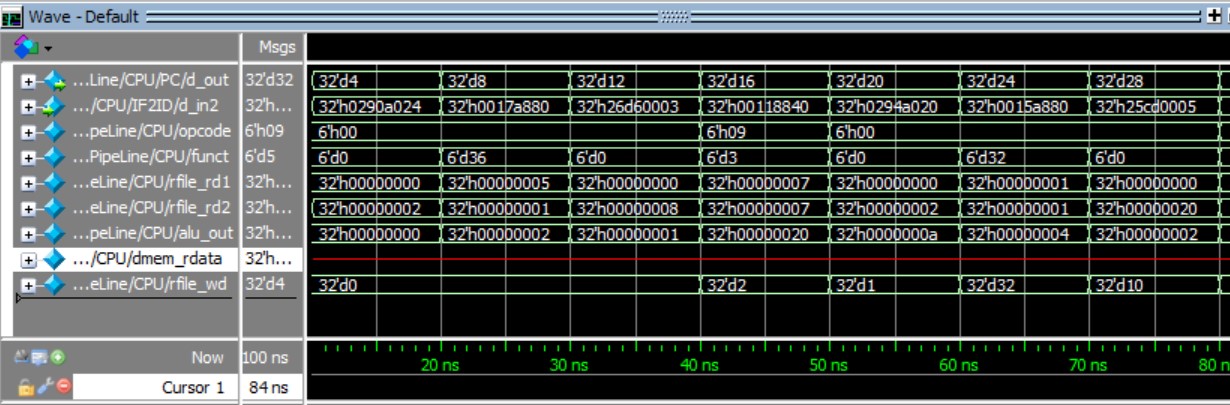
5.回傳

3.執行

2.分析指令

1.讀取指令

4. // AND, 20, 16, 20 (5&1 = 1)



4. 不用記憶體讀/寫

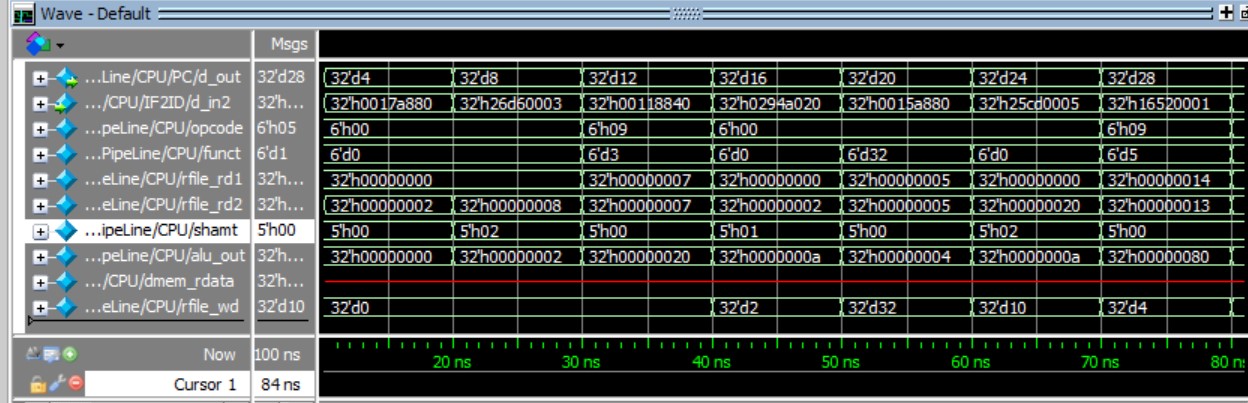
1.讀取指令

2.分析指令

3.執行

5.回傳

5. // SLL 21, 23, 2 (8<<2 = 32)



4. 不用記憶體讀/寫

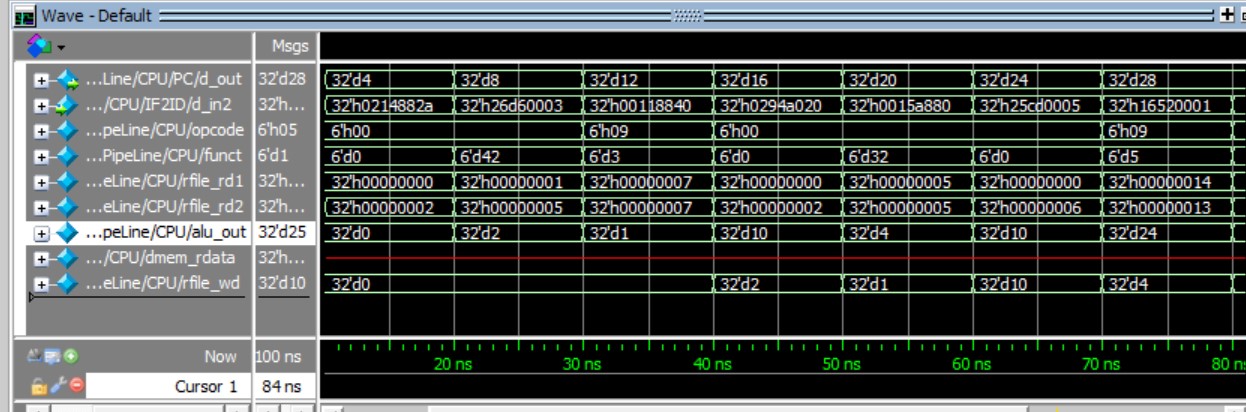
1.讀取指令

5.回傳

3.執行

2.分析指令

6. // SLT 19, 19, 16 ( 1-5=-4, slt=1)



4. 不用記憶體讀/寫

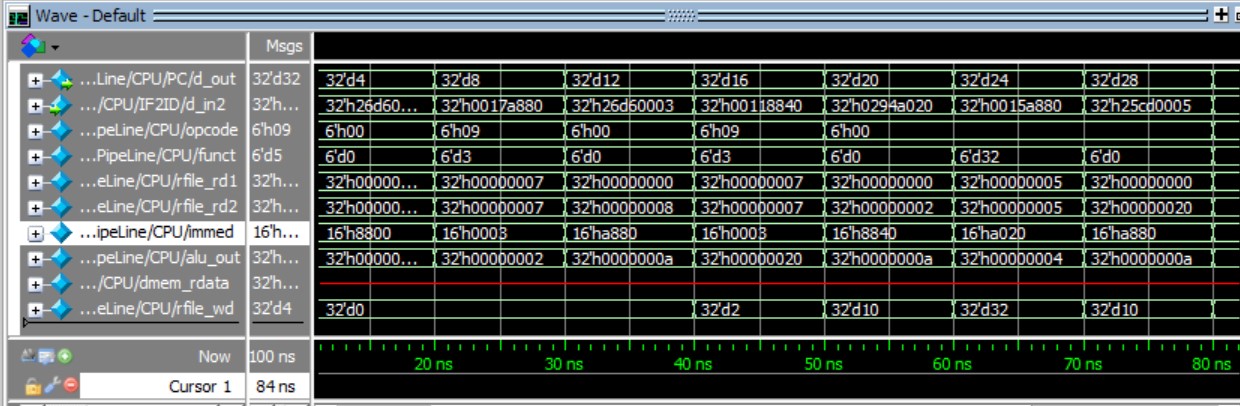
5.回傳

3.執行

2.分析指令

1.讀取指令

7. // ADDIU 22, 22, 3 (7+3=10)



4. 不用記憶體讀/寫

1.讀取指令

2.分析指令

3.執行

5.回傳

8. // multu $s6, $s7 (7X8=56)

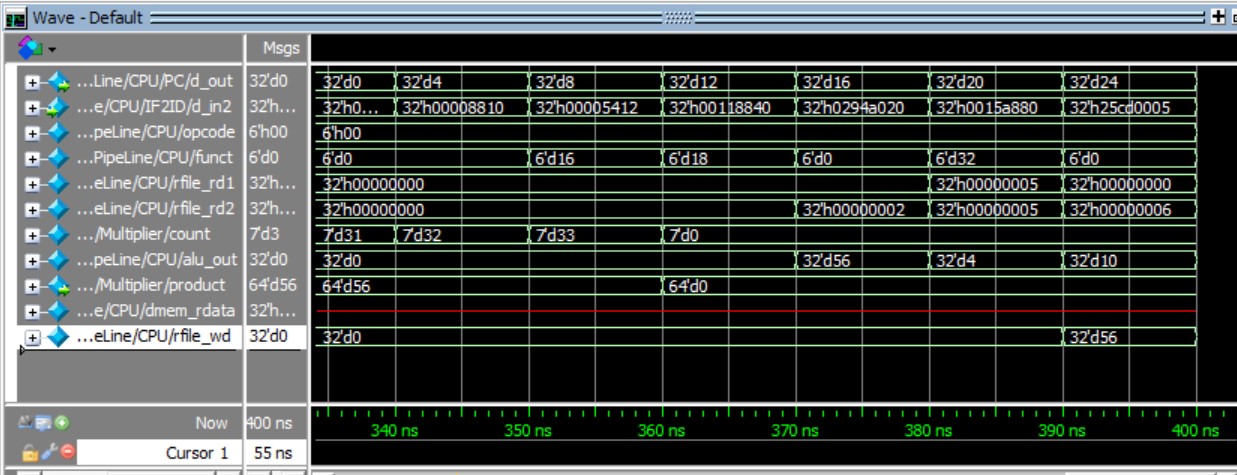


3.執行

2.分析指令

1.讀取指令

9. // mfhi s1 (54’s hi = 0 )



4. 不用記憶體讀/寫

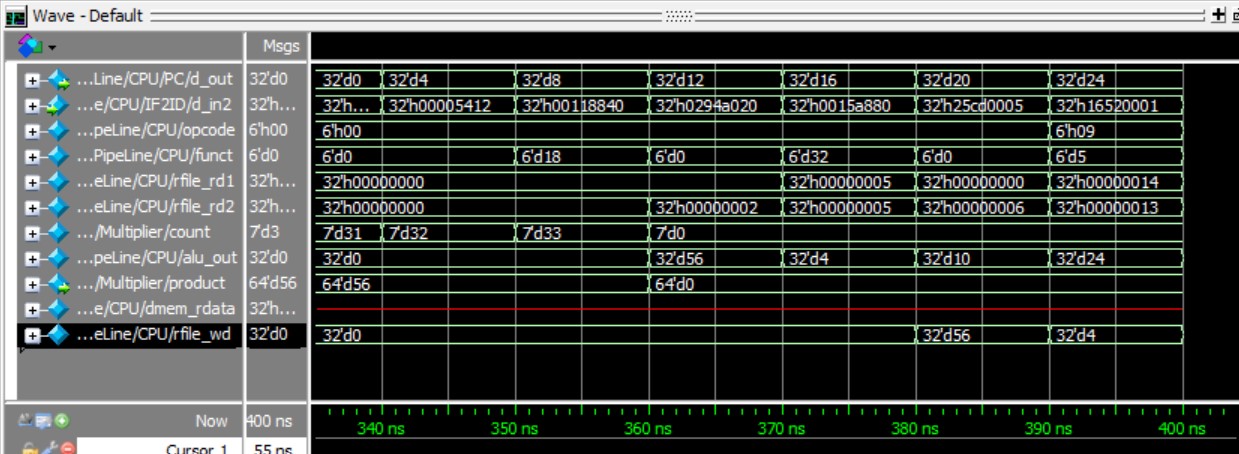
3.執行

5.回傳

2.分析指令

1.讀取指令

10. // mflo s5 (54’s low = 54 )



4.不用記憶體讀/寫

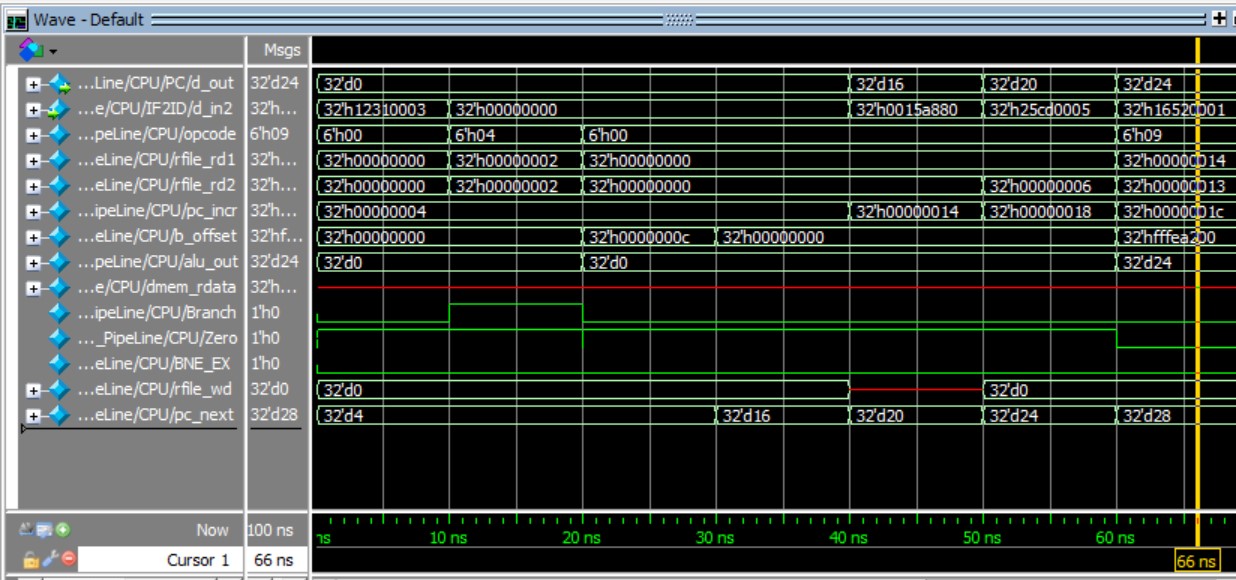
3.執行

5.回傳

2.分析指令

1.讀取指令

11. // beq $s1, $s1, 3 (PC = PC+4+4\*3 = 0 + 4 +12 = 16 )



3.執行

rd1-rd2= 2-2=0

Offset = 12

Zero=1

Beq成立

5.不用寫回register

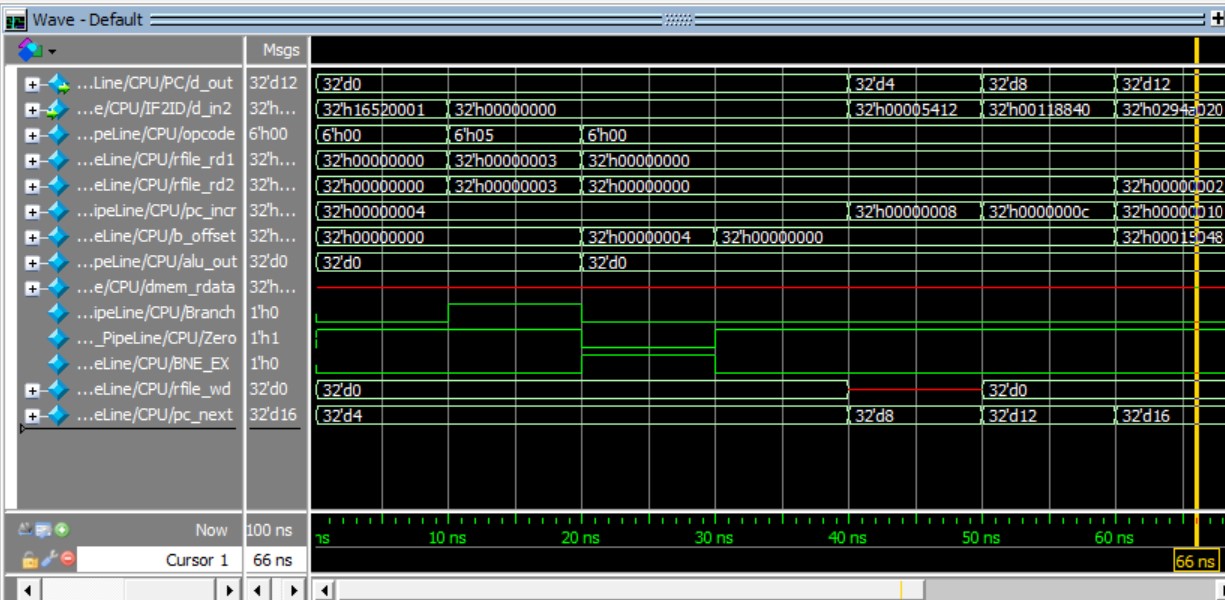
4.回傳

nextPC=16

2.分析指令

1.讀取指令

11. // BNE 18, 18, 1 (PC = PC+4 = 0 + 4 = 4 )



3.執行

rd1-rd2= 3-3=0

Offset = 4

Zero = 0

Bne不成立

5.不用寫回register

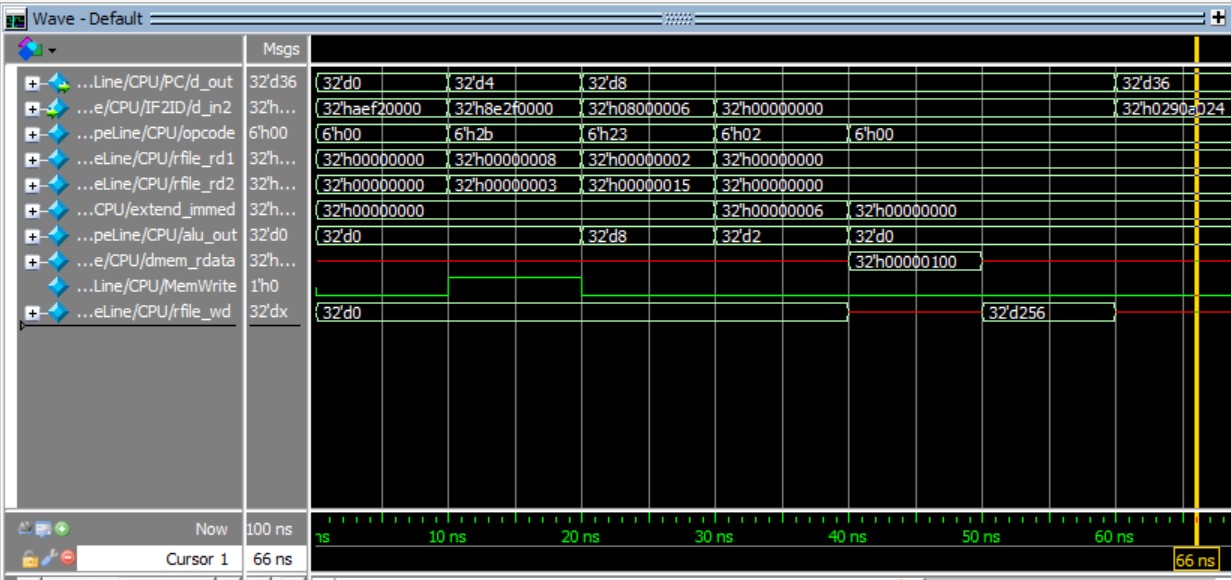
4.回傳

nextPC=4

2.分析指令

1.讀取指令

12. // sw $s1, $s2, 0



3.執行

8+0=8

4. 記憶體讀/寫

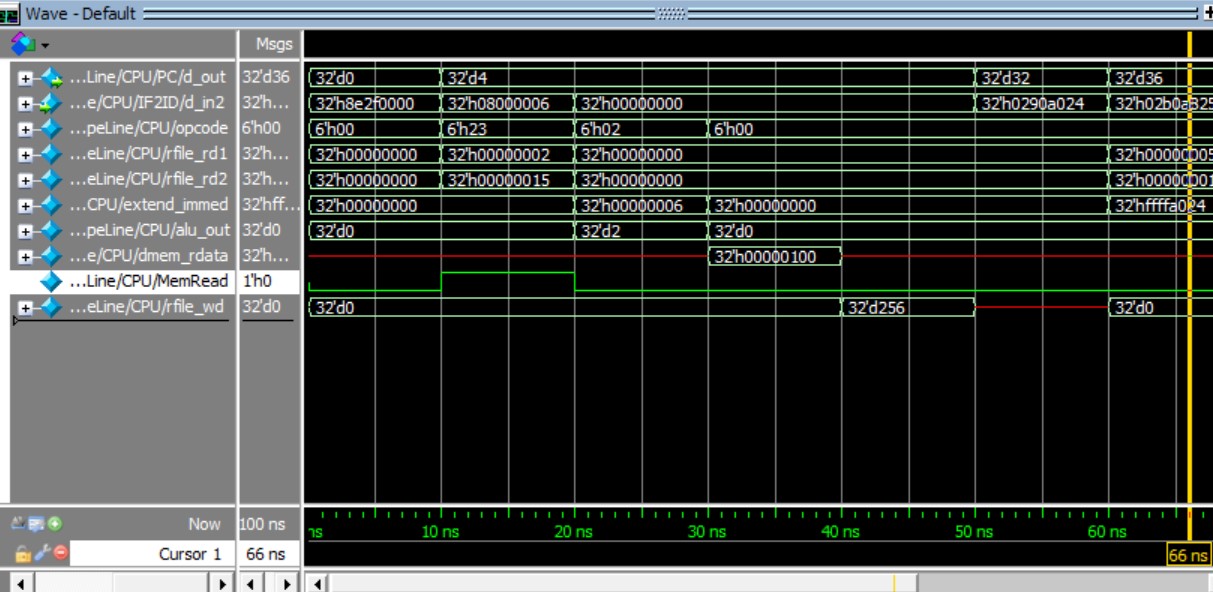
MemWrite=1

把WD寫到addr的位置(8)

2.分析指令

5.不用回傳

1.讀取指令

// lw $s1, $t7, 0

4. 記憶體讀/寫

MemRead=1

讀算出memory位置的值

5.寫回RegisterFile\_WD

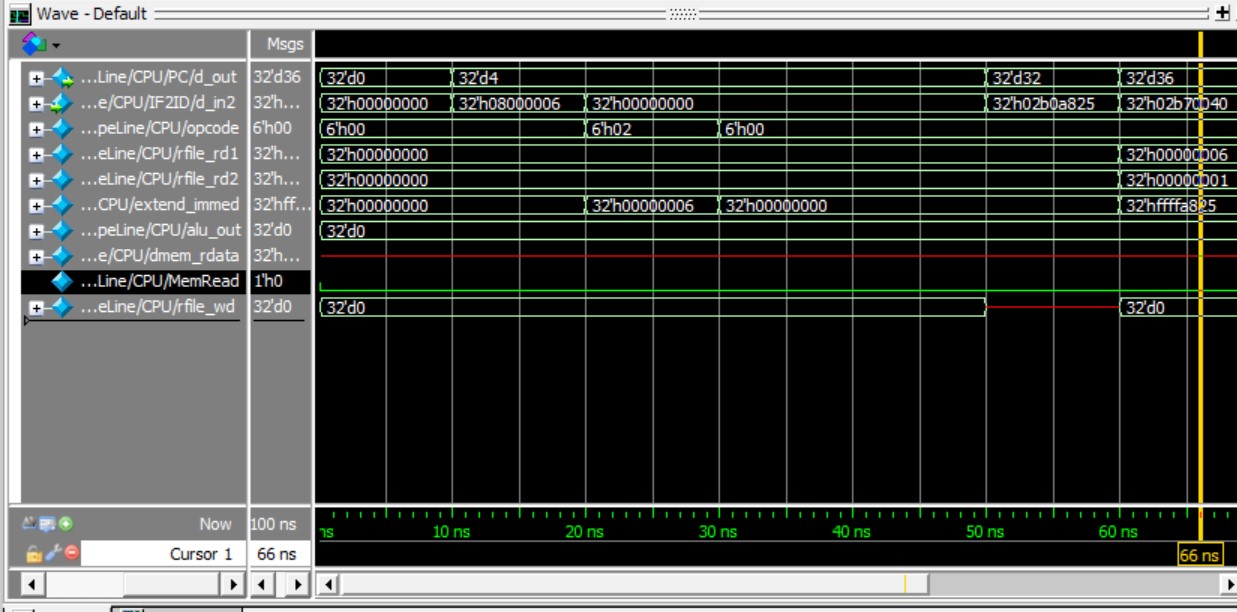
3.執行

2+0=2

2.分析指令

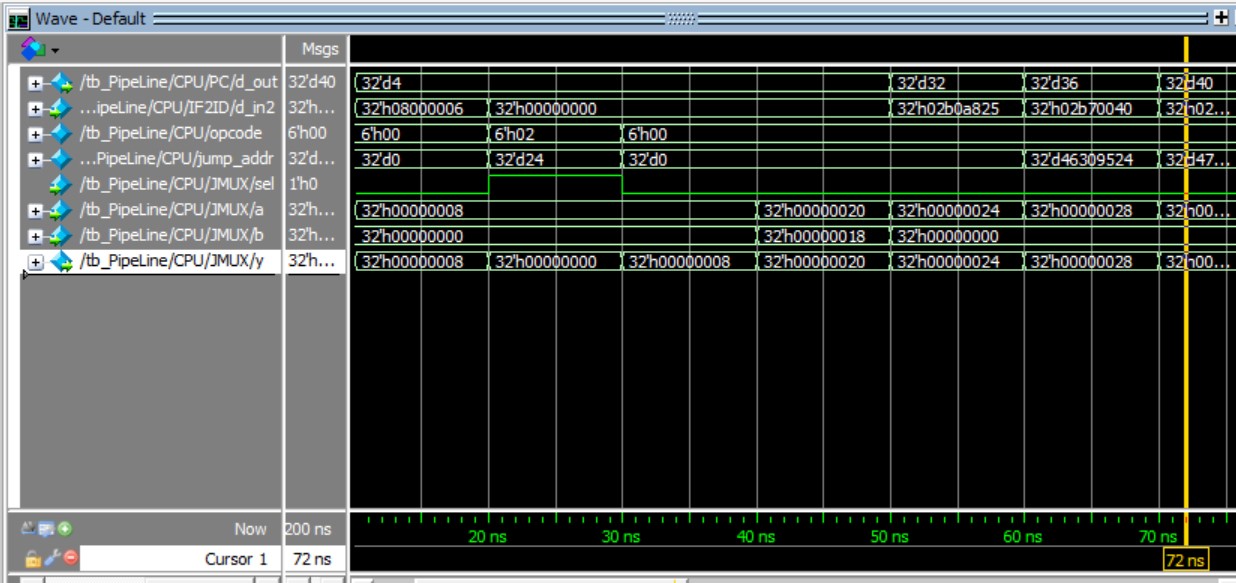
1.讀取指令

// nop



1.讀取指令

32’d0

// j 6

3. jump完後

1.讀取指令

2. 跳多少位置

五、心得感想:

這次的project讓我把課本上學到的計組知識實際上操作了一遍。像是將訊號從testbanch傳到要執行的模組並且實行了pipeline。Pipeline把執行一道指令切成5部份，可以在執行一道指令的一部分時，同時處理另一道指令，因此會較有效率。但是也因此指令有可能會衝突到，所以寫了**nop\_detector**進行延遲來解決。我覺得這次project雖然較為複雜且要花很多時間debug，但是過程中也因此更了解了mips pipeline實行的流程，是次很棒的學習經驗。

六、各組員分工方式與負責項目

程式碼 主要由 蔡允齊 負責

書面報告由 蔡旻樺 陳寬祐 負責