Final Project

106學年度第2學期

Pipelined CPU Design

老師：　朱守禮　老師

學生： 10527130 陳少洋

學生： 10527132 林亞吟

學生： 10527135 張智欽

學生： 10527140 初　元

一、背景

本Final Project主要為設計一個Pipelined CPU、第三版除法器，其目的主要為實現十六項功能：AND、OR、ADD、SUB、SLT、SLL、SLTI、LW、SW、BEQ、J、JAL、DIVU、MFHI、MFLO、NOP。

本次Project我們使用ModelSim來compiler以及執行，透過模擬waveform來檢查是否與Testbench的計算結果相同。

本次Project須符合以下規範：

1. 一個Module一個檔案，同時檔案名稱需與Module名稱相同。
2. 須放置以Visio、Word或PowerPoint繪製的Datapath架構圖。
3. Testbench須依助教所提供之參考設計。
4. 設計均以Verilog完成，且須通過ModelSim模擬執行。
5. Verilog設計只能包含規定的十六項功能。



二、方法

1. **ALU(組合邏輯)：**

沿用MidTerm Project設計的ALU，新增SLL指令。

1. **設計重點**

我們先設計1-bit的ALU，instance 32次，達到Ripple Carry的效果。

1. **說明**

* ADD、SUB、SLT不可直接使用”+”、”-“，必須以邏輯閘撰寫設計。
* AND : 000 / 36 / 100100

OR : 001 / 37 / 100101

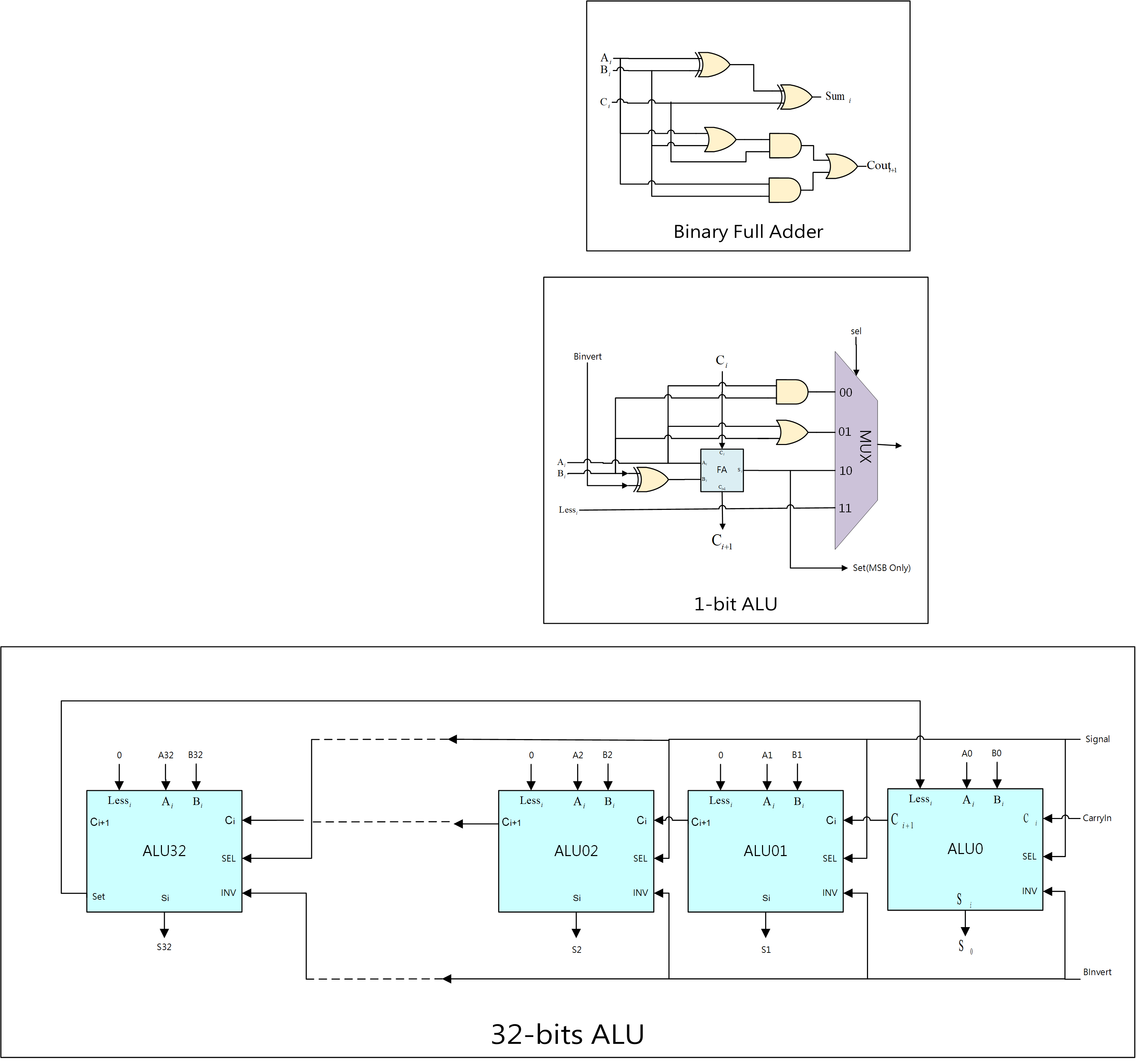
ADD : 010 / 32 / 100000

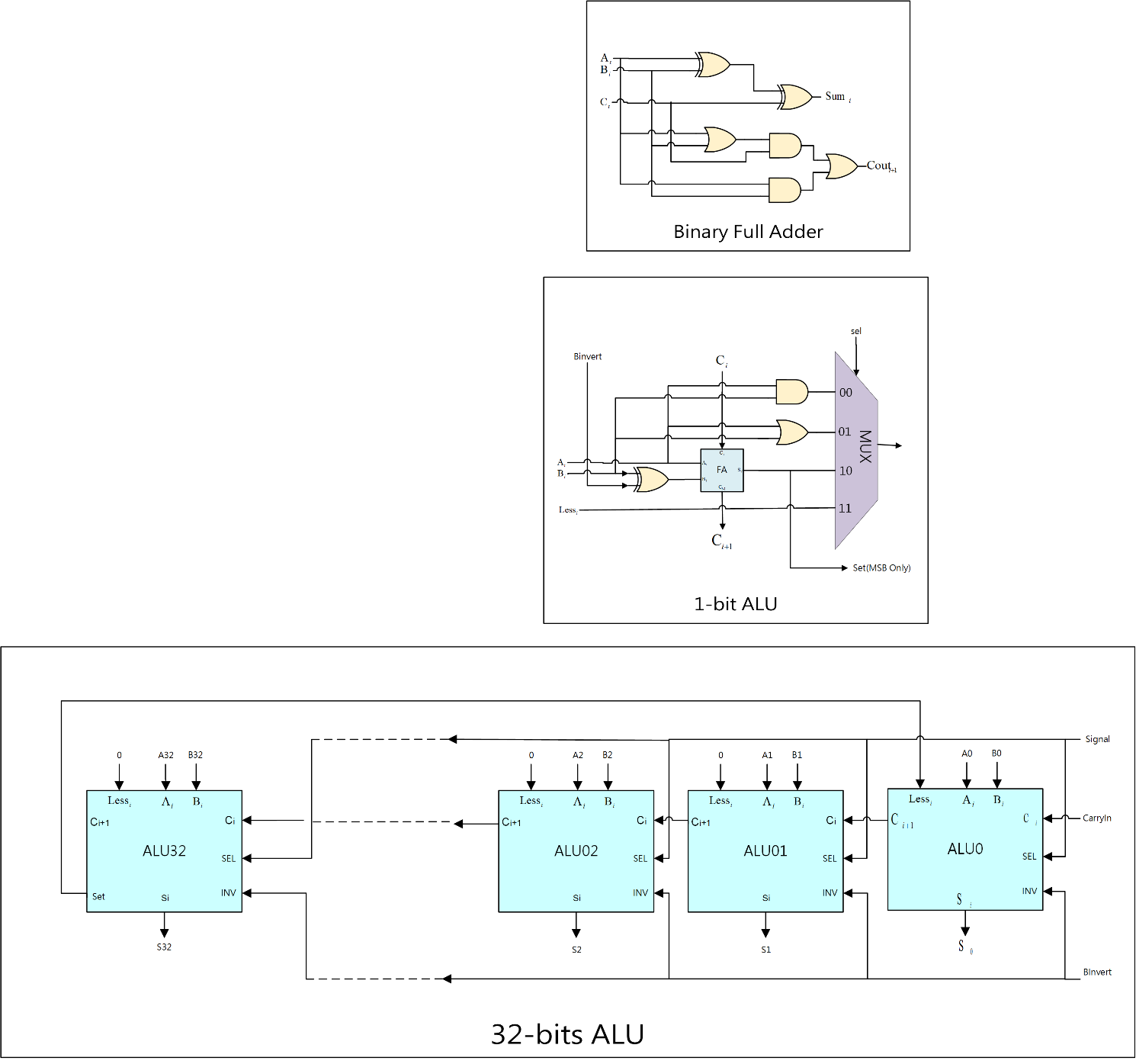
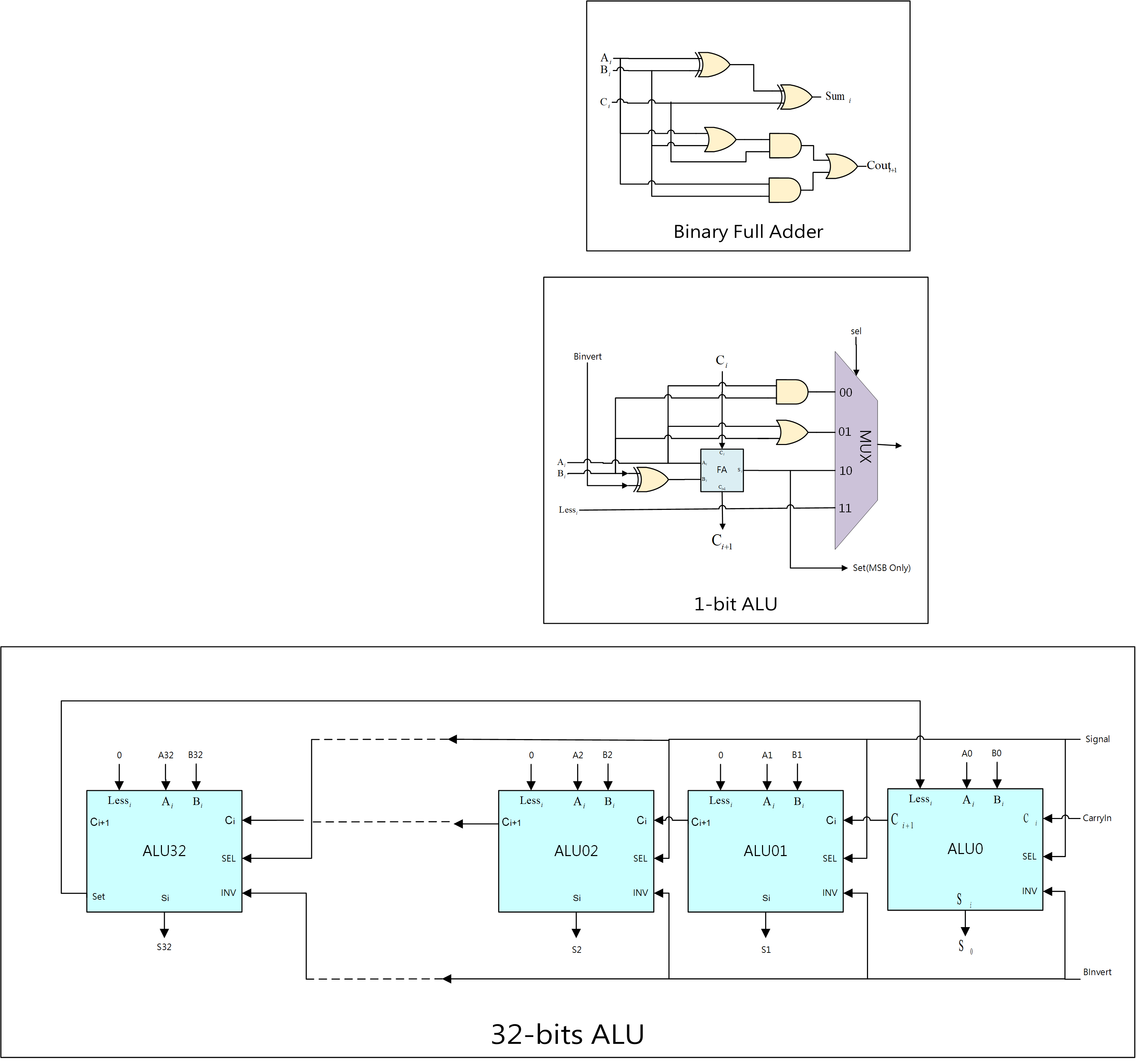
SUB : 110 / 34 / 100010

SLT : 111 / 42 / 101010

SLL : 011 / 0 / 000000

* AND : 000 第一位元為Bit Invert，二、三位元為Selection，36則為指令代碼，100100則為6-bits Signal，為36轉binary後的值，我們讓他1 bit 1 bit 做AND直到32bit結束，OR也是一樣的操作。
* ADD : 010 第一位元為Bit Invert，二、三位元為Selection，32則為指令代碼，100000則為6-bits Signal，為32轉binary後的值，設計ADD、SUB時我們都是使用Full Adder，差別在於Binvert，從AND、OR、ADD、SUB、SLT的Signal可以發現，只有SUB、SLT的Signal[1]為1，因此我們利用Signal[1]來做判斷，決定Binvert是0或1。
* 使用SUB、SLT時，兩指令都會有相減的動作，因此我們設計了一個BinvertCarry，用來當成第一bit的ALU的輸入，若指令為SUB、SLT，BinvertCarry就設定為1，其餘指令為0。
* SLT會利用SUB判斷兩數的關係，若兩數相減大於等於0，SUB結果的最高位元為0，反之，兩數相減小於0為1。最後會將此最高位元設定成輸出值的最低位元。
* SLL是使用Midterm Project寫好的Shifter放入ALU，實現邏輯左移的功能，ALU再依據輸入指令來決定是否要輸出SLL結果。



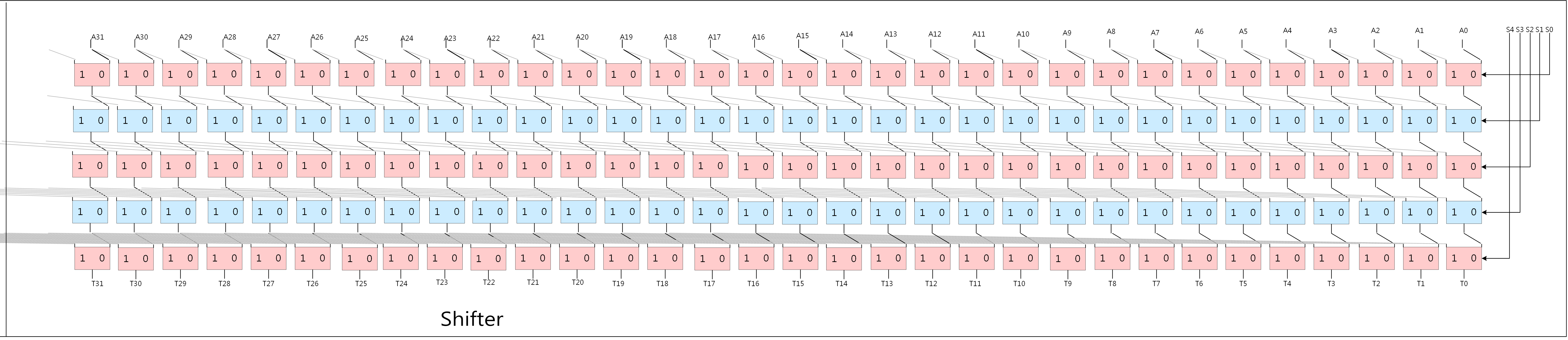


1. **Shifter(組合邏輯)：**
2. **設計重點**

沿用MidTerm Project的Shifters，主要目的還是實現SLL，邏輯左移的運算，用Shamt位移 in，將結果紀錄在out。

1. **說明**

* 用5-bits 控制位移量，因為2的5次方最多可位移31 bits，故以5-bits 做為控制位移的 ，若要位移32 bits 則全部補 0。
* 以一個5bits的線接收Shamt的前5bits判斷位移量，從第1個bits 判斷到第5個bits 依序可位移 1、2、4、8、16。



1. **Register\_File :**
2. **設計重點**

Register\_File主要在實現R-type、I-type各指令時，用來存放的暫存器。

1. **說明**

* 我們依照講義上給的圖下去設計Register\_File，R-type指令，需要用到RN1、RN2、WN三個input，RN1、RN2分別讀入rs、rt暫存器的值，並將RN1值->RD1，RN2值->RD2，之後進入ALU進行運算，之後運算結果傳回WD，WD在寫回WN暫存器(rd)。
* I-type指令則有分三類，因為I-type指令只有兩個暫存器，若是lw指令，RegWrite會設為1，需要RN1、WN兩個input，RN1讀入rs暫存器的值，並將RN1->RD1，然後offset進行有號數擴充，與RD1一起放入ALU運算，WD則接收Data Memory的RD，然後將WD->WN(rt)。
* 若是sw指令，RegWrite會設為0，需要RN1、RN2兩個input，RN1讀入rs暫存器的值，並將RN1->RD1，RN2讀入rt暫存器的值，並將RN2->RD2，然後offset進行有號數擴充，與RD1一起放入ALU運算，然後RD2將值給Data Memory的WD，然後ALU將運算結果的位址給Data Memory的ADDR。
* 若是beq指令，RegWrite會設為0，需要RN1、RN2兩個input，RN1讀入rs暫存器的值，並將RN1值->RD1，RN2讀入rt暫存器的值，並將RN2值->RD2，之後進入ALU運算，offset進行有號數擴充，並在左移兩位元達到乘以4的效果，若RD1 = RD2，則將PC+4與有號數擴充後的值相加回傳給PC，RD1 != RD2，則只將PC+4的值回傳給PC。

1. **Signal\_Extend :**
2. **設計重點**

Signal\_Extend主要在執行I-type指令時，將其16 bits的offset進行有號數擴充成32 bits。

1. **Data Memory :**
2. **設計重點**

Data Memory主要在於判斷I-type的lw、sw，是否需要寫入記憶體或從記憶體中讀取資料。

1. **說明**

* 我們照著講義給的架構圖下去設計，如果是lw指令，MemWrite會設為0，MemRead會設為1，從MemRead = 1我們可以判斷，這時我們需要將位址讀入暫存器，因此將從ALU讀到ADDR的位址(rs + offset)，透過RD寫回Register File的WD。
* 如果是sw指令，MemWrite會設為1，MemRead會設為0，從MemWrite = 1我們可以判斷，這時我們需要將位址寫入記憶體，因此我們將從ALU讀到ADDR的位址(rs + offset)，並把RD2的值給WD，之後WD將值寫入ADDR的位址。

1. **Mips\_Pipeline：**
2. **設計重點**

將各項module建立並執行，Pipeline主要將Datapath分成5-stage，分別為IF、ID、EX、MEM、WB，我們設計4個暫存器，分別為IF\_ID、ID\_EX、EX\_MEM、MEM\_WB，利用這4個暫存器達到永遠可讀，特定時間可寫的效果。

1. **說明**

我們依照講義給的Datapath下去設計，第一階段為IF，主要在做的事為Instruction Fetch，從Instruction Memory中讀取指令，並將PC+4，將IF階段的結果存在IF\_ID暫存器裡。下一階段為ID，主要在進行Instruction decode，從ID\_IF暫存器取出IF的結果，將指令解碼，判斷指令是I-type、R-type、J-type，將ID階段的結果存在ID\_EX暫存器裡。下一階段為EX，主要在做的事為Execute，從ID\_EX暫存器取出ID的結果，執行指令的運算，若指令為BEQ，則在此階段會判斷結果，若是相等，Zero = 1，則會在此階段將PC+4與offset擴充偏移完的值相加回傳給PC，若不相等，則回傳PC+4給PC，運算完後將EX階段的結果存在EX\_MEM暫存器裡，下一階段為MEM，主要在做的事Memory Access，從EX\_MEM暫存器裡取出EX的結果，看是否要進行存取，將結果存入MEM\_WB暫存器裡，下一階段WB，主要在進行Write Back，從MEM\_WB取出結果，利用多工器判斷是否要寫回。

1. **ALU Control**

ALU Control主要在控制ALU，以輸入的input決定訊號，再依訊號決定執行哪項功能。

1. **Control Unit**

Control Unit會進行解碼，判斷是R-type指令還是I-type指令，決定各個多工器的訊號為1還是0、以及給ALU Control一個ALU OP。

ALUOP / Ctl / Ctl轉二進位

* R-type : 10 / 0 / 000000

SLTI : 11 / 18 / 010010

LW : 00 / 35 / 100011

SW : 00 / 43 / 101011

BEQ : 01 / 4 / 000100

J : 01 / 2 / 000010

JAL : 01 / 3 / 000011

1. **2-1 MUX(組合邏輯)**

接收Control Unit給的訊號。

1. **NOP指令**

* 為了避免hazard，在遇到J-type指令、BEQ、連續指令使用到相同的暫存器，這時需要給NOP，避免hazard的情況。
* 指令輸入00000000，CPU不做任何事。

1. **TB\_Pipeline**
2. **設計重點**

TB\_Pipeline主要目的為一個測試平台，從Testbench輸入，驗證各功能的運算結果是否正確。

三、結果

1. add $s1(r17,2), $s0(r16,1), $s1, 0

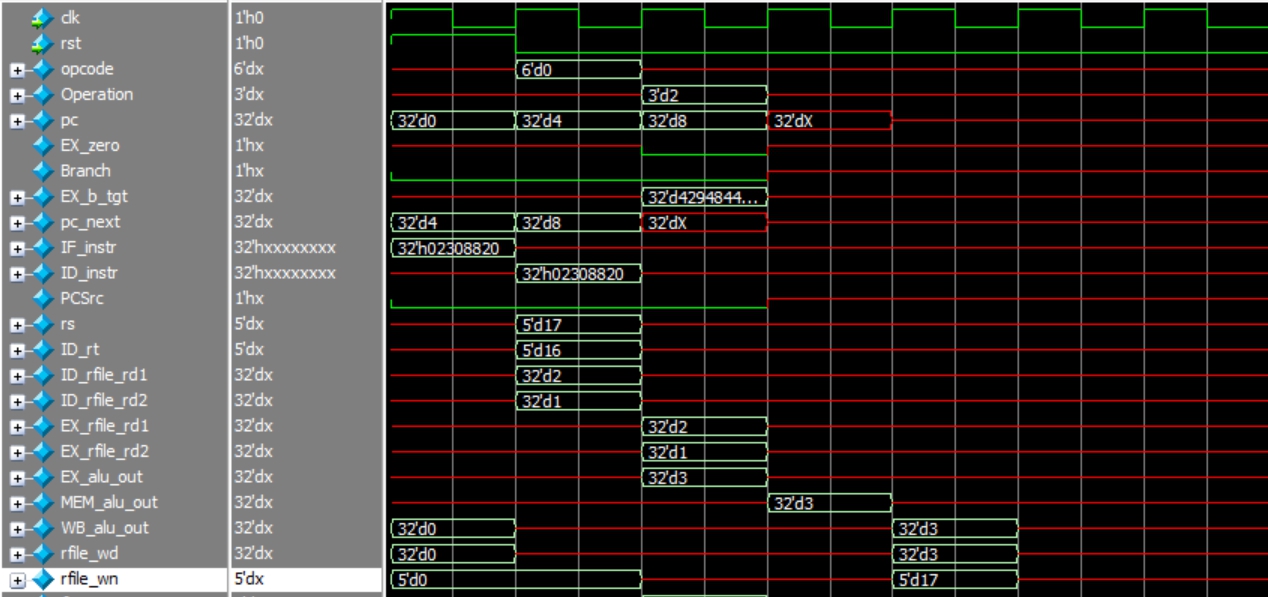
(rs) (rt) (rd)

20

88

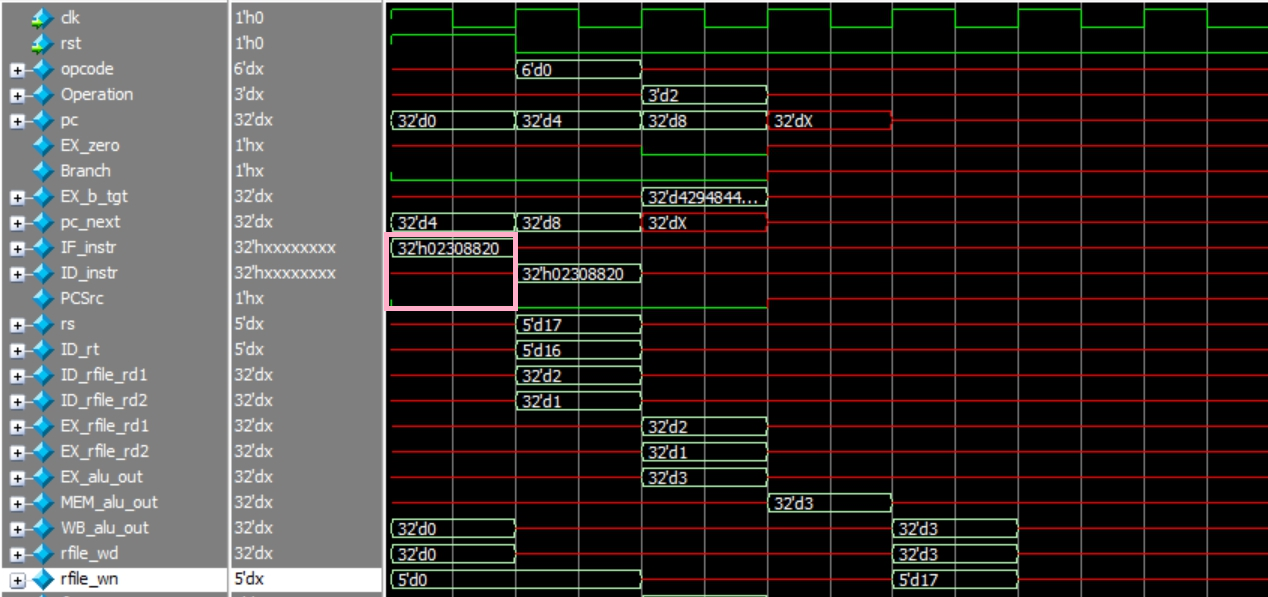
30

02



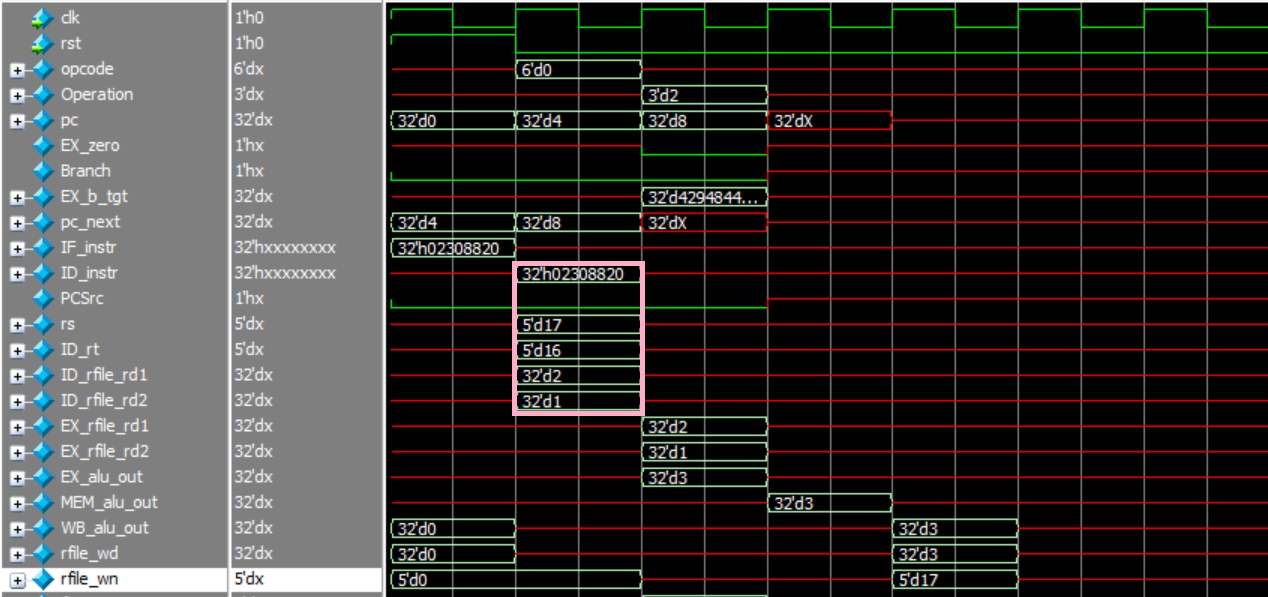
電腦讀取會以Little Endian，因此會顯示02308820。

第一個cycle先Fetch指令(粉紅框框的位置)。

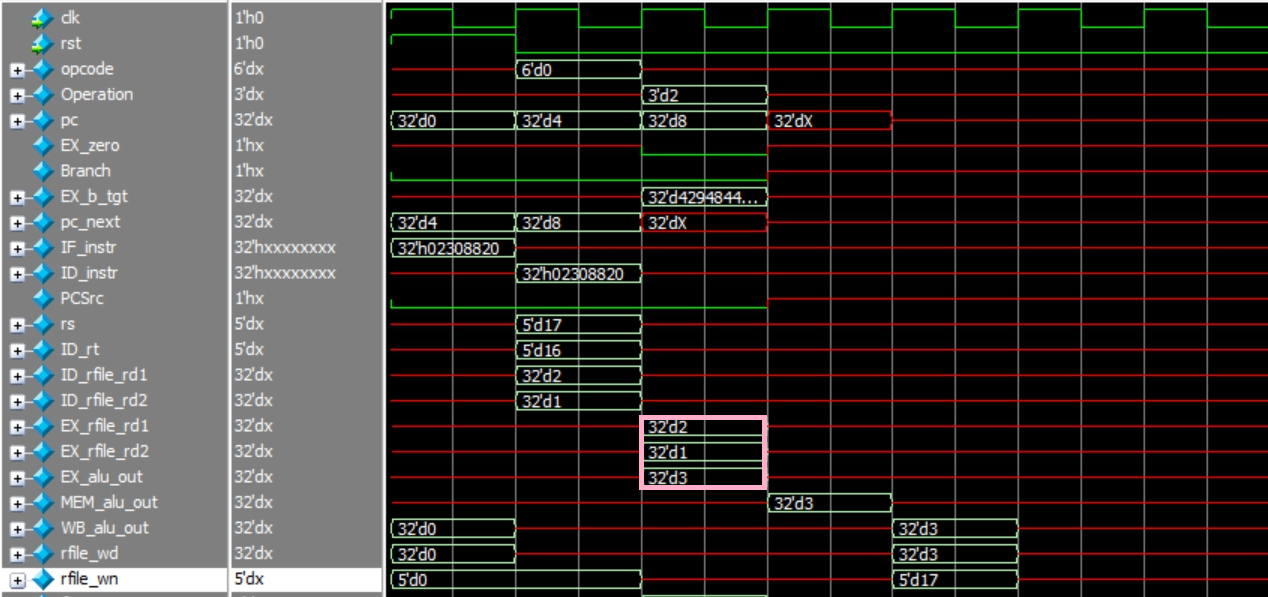


第二個cycle會將指令解碼，找到rs = 17、rt = 16兩者為暫存器的位置。

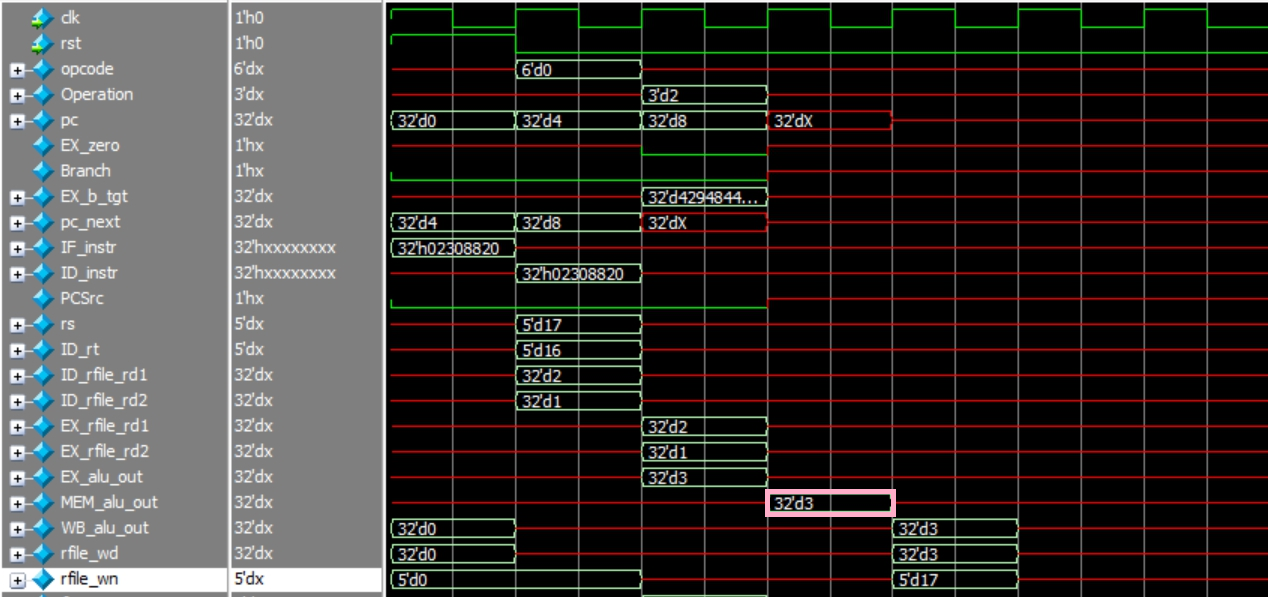
rd1 = 2、rd2 = 1，兩者為暫存器的值。



第三個cycle會將rd1、rd2放入ALU進行運算。



第四個cycle，由於R-type不需要經過Data Memory，但還是需要傳遞。



第五個cycle，要將ALU計算完的值Write Back給rd暫存器，這時rd暫存器

的位置為17，值為3。



因為R-type指令的運作幾乎相同，只差在ALU運算的結果，因此我們以

add為例。

2. lw $s1(17,2), $t7(15,21), 0

00

00

2F

8E

// nop

00

00

00

00

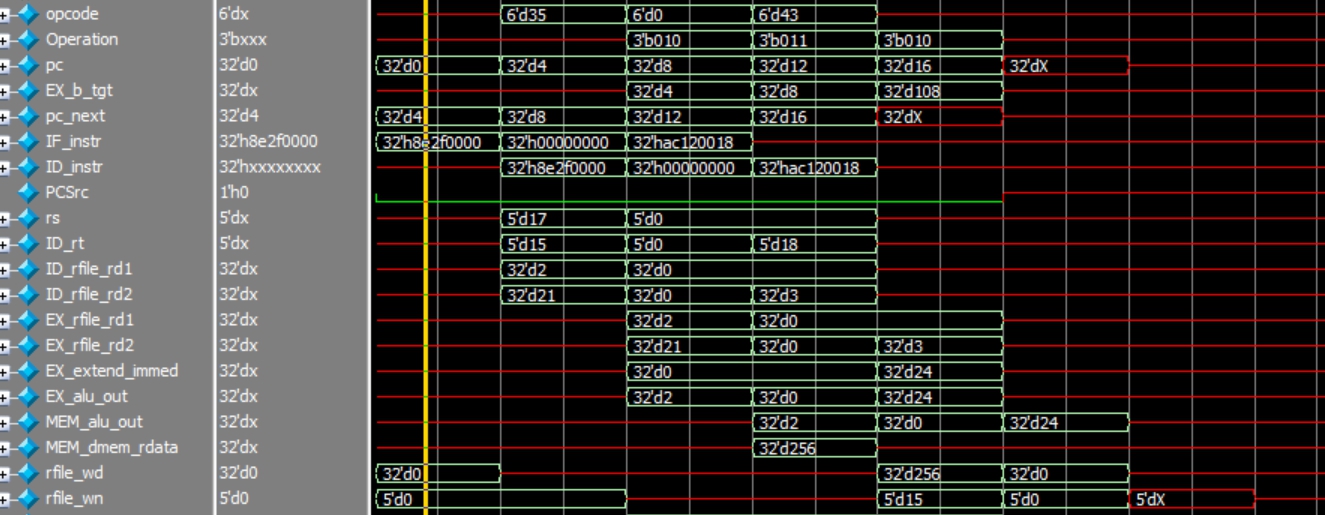
sw $zero, $s2, 24

18

00

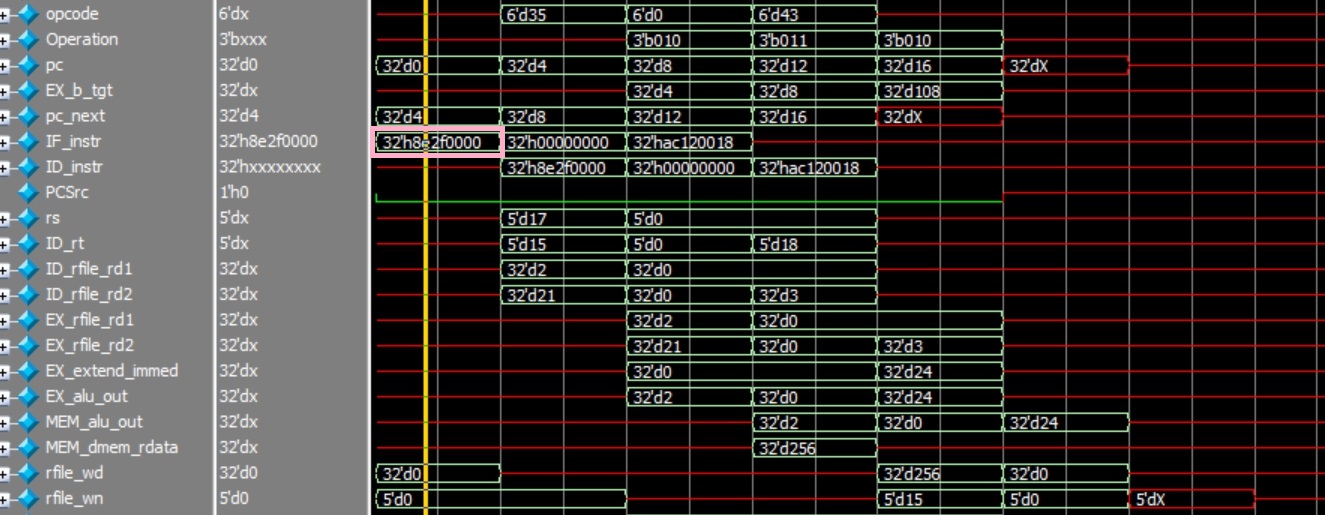
12

AC



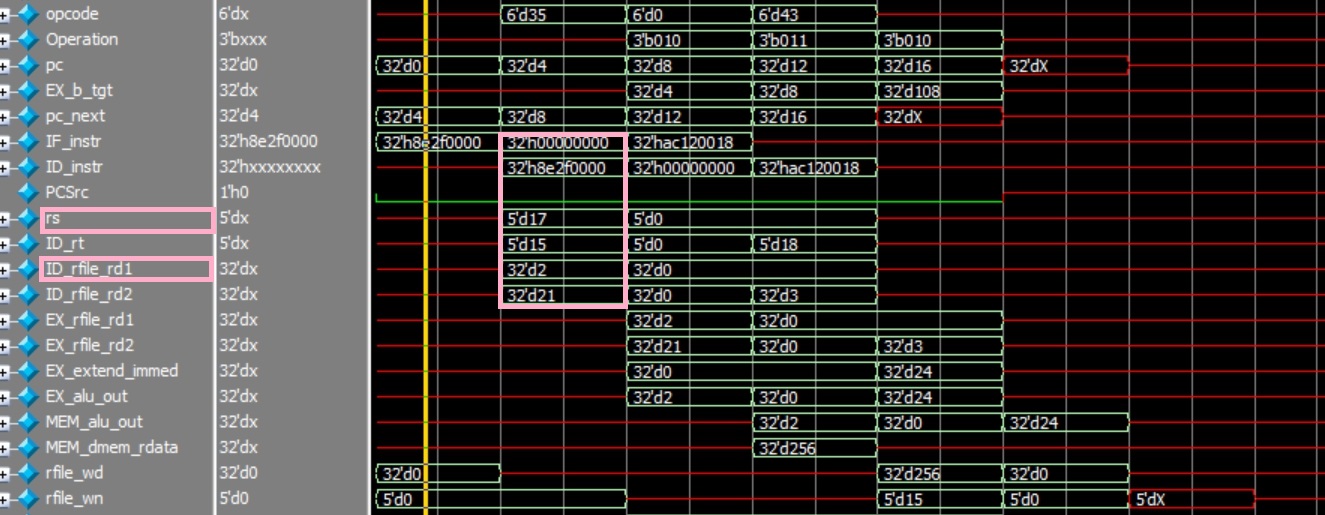
我們把這三個指令的cycle一起分解。

第一個cycle先fetch指令(LW、粉紅框框的位置)。



第二個cycle將LW指令解碼，並fetch NOP指令，因為I-type只有兩個暫存

器，而LW只用到rd1，因此我們這時只需要看rs = 17的位置，rd1 = 2的值。



第三個cycle LW進行運算，並fetch SW指令，我們給的立即值為0，因此

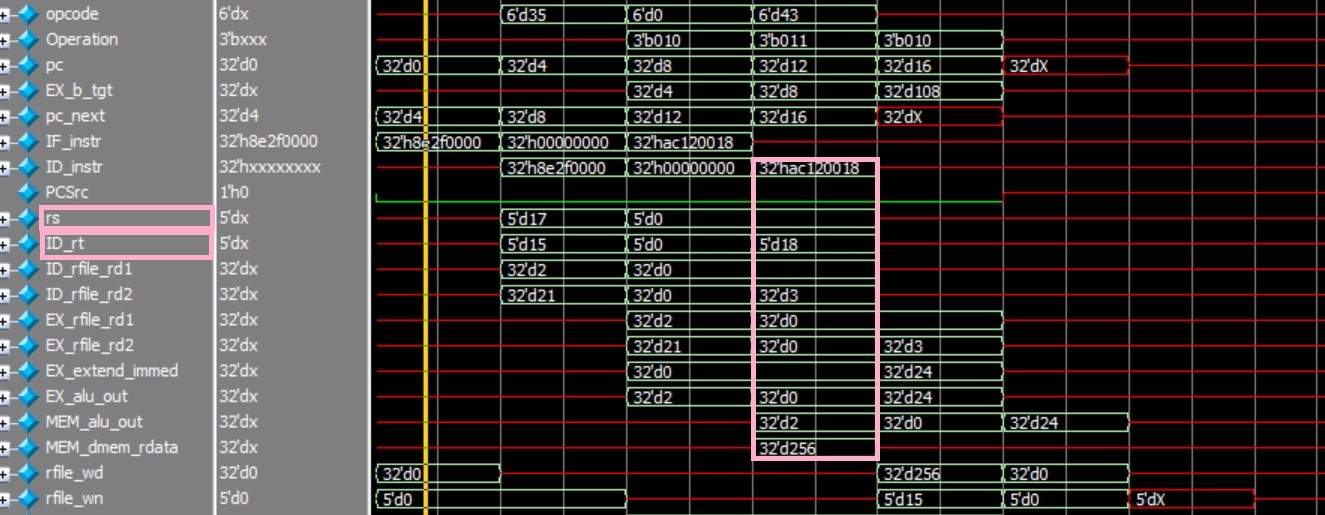
ALU進行運算後的位置為2。



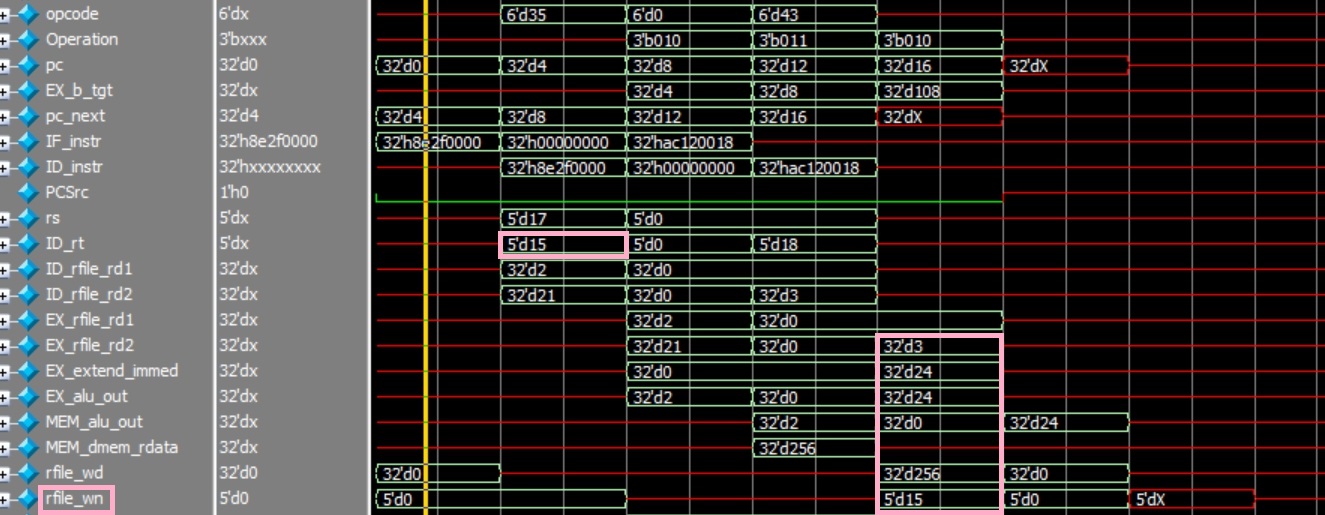
第四個cycle SW進行解碼，SW用到的是rs、rt暫存器，rs = 0、rt = 18，rd1 =

1. rd2 = 3，這時LW正在進行Memeory Access，我們從ADDR讀到暫存器的

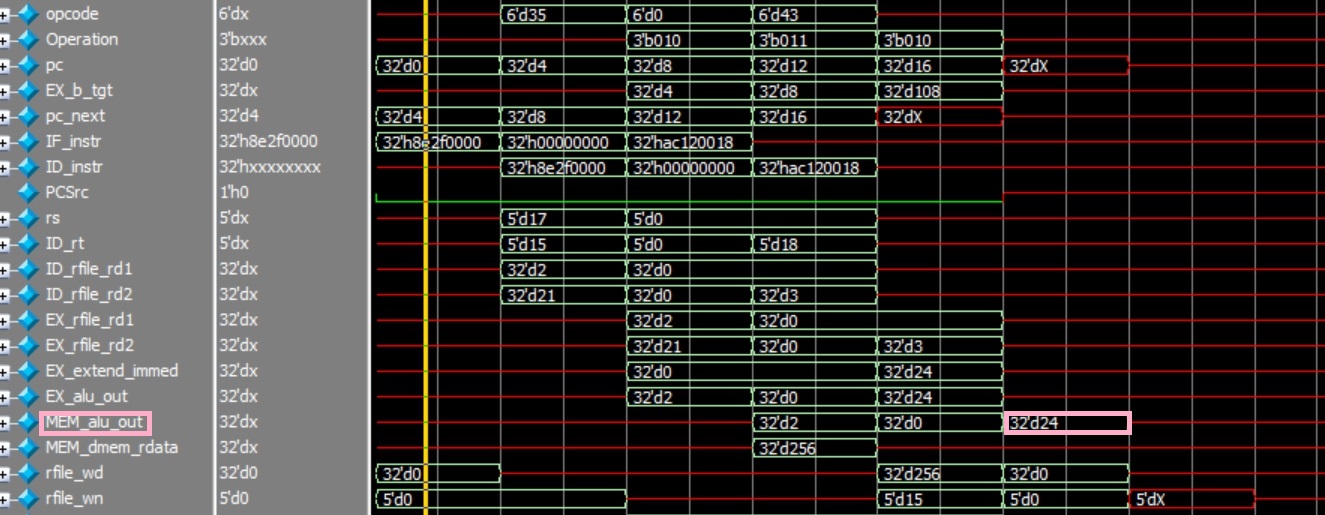
位置 = 2，得到的記憶體的資料 = 256。



第五個cycle SW進行運算，SW的立即值為24，因此ALU計算後的位置為24，LW將計算後暫存器Write Back回Register file的WN(rt)。



第六個cycle SW進行Memory Access，我們在Data Memory裡將rd2的值 = 3(wd)寫入暫存器(addr)的位置。



3. slti $s4, $t8 #16

10

00

98

4A

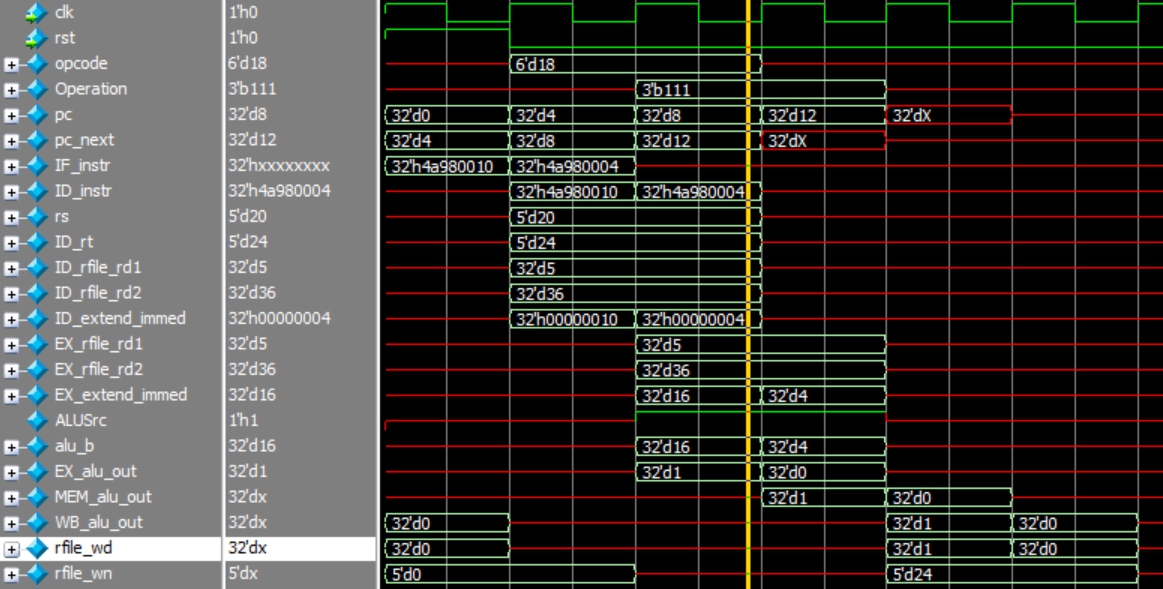
slti $s4, $t8 #1

04

00

98

Slti指令與R-type指令差不多，只差在ALUSrc多工器時讓偏移量進入比較。



4. beq rs rt imm

beq s1, s1,#5

05

00

31

12

nop

00

00

00

00

nop

00

00

00

00

add $s2(r18,3), $s0(r16,1), $s2(r18,4)

20

90

50

02

sub $s1(r17,1), $s0(r16,1), $s1(r17,0)

22

88

30

02

sub $s3(r19,4), $s0(r16,1), $s3(r19,3)

22

98

70

02

or s4(r20,5), s0(r16,1), s4(r20,5)

25

A0

90

02

add $s2(r18,4), $s0(r16,1), $s2(r20,5)

20

90

50

02

sub $s1(r17,0), $s0(r16,1), $s1(r17,-1)

22

88

30

02

sub $s3(r19,3), $s0(r16,1), $s3(r19,2)

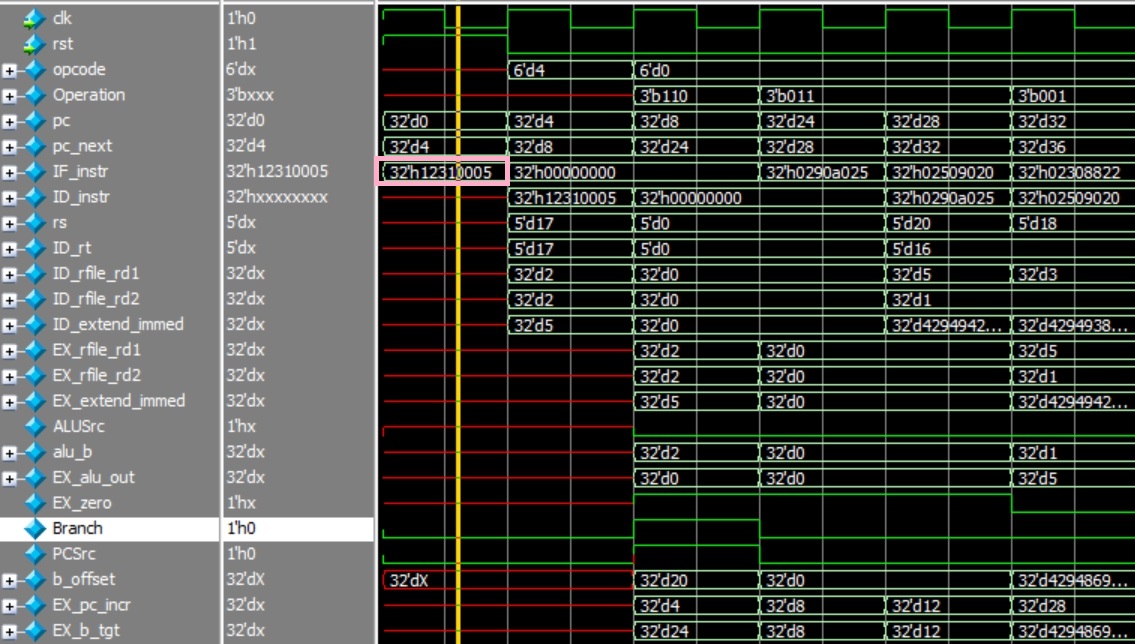
22

98

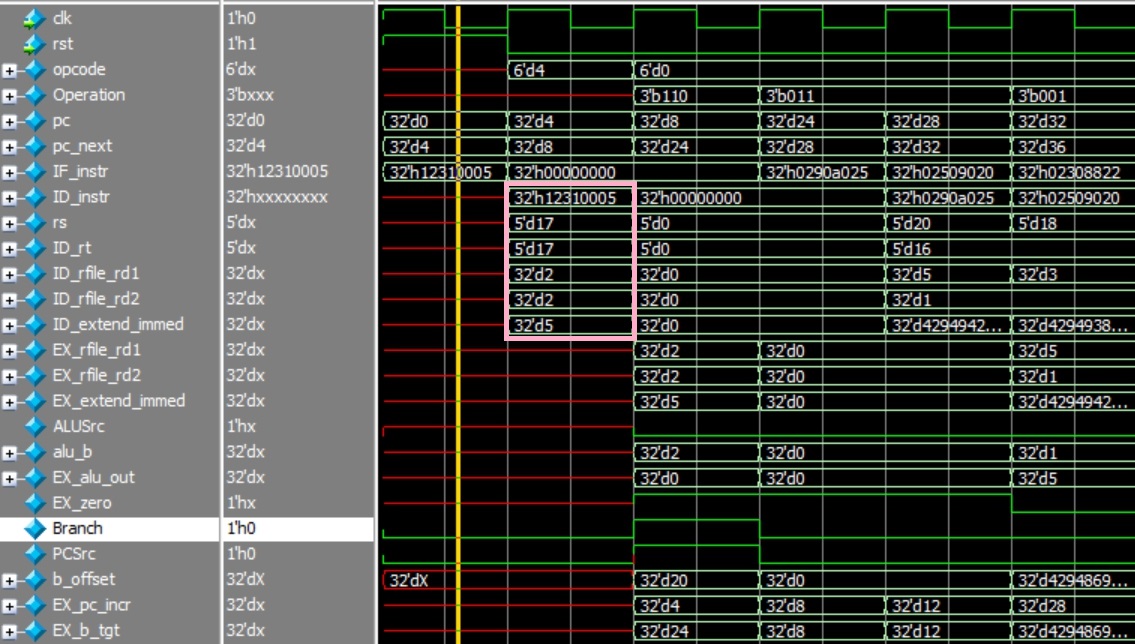
70

02

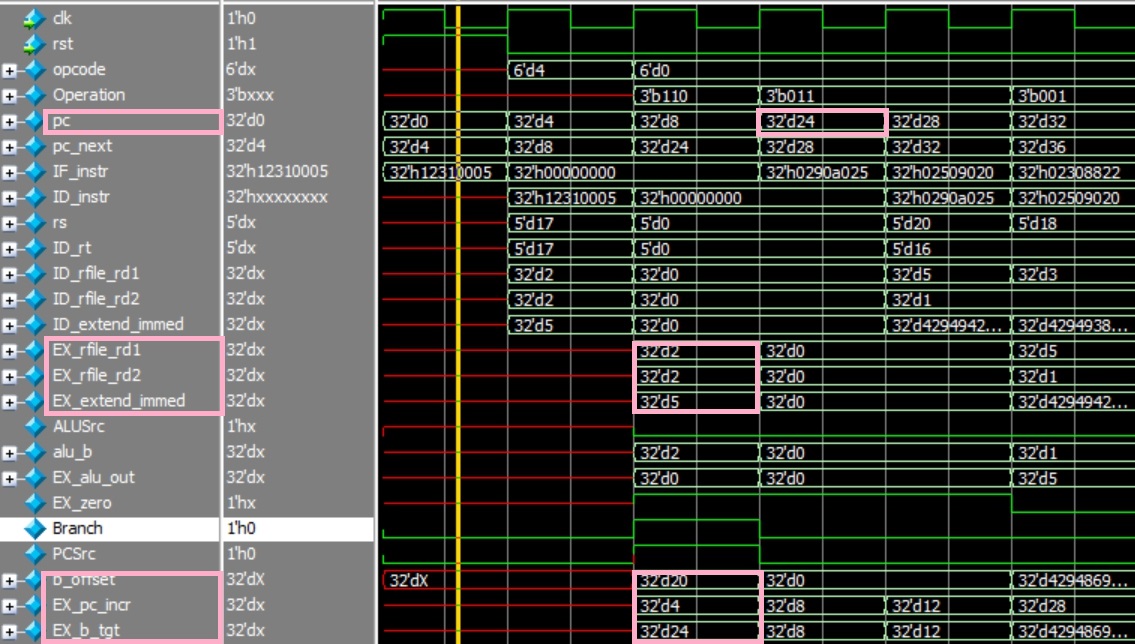
BEQ指令第一階段先fetch指令(粉紅框框的部分)



第二階段進行解碼。



第三階段進行運算，判斷rd1、rd2的值是否相等，由圖可知相等，偏移量 = 5，因為PC計算increment單位為4，所以offset為20，原本PC為0，將offset與PC+4相加，得出24，回傳給PC。



5. sll $t0(rd,r8,8), $t1(rt,r9,9), 2 => ans = 36

80

40

09

00

lw $s1(17,2), $t7(15,), 0

00

00

2F

8E

sub $s2, $s0, $s2

22

90

50

02

12 add $s1, $s0, $s1, 0

20

88

30

02

jal 7

07

00

00

0c

nop

00

00

00

00

nop

00

00

00

00

add $s2(r18,3), $s0(r16,1), $s2(r18,4)

20

90

50

02

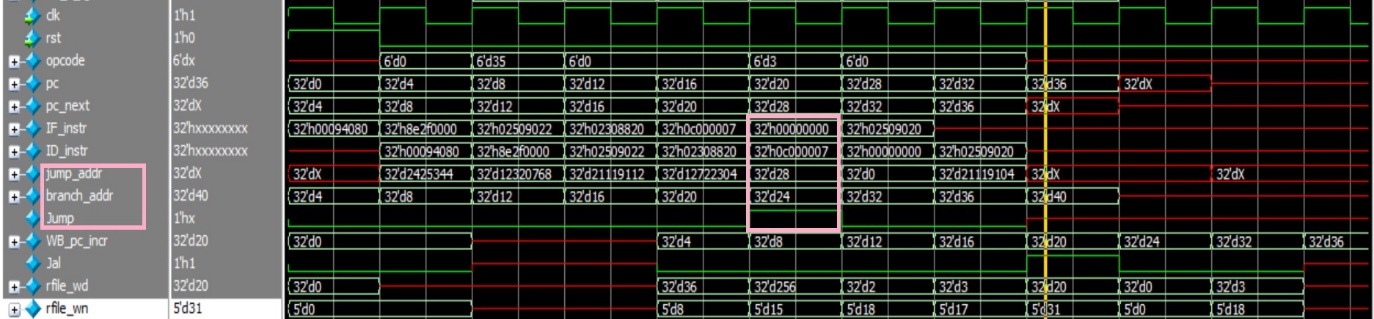
下面是J和Jal都會做的事。

當第二個cycle時，會把instrc[25:0]左移兩位元得到28位元的jumpoffset

再把pc+4[31:28]與jumpoffset做Concat得到jumpaddr

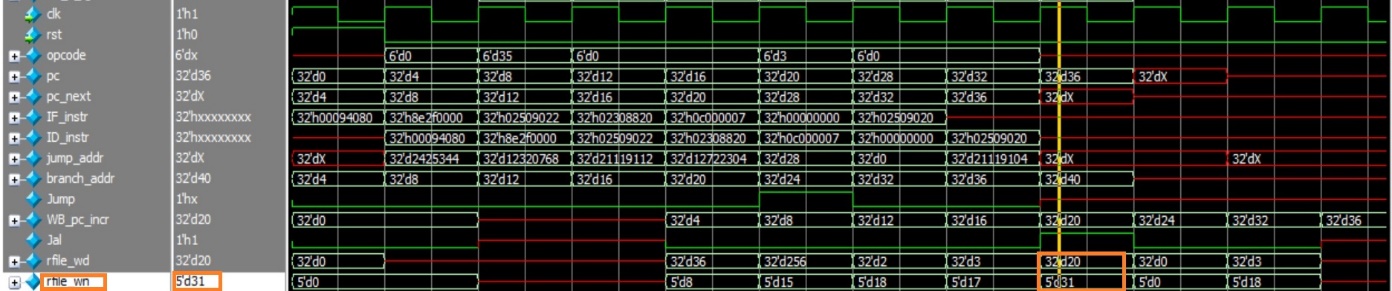
同時會把 control Unit會傳出Jump指令給JUMPMUX去選擇

最後設定PC\_next。



Jal在跳躍的同時會把Jal的下道指令PC值儲存到r31暫存器。

當第五個cycle時，MEM\_WB暫存器會傳給WB Jal和PC+4，Jal為1將寫回的值設為PC+4，同時將寫回的暫存器設為31。



四、討論

經過了Midterm Project的練習，這次在討論上思路就比較清晰，也比較有方向可以進行，Final Project的部分元件，是利用的期中寫好的東西，而Datapath、切pipeline等比較新的東西，就要從老師上課所教的、講義給的架構圖去慢慢討論、思考，一步一步完成Final Project。

五、結論

我們的結論是相比於Midterm Project，我們對於在遇到比較需要思考的問題，像是要建立在clock時序邏輯上來實現循序邏輯、blocking 跟 non-blocking的使用時機、從軟體的角度轉變成從硬體的角度等等，都比較有概念可以去撰寫，但在實現循序邏輯這方面，還是會遇到些許的小瓶頸。而這次面對Pipeline這個新的東西、難度較高的問題，我們也是在花了許久時間，從上課吸收老師上課內容、理解講義上的架構流程等等…，然後嘗試著把Final Project的功能一一實現。而遇到比較小的問題像是對於Verilog語法的不熟悉，可能接觸C、C++的時間較長，有時候還是會不小心用到C的語法、還有在Verilog撰寫時接線的部分，有時沒接好，導致輸出的waveform出現紅線…，而隨著我們練習的次數增加，就能一步步改正上面提到的小錯誤。

* 心得

雖然經過了Midterm Project的磨練，但我們這次在寫Final Project時還是遇到許許多多的難題，因為這次要寫的是一整個CPU，因此要考慮到的東西變多、也變得比較複雜，老師在上課有說到，畫圖是設計硬體的第一步，也從這句話能了解到Datapath的重要性，幸好老師在上課時，對於Datapath這一段有一再地強調、重複，因此對於各元件功能的Datapath，我們就比較沒有那麼不熟悉，比較好掌握，也比較好實現，然而像是Pipeline這東西就讓我們困擾了很久，在聽老師在上課時的講課內容時，雖然能理解Pipeline如何運作，還有怎麼實現的例子等等，但在實際撰寫Verilog程式碼時，還是遇到很多問題，像是實現Pipeline的概念，在不同stage同時做不同指令，節省時間以提升效率的時候，會因為沒有抓到前幾個stage的指令，然後就會發生一些像是在執行後面階段時，前面的可能無法fetch到指令，幸好後來與同學交流討論後解決了。最後，很開心這學期在這計組這門課，學到了非常豐富的知識，雖然遇到很多困難、錯誤，但也因為這些錯誤，讓我們有更多練習機會，也學習的更深入，再經過兩次的Project後，也讓我們更加理解這門課程的學習內容。

六、未來展望

終於完成這學期計組的兩次Project，從一開始對這門課感到畏懼，到最後寫完、完全了解這門課的學習內容，經歷了很多挫折，解決了很多困難，雖然很辛苦，但這些辛苦都很值得，也透過計組這門課，點燃了我們對硬體領域的些許興趣，希望之後能在計組這領域繼續學習更多知識。雖然這學期了解的並沒有到很扎實，但還是希望計算機組織能在未來成為我們的一技之長。

七、分工

10527130 陳少洋：部分程式碼、Debug

10527140 初　元：架構圖繪製、Debug

10527132 林亞吟：程式碼、報告結果討論、Debug

10527135 張智欽：撰寫報告分析、報告結果討論、Debug