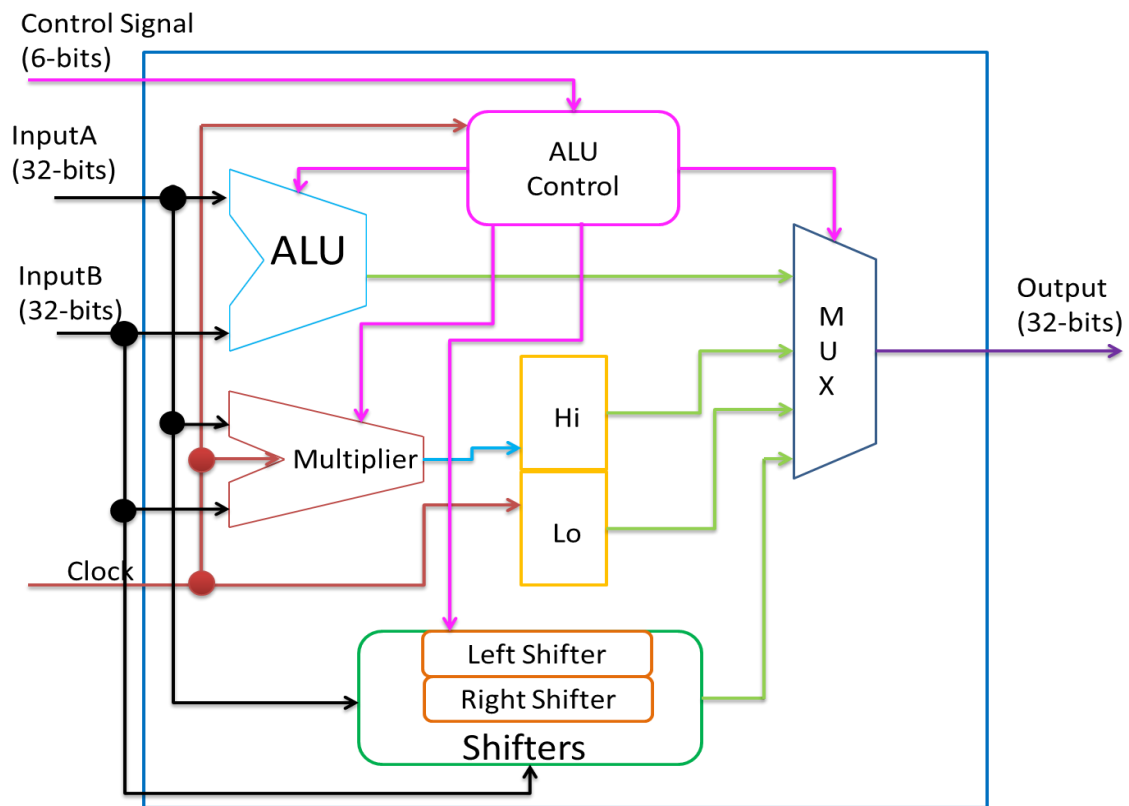


計算機組織期中 Project

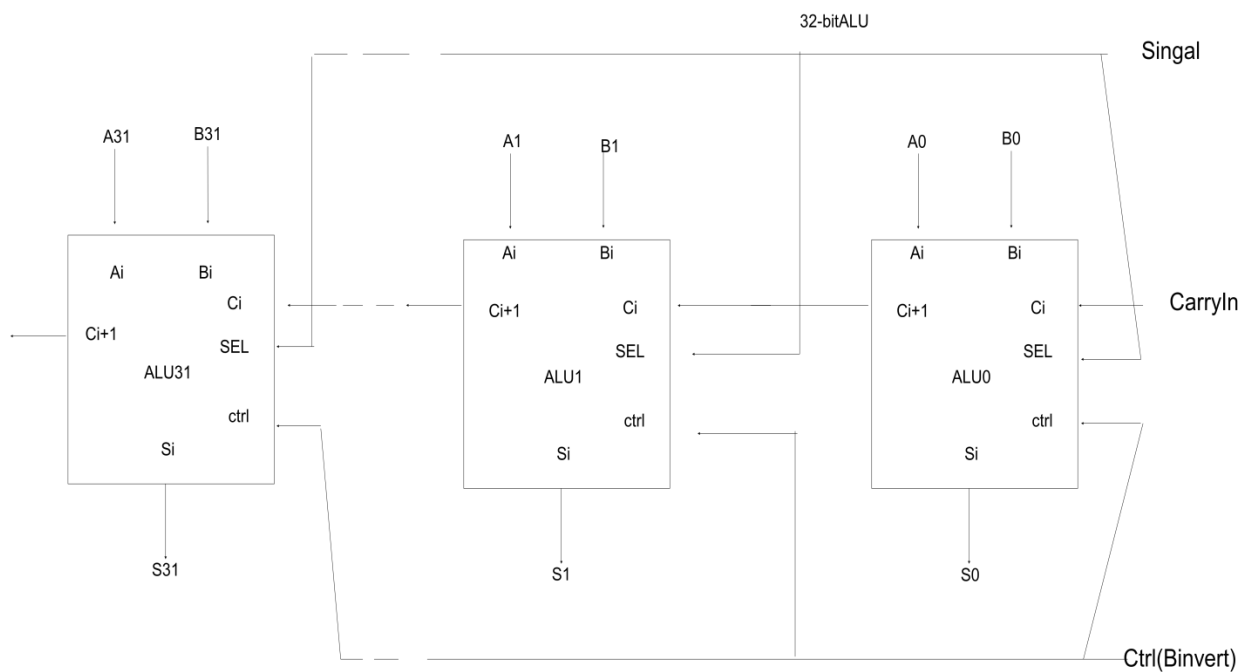
Big-ALU 設計

一、Datapath 與詳細架構圖

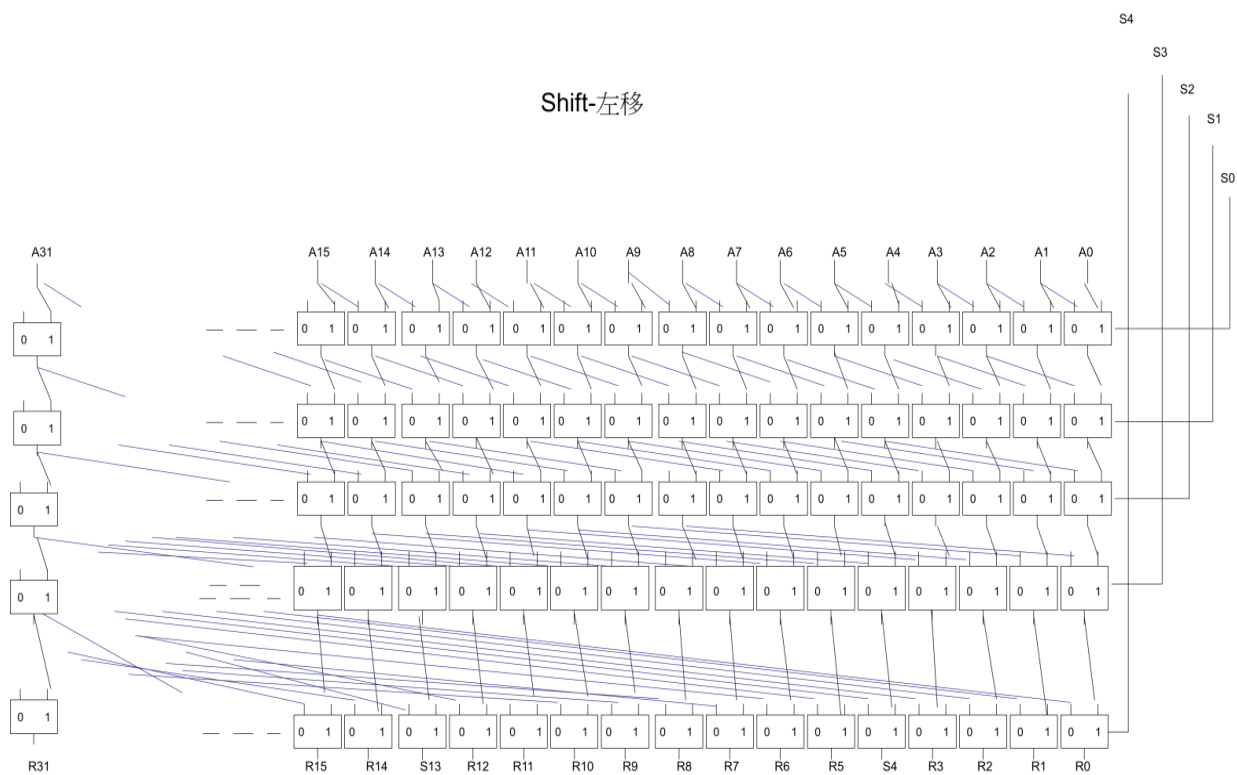
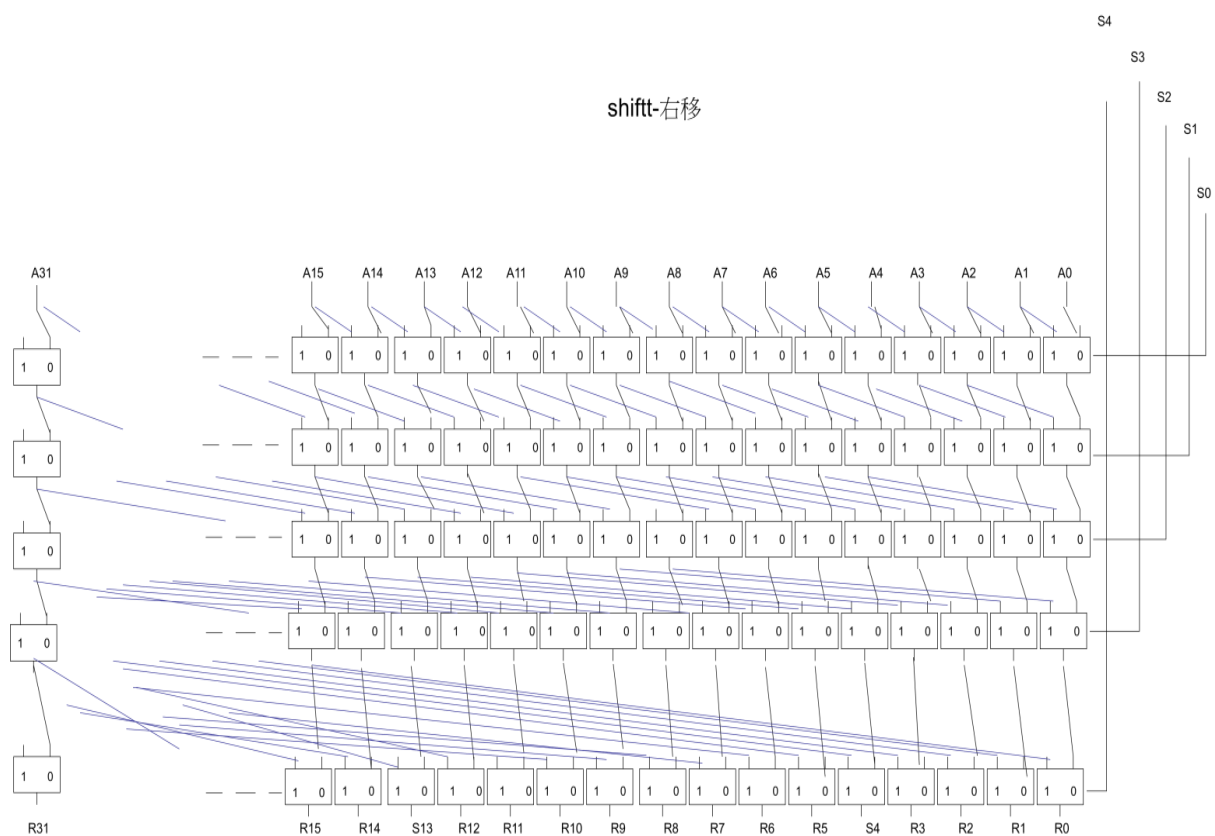
大 ALU:

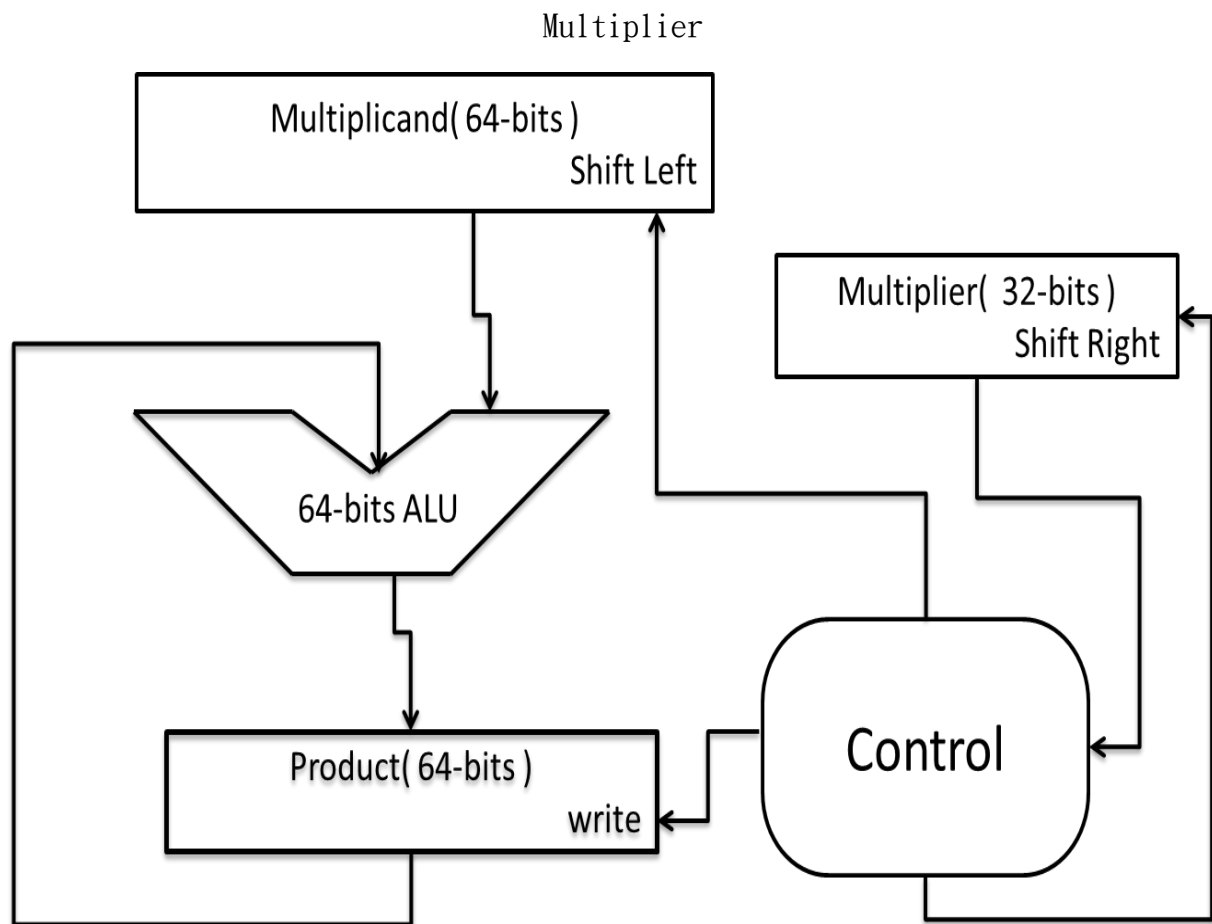


ALU(ADD, SUB, SLT):



SHIFTER:





二、設計重點說明

ALUControl:

接收 input 的訊號, 並且把訊號傳給 Mux, ALU, Shifter, Multiplier

AND:

呼叫內建的 and module 去做運算

OR:

呼叫內建的 or module 去做運算

ADD & SUB:

用 Fulladder 去做, Add 時, ctrl==0, SUB 時, ctrl==1, 第二個 input, 需與 ctrl 做 XOR

SLT:

利用 SUB 的結果, 取最高位元來判斷是大於還是小於, 如果 $input1 < input2$, 結果是 1, 反之則為 0

ALUbit:

把 AND, OR, ADD, SUB, SLT 一個 bit 一個 bit 做, 然後用訊號來決定 output 是誰

ALU:

利用訊號決定 cIn, 和 ctrl, 然後呼叫 ALUbit 32 次, 再把結果輸出

Shifter:

把要位移的數字改為 2 進位, 第一個 bit 的數字是 1, 就移 2 的 0 次方, 第 2 個數字是 1 就移 2 的一次方, 第 3 個數字是 1 就移 2 的 2 次方, 以 2 的指數成長, 以此類推, 如果數字是 0 就不用位移。

Multiplier:

當信號改變時, 判斷信號是不是乘法信號, 一旦是乘法信號, 就將被乘數移到一個 64 位元的暫存器中, 成數移到 32 位元的暫存器中, 並將乘積設定為 0, 最後加上一個計數器, 用來計算現在目前算到哪一個位元。一旦 clk 敲響, 就判斷是否需要歸零, 如果不需要則將乘數的第 0 個位元拿來看是不是 1, 如果是 1, 就將目前的成績加上現在的被乘數, 如果是 0 則將目前乘積放入乘積中。每做完一次就將乘數右移 1 位元, 被乘數左移 1 位元, 並且將計數器加一, 一旦計數器大於等於 32, 就算是 clk 敲響也不會再做運算, 最後再把目前乘積指定到輸出結果。

HiLo:

接收 Multiplier 64bits 的 result, 把資料第 33~64 位元放入 32bits 的 Hi 暫存器和把第 1~32 位元放入 32bits 的 Lo 暫存器

Mux:

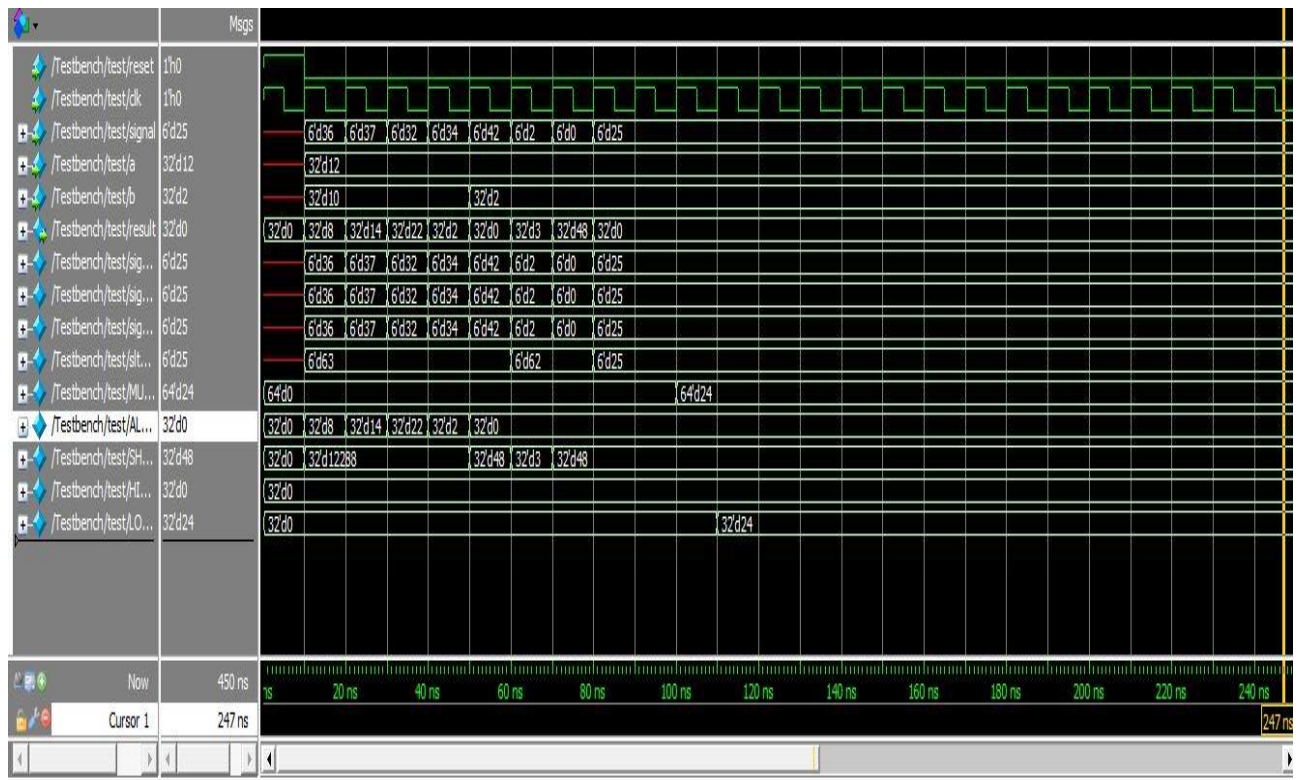
負責接收 ALU, Shifter, Multiplier 的 result, 並由訊號去選擇結果

BigALU:

把 ALUControl, ALU, Multiplier, Shifter, HiLo, Mux 整合在一起, 讓 TestBench 可以運作

三、 ModelSim 驗證結果與 Waveform 輸出圖形

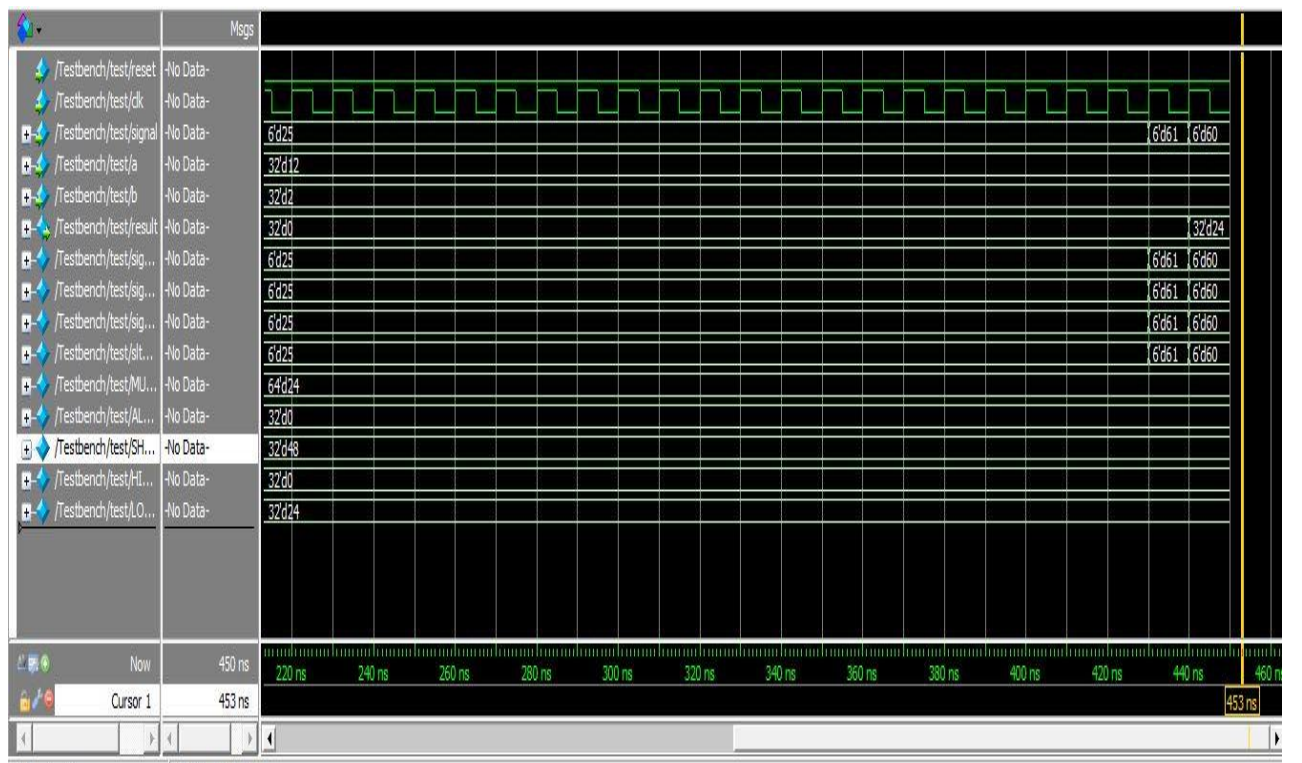
```
start
1: Input: AND(36)      12      10
2: Correct: Your answer is: 8,
   Correct answer is: 8
2: Input: OR(37)      12      10
3: Correct: Your answer is: 14,
   Correct answer is: 14
3: Input: ADD(32)      12      10
4: Correct: Your answer is: 22,
   Correct answer is: 22
4: Input: SUB(34)      12      10
5: Correct: Your answer is: 2,
   Correct answer is: 2
5: Input: SLT(42)      12      2
6: Correct: Your answer is: 0,
   Correct answer is: 0
6: Input: SRL( 2)      12      2
7: Correct: Your answer is: 3,
   Correct answer is: 3
7: Input: SLL( 0)      12      2
8: Correct: Your answer is: 48,
   Correct answer is: 48
8: Input: MULTU(25)      12      2
41: MULTU End
Move Hi
44: Correct: Your answer is: 0,
   Correct answer is: 0
Move Lo
45: Correct: Your answer is: 24,
   Correct answer is: 24
```



上面數來第六個輸出,result 是我們計算出來的結果

然後他的上面是我們得到的 inputA, inputB

再上面一個是 signal, 就是我們所要做的事情



從這個圖可以知道

Hi 暫存器中放的是 0, Low 暫存器中放的是 24