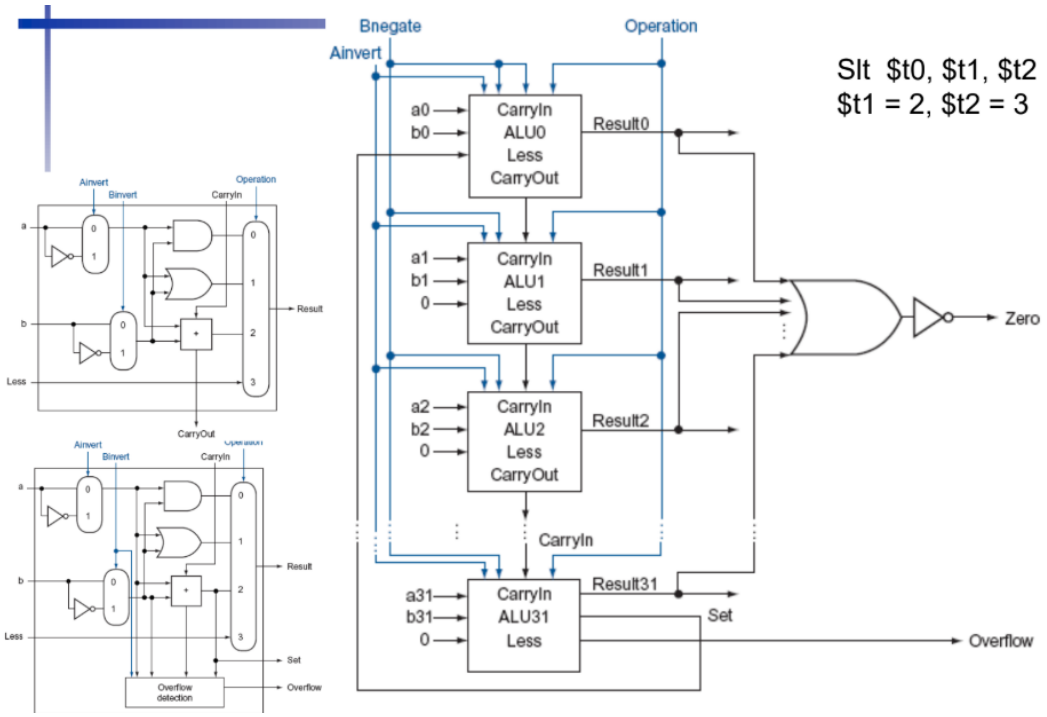
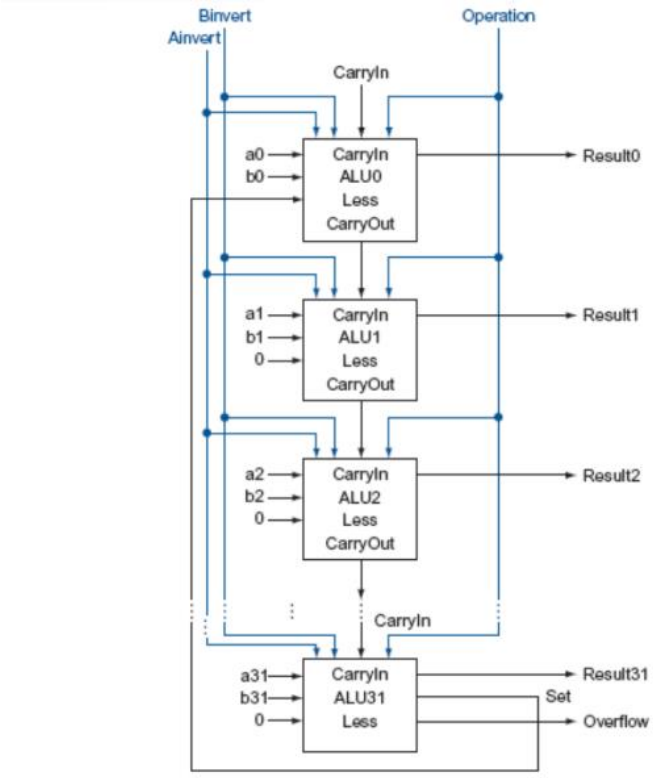
Computer Organization

Architecture diagrams:



Hardware module analysis:



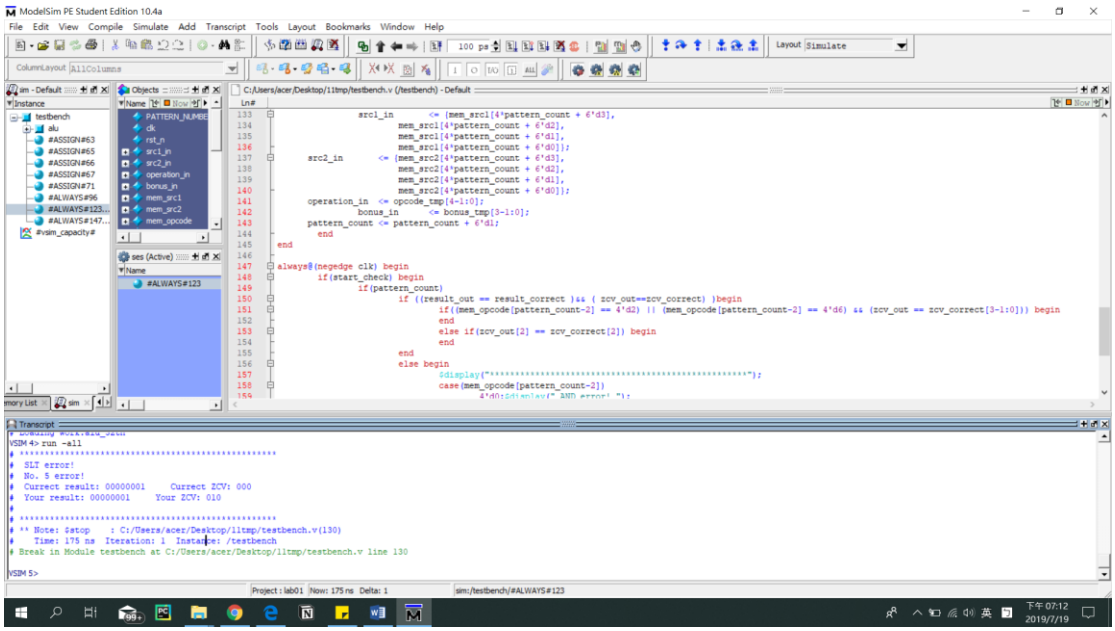
alu_top & alu_32th：

1. 透過 A_invert 跟 B_invert 來做 src1、src2 以及 ALU control 的判斷
2. 進入 always 中，先判斷要做的 operation 為何，再進一步去 assign result 的值

alu

1. 呼叫 32 次 module 來實現 32 bit alu
2. 在 always 中將上面產生的結果(result/less/overflow)輸出成 output

Experiment result:



Problems you met and solutions:

1. **在將第 32 個 ALU 傳回第 1 個 ALU 時發生錯誤**

   Sol:將 module alu_top 中的 less 設為 input 與將 alu_32th 中的 less 設為 output

2. **Overflow 值錯誤**

   Sol:把 overflow 跟 carry_out 搞混。將 overflow 修正為第 32 個 ALU 的 cin 與 cout 的 XOR。

3. **Cout 的傳值有誤**

   Sol:將 And 與 Or 與 Nor 的 cout 都設為 0，其餘改變原本在 always 外去 assign cout 的方法，而是去判斷 op code 後再去做 cout 的給值。

4. **最後產生的 result 有誤**

   Sol:將原本 alu.v 中 always 括號內再加入 cin

5. **Slt 原本完全沒有對 result 做處理，因此出來的值會錯誤**

Sol:更改為用 less 的值去做判斷，若 less 已被更改為 1，則 module 中 1bit 的 resulr 輸出為 1，其餘情形輸出為 0。

Summary:

之前沒有接觸過 verilog，因此一開始連架構都還搞不清楚該怎麼寫好。再更詳細了解 ALU 參數輸入輸出及運作原理以後，藉由摸索 modele 的模樣(上網 google)才慢慢將 ALU 的雛型拼湊出來，雖然還是遇到了很多不知該從何下手的問題，總歸這次 lab 算是讓我們熟悉 verilog 的寫法以及更清楚 ALU 的 module 長什麼樣子。