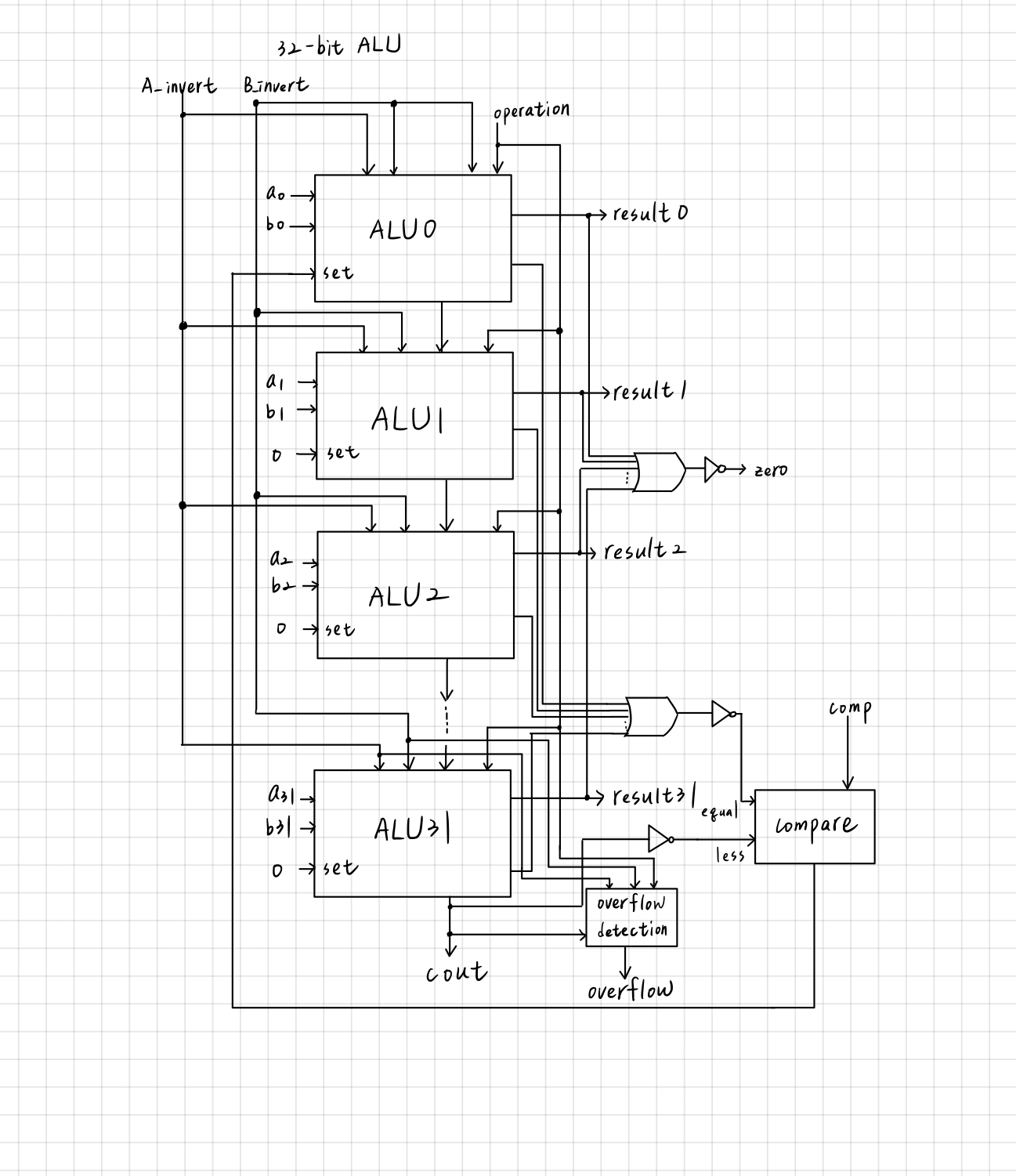
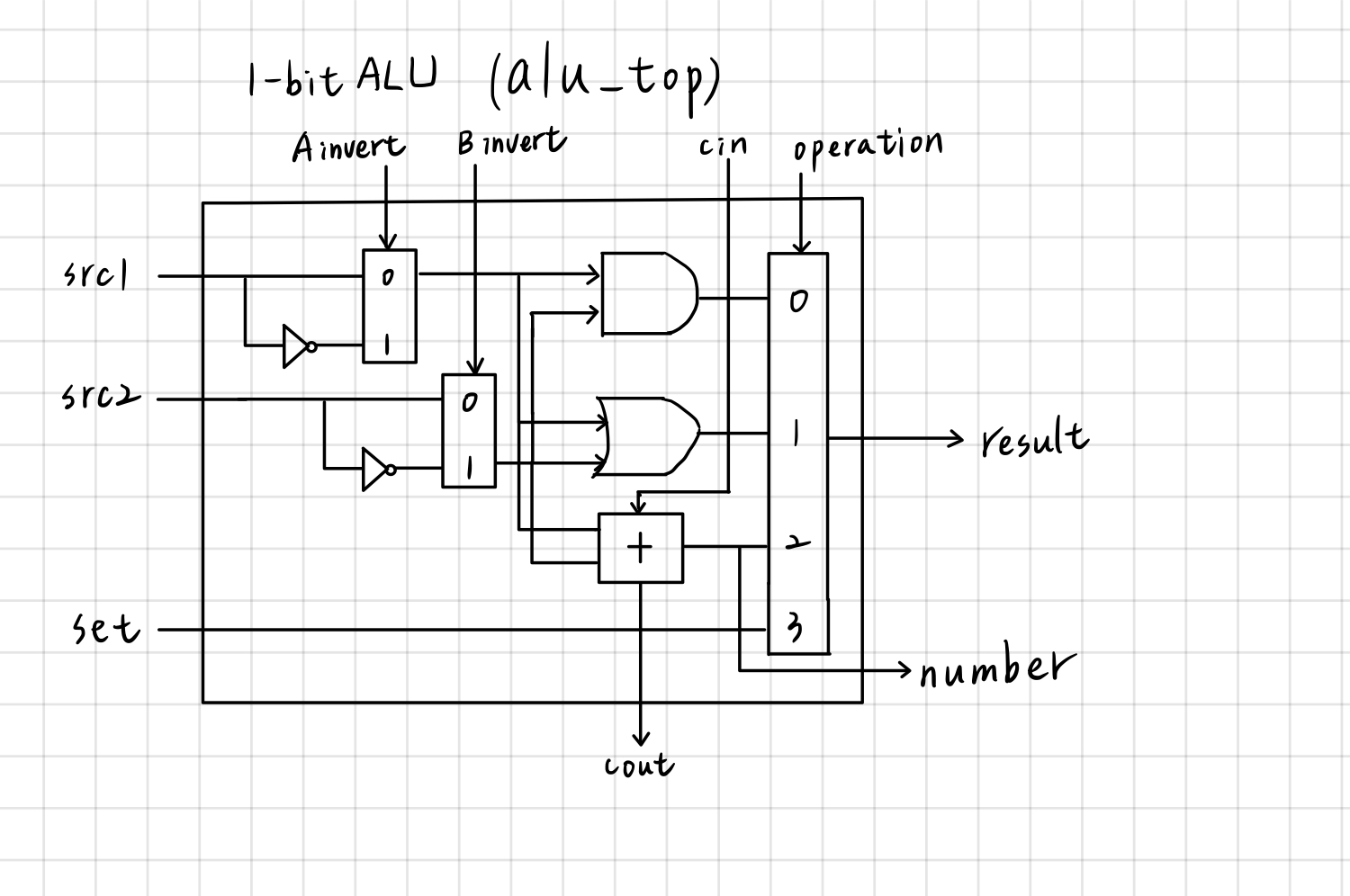
**Computer Organization**

**Architecture diagram:**



**Detailed description of the implementation:**

**alu**

我們這次主要是在alu裡用了32個alu\_top的模組完成，可以將32bit的src1與src2接進去，並作依據ALU\_control與bonus\_control作加減法、邏輯運算以及比大小等等的功能，並將結果由result顯示，除此之外，還可以判斷結果是否為0、有無overflow與輸出最後的cout功能。

**Zero**

在result全部都等於0時，會輸出1，其餘時候為0，所以在邏輯上是將全部的result放進or gate裡面再用not輸出。

**Overflow**

overflow的判斷條件是做加法與減法時才有可能是1，若operation為加法，且carryout為1時，overflow為1，若operation為減法，且carryout為0，則overflow也是1，其餘的overflow皆是0，這是unsigned number的判斷方法，可以用簡單的加減法推論出(若是signed number則是看最後一個bit的cin與cout是否相同為判斷)。

**alu\_top**

alu\_top裡面的A\_invert與B\_invert可以從ALU\_control[3:2]接，可以決定A、B是否要invert，operation則是ALU\_control[1:0]接，決定A、B的運算，可以做and、or與加法。

在alu\_top[0]裡面的cin和B\_invert接在一起，因為減法是用two’s complement實作，所以-B為~B+1，其餘的alu\_top都是由前一個的cout接，至於alu\_top[0]的set是由compare的輸出接回，其餘的set都是接0，因為比大小時只有result0會是1或0，都是result1~result31都只會是0。operation可以在最後決定要選擇的是哪條線，並送到result。

**Compare**

在compare的模組裡面，我們是將equal(A與B做完減法的結果是否為0)與less(carryout = 0 && !eqaul，此為unsigned number的結果)作為輸入，並由bonus\_control來判斷是否符合條件，輸出1或0。

**Problems encountered and solutions:**

1. **Overflow只有偵測加法**

在做減法時，因為是unsigned number的關係，所以沒辦法表示負數，但其實應該是result的結果加上負號，我們不太確定這種狀況算不算overflow，剛好測資給的減法不會發生overflow，而bonus裡面有一個小數和大數比大小，結果也沒有overflow，後來我們就寫信問助教這種狀況，得到的回覆是算是overflow所以我們就把它加上去，至於比大小時減法沒有overflow是因為比大小是可以比的，結果不可能不正確，因為不是0就是1，沒有不能表示的問題，而小減大會是負數，結果無法表示(unsigned number)，所以是overflow。

1. **Compare加在每個Alu\_top裡面，equal與less之外還有notEqaul與greater**

原本我們的Alu\_top裡面都有compare模組，導致於原本的一個bit的less與equal都要有兩個bit判斷大於與小於以及等於與不等於，因為若是只有一個bit(less與equal)，將0,0輸入alu\_top[1:31]的less與eqaul時，compare在判斷大於的時候，會因為0, 0而輸出1(~less & ~equal => greater)，但後來我們覺得應該可以再進行化簡，所以選擇了一個都沒有用到的1,0,0作為alu\_top[1:31]比較大小的control，若是1,0,0的話就輸出0，原本是這麼打算的，但後來覺得這樣太不直覺了而且要拉很多線，所以更好的方法應該是把compare的模組從alu\_top裡面移除，直接加在alu的裡面(也就是拉在alu\_top的外面)，這樣就可以把compare的結果直接拉在alu\_top[0]的set，其他的alu\_top的set都給0。

**Lesson learnt (if any):**

應用平常計組上課所學的知識，知道ALU要怎麼實作，不過平常上課學的是signed number，所以換到unsigned number的時候要想一下該怎麼做，然後bonus的compare看起來很簡單不過要加進去也試了很多不同的方法，最後才弄出一個比較合理也簡潔的結果。