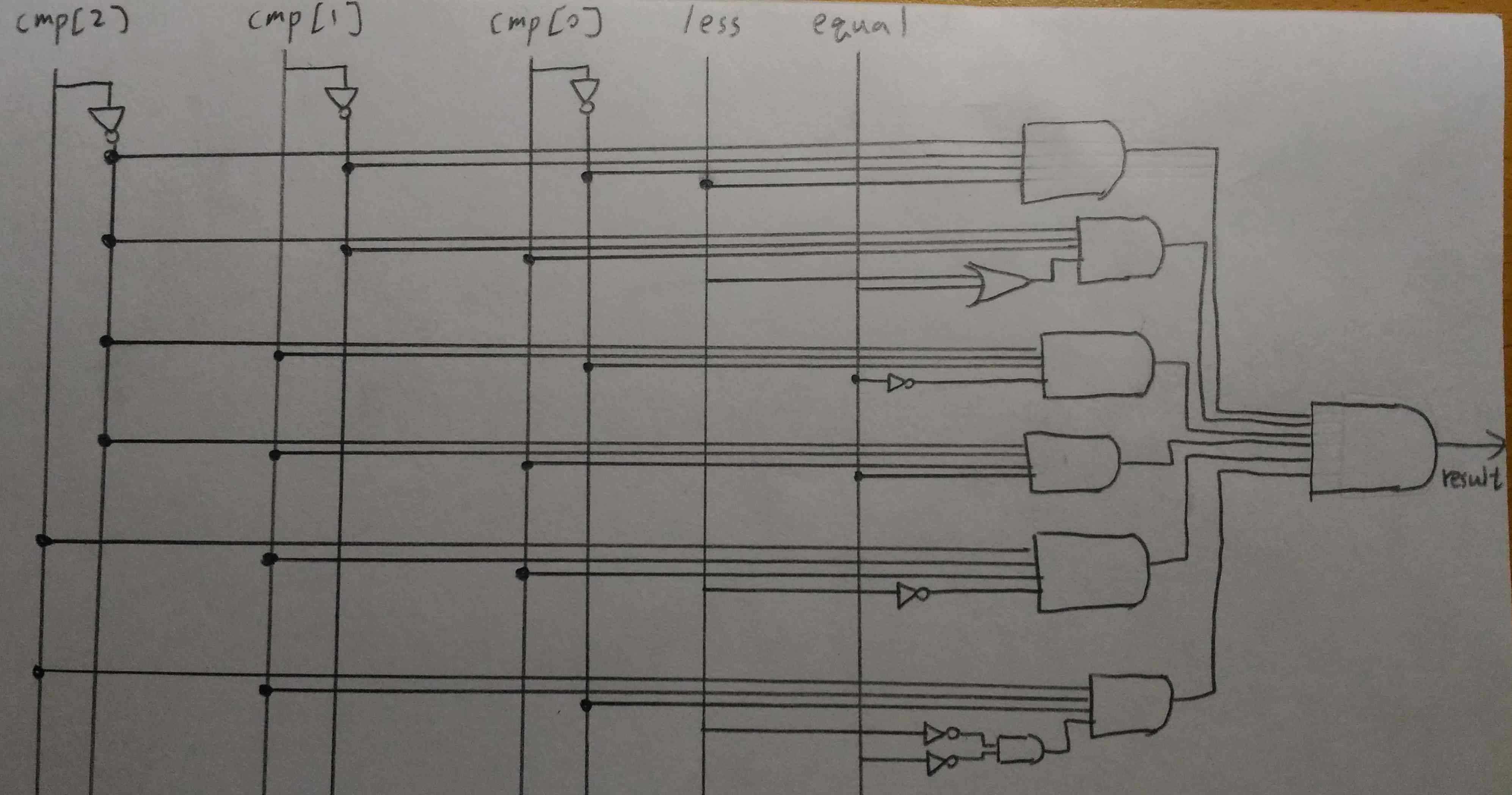
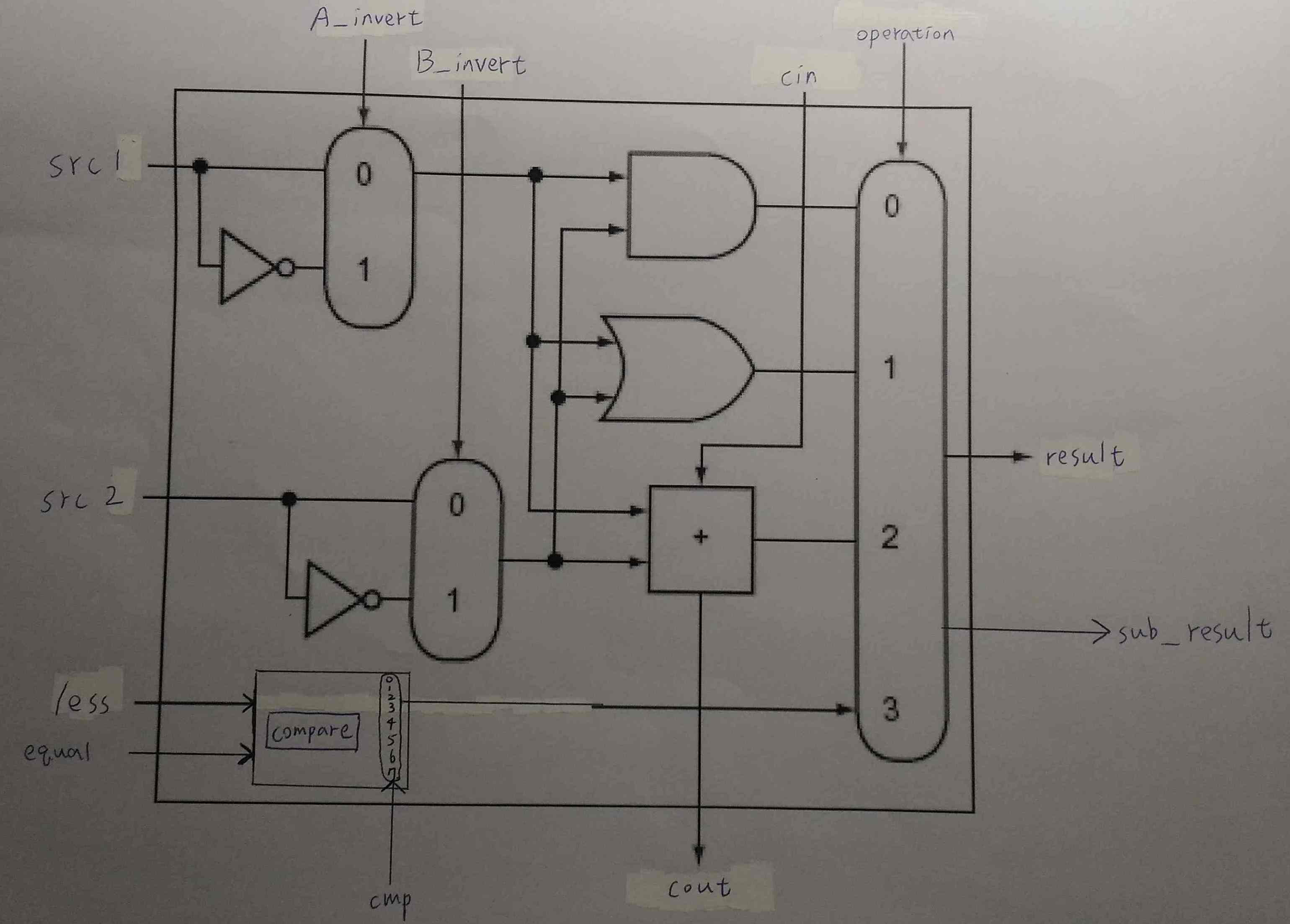
**Computer Organization**

**Architecture diagram:**

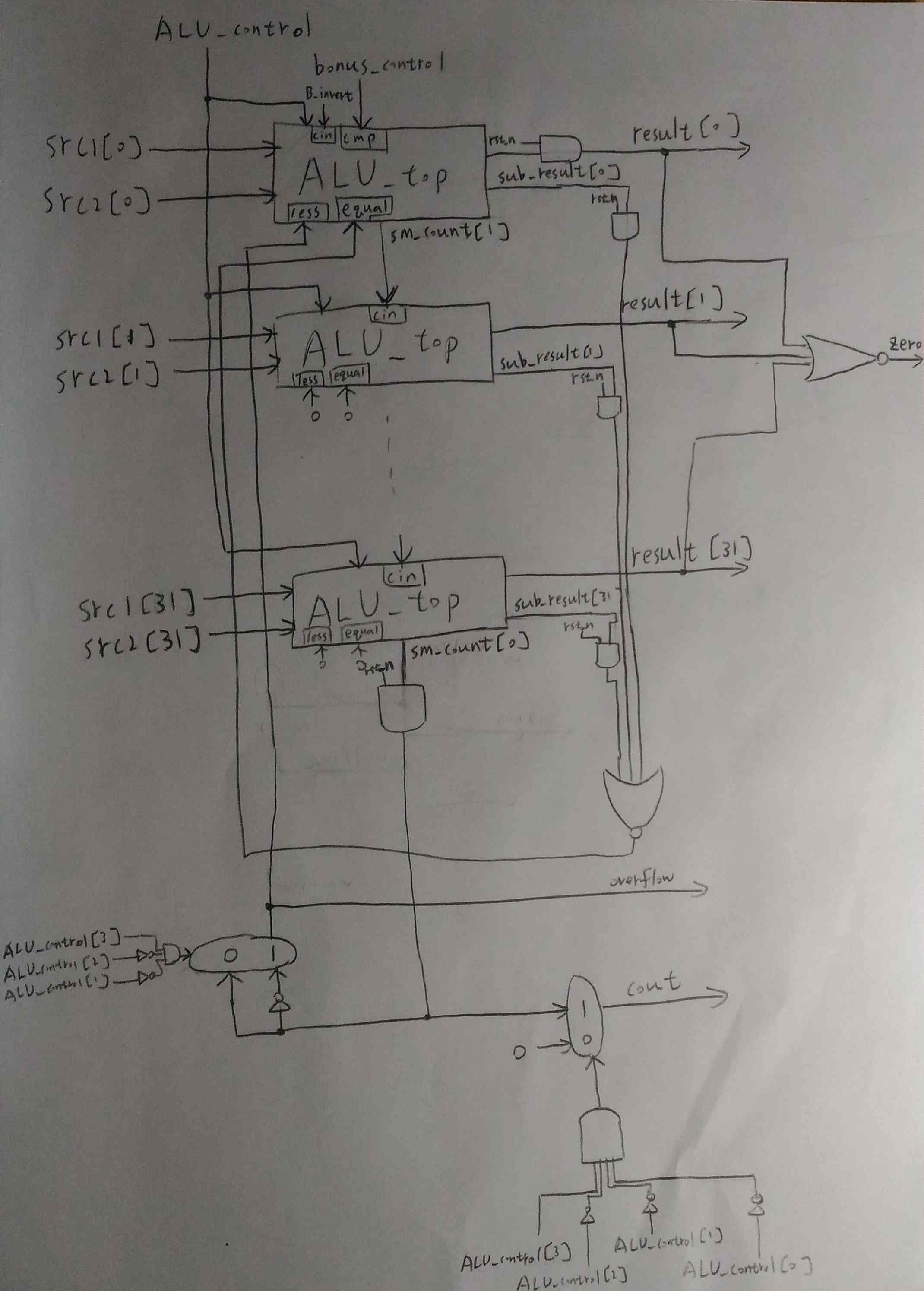
compare



ALU\_top



**ALU**

****

**Detailed description of the implementation:**

沒有bonus的部份就是依照課堂中教的東西實踐出來，比較特別的是電路圖中，因為多加了rst\_n的判斷，因此在電路圖中幾乎所有的輸出都and了rst\_n，讓只有在rst\_n=1時輸出才有機會不等於0

而bonus的部份，首先是先多加了compare這個module，可以根據不同的bonus\_code來利用less和equal決定output，而less就是接上原本的less，equal則是在ALU\_top裡多輸出了sub\_result，代表相減時的結果輸出，最後在ALU裡判斷這個sub\_result全部位元是否皆是0來作為equal的輸入

**Problems encountered and solutions:**

第一個遇到的問題當然就是verilog不熟悉，這個在看了網路資源後解決了

第二個是在判斷unsigned int overflow的部份，想了一段時間，才想到負數的overflow跟正數的overflow判斷方式剛好相反（一個是最後一個ALU\_top cout為1時overflow，另一個反而就是要cout=1才不是overflow）

第三個問題是在判定兩數是不是equal的情況，一開始是直接將zero接到equal那邊，但這樣的結果會不停止的振盪。因為如果一開始的result=0，zero就會是1，接回來equal之後，因為符合了equal的情況，result就會變成1，而zero就變成0了，這樣result又會不符合，因此一直來回振盪。最後因為想不到其他方法，只好再接一個sub\_result代表相減的結果，直接利用此來判斷是否相等當作equal輸入才解決了問題

**Lesson learnt (if any):**

一、對於ALU與其內部的結構實際了解了一遍

二、overflow的判斷方法

三、學會寫verilog