計算機組織LAB02心得 9817059 李揚

這次作業我寫成這樣的架構:

1. Decoder把instruction讀進來，先把前面的OPCODE轉換成ALU讀取的形式。然後把A、B存入Rs和Rt的訊息，A都是register值，但B有時是shift amount，有時是shift amount，有時是register值
2. 有了這些值之後，ALU便可根據這些訊號做運算，並把算出來的結果回傳回去給Decoder
3. Decoder得到ALU傳回來的值之後，找到相對應的register把ALUout傳回去。

我在這次作業遇到以下幾個難關

1. arithmetic shift right這個指令令我想了一段的時間，硬體實作方面，這個指令是利用把MSB同時拉到第一個(MSB)和第二個bit(第二大)，但如果shift 1~shift 31都拉線那就真的會寫到天昏地暗。雖然也可以用c++產生CODE，但畢竟也要寫一段時間。於是我上我查了一下發現也>>>這個operator可以解決這個問題，這個operator可以達到arithmetic shift right的效果(但一定要是裡面要shift的東西是signed的情況之下)。我後來仔細想想這其實應該算是必要的指令，否則用單純的shift right指令去兜arithmetic shift right是需要一段時間的。
2. 在考慮所有可以shift的時候有在想如果shift一個很大的值時要怎麼處理，結果思考一下發現，每shift32個bits之後就會回到原點，所以就利用這項特質，去把指令裡的shift amount除以32取餘數，也就是說把shift amount的值裡取[31:0]，就能解決這個問題了。
3. 因為之前沒什麼在寫TESTBED，所以對裡面的語法沒有這麼的熟悉，但這次又需要自己產生一些測資來做測試，所以上網查了一下TESTBED的各個語法，不過大致上都跟C蠻像的。
4. 其實…造成了我一些小困擾，ALU”C”tl , ALU”o”ut，這大小寫之差也真的令我打錯蠻多次的@@

心得:

這次時做了課本上寫的ALU(雖然沒有set less than，但多了shift right and shift left)，對課本的內容有了更加實際的概念。然後這次作業感觸最深的就是shifter這個硬體了!!因為我打CODE的時候就在想，如果電路裡要shift的一個值得時候，是要怎麼實做呢? 我大概想到了實作方法:.用一個一次可以shift left 一個bit 和一次可以shift right一個bit的硬體去實做他，然後傳給他shift amount值，然後一個一個去做shift(其實這就是數位電路裡教的shifter)，但我想到了一個小問題，這樣不是很慢嗎??後來我去查了一下一般電路裡有做shift的部分，如果是常常在做又是固定的shift的邏輯其實大部分都是用線接死的，如此一來才可以提升速度。仔細想想，這可以算是一種以”空間換時間”的方法之一，原本兩個shifter就可以做的事，現在為了提升速度，又再去做了一些把線接死的shifter。我也蠻好奇像我這樣寫的CODE會是其中的哪一個，浪費空間的32個shifter呢，還是浪費時間一個general shifter。我個人猜測是前者，因為這樣對synthesis的模擬器來說應該比較容易辦到。且這些指令又是在一個cycle內完成的!!後來去問了一下同學，也真的是如此XD。看來每打一行都是好幾個gates呀!

然後，我猜~~照這個進度來看，下一次的作業是小型的Processor。