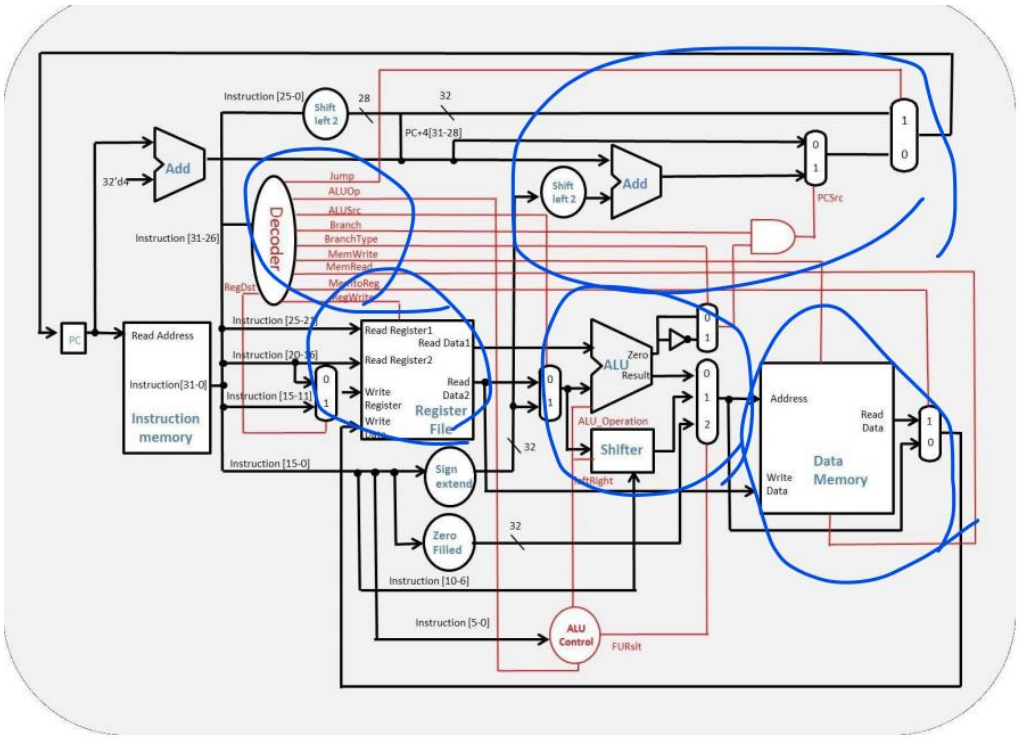
LAB3

1. Requirement

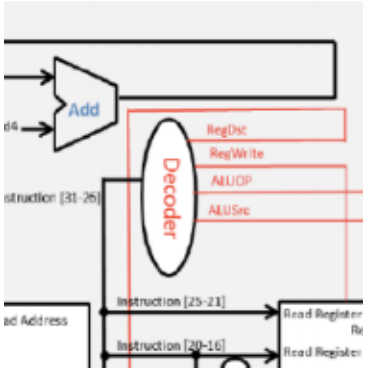
0613304 工工系 10 高宗霖

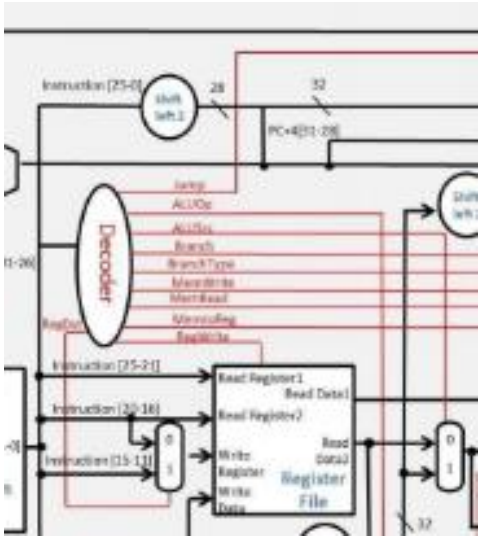
Set A. 完成基本指令



根據題目要求，我們需要修改的地方為以上畫圈的部分。雖然看似只是多了 data_memory 的部分，但 Decoder、許多 multiplexer、jump 就多了許多要該改的部分。

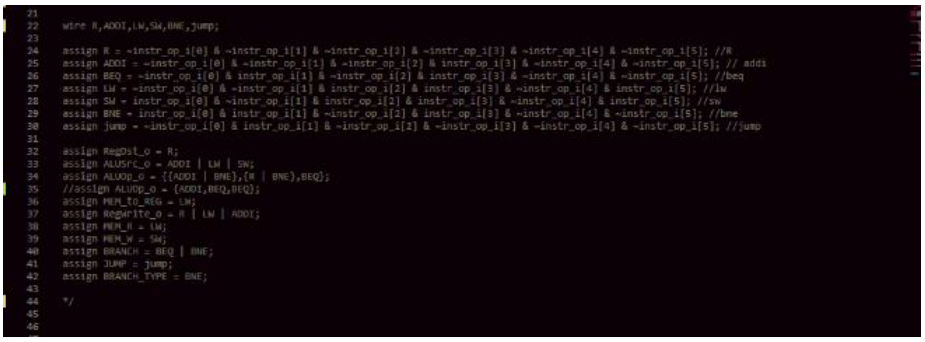
首先看 Decoder

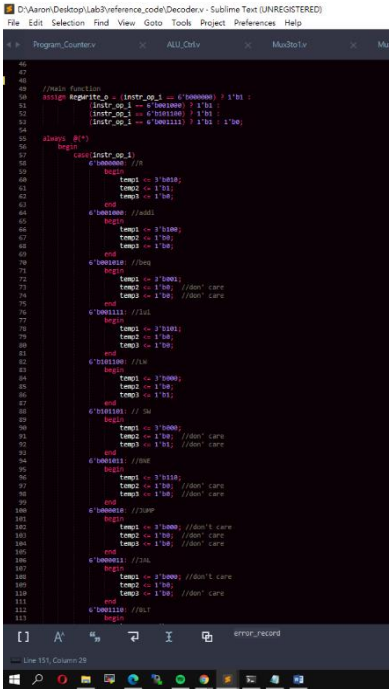




原本 Decoder 的 output 只有四根，現在有 10 根。因此我們要修改 Simple_Single_CPU.V 與 Decoder.V 內的參數設定。

其中 Decoder.V 中，我們要根據不同的 Opcode 來做相對應 control Line 的設定。





我原本是用 case by case 硬刻出來所有的 control bit(如右圖)
但後來查講義的時後意外發現其實可以用 gateway 的方式比較輕鬆好看。所以我也試著把它打出來了(結果也正確開勳~)
但為了保險起見，我還是以 case by case 為主，因此將 gate way 的方法註解掉。

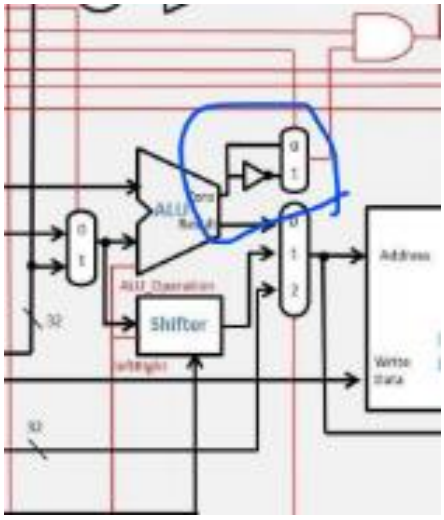
而以上實作為查表，並作對應，在此不多做說明

ALU_Control 的部分

比照原本的範例，現在題目要求只多了 lw、sw、branch 等要求。因此，將其對應的 ALUOP 寫上就完成了。

其中，Addi 的 ALUop 有改，要改為 1’b100 才有符合題目的設定。

接下來，因為我們有 Bne 的指令。因此要多一個 multiplexer 接在 ALU Zero 後面。



此 Multiplexer 的控制線為 Branch_Type

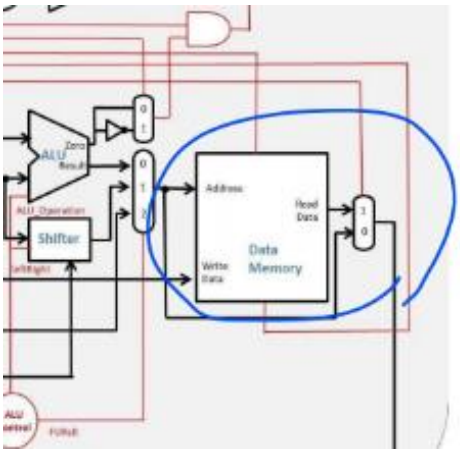
當指令為 Bne 時 Branch_Type 設為 1，其餘指令皆設為 0。

因此，當 ALU 比對結果為不一樣，Zero 會跑 0，接 NOT 後變為 1 接在 multiplexer 的 1 端。

接下來放入，data_Memory。該 module 不需要更改，只需要在 Simple_Single_CPU 內做對應的設定即可。

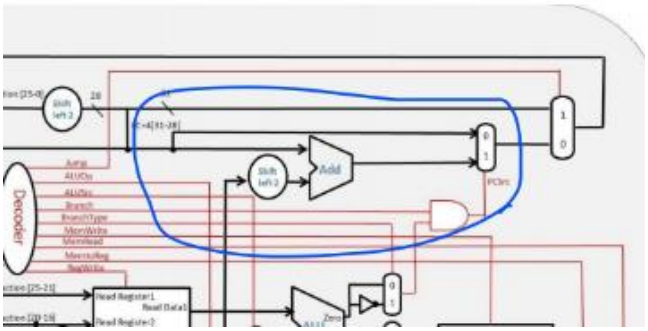
另外，我們要再加一個 multiplexer 控制輸出資料來源

最後，輸出的 wire 接回 register，基本上一個 cycle 就完成了。



接下來為 Branch

Branch 為 PC+4 再加上指令後面的 16bit 位址，因此我們要多加一個 Adder。



將 16bit sign extension 的結果與 PC+4 放入新增的加法器，output 即為 branch 的位址。

我們必須再新增一個 multiplexer 做為控制 Branch 或 PC+4 的路徑。而此 multiplexer 的控制線為下面 AND GATE 的結果。

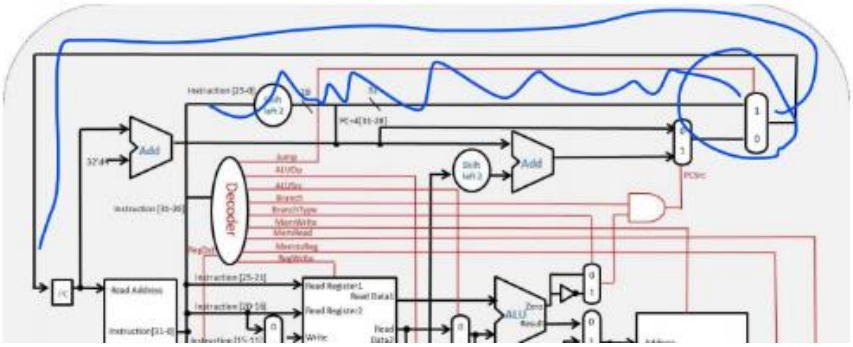
```
137
138     assign PCSrc = BRANCH & Branch_col;
139
140
141     Mux2to1 #(.size(32)) GO_Branch(
142             .data0_i(add_pc),
143             .data1_i(add_branch),
144             .select_i(PCSrc),
145             .data_o(select_branch)
146             );
147
```

而該 multiplexer 的控制線輸出，我用 assign 的方式直接將 Decoder 輸出的 Branch 與先前 multiplexer 的輸出做 AND。

最後一步，將 Jump 接上

Jump 指令為 PC+4 前 4bit 接上 instruction26bit left shift 2 bit

用一個 multiplexer 將 Jump 與剛剛 Branch multiplexer 輸出再做一次路徑控制，control selection 用 Decoder 輸出的 JUMP。將最後的輸出接回 PC，即大功告成。



Set B 增加 JAL、JR 指令

因為 JAL 需要將 PC+4 的 addr 再次存入 Register31 內。因此在線路設計需要稍微更改 Set A 的內容。

JAL Jump 的部分一樣不須多做設定，但我們需要拉一個線路，將 PC+4 拉到 Mem_to_Reg 這個 multiplexer 上面，使原本的分類器變為 3 to 1 multiplexer。然後將相對應的 control line 設定好成寫入模式(尤其是 RegisterWrite 一定要記得設成 1)。

JR 比較麻煩一點點，因為它要讀 Reg31 的位址並 Jump。

順應的 machine code ，rs_i、rt_i 輸出 rs_o、rt_o。因為 rt_o 為 0(zero)，因此經過 ALU 相加後即為要跳的地址。接上 multiplexer (3 to 1)即可完成。

這裡要注意的是，RegWrite 要特別注意，不能把它當 don’t care，不然它會寫東西進去 Reg 內(我一開始沒注意 Debug 了好久才抓到)

Set C 增加 BLT、BNEZ、BGEZ 指令

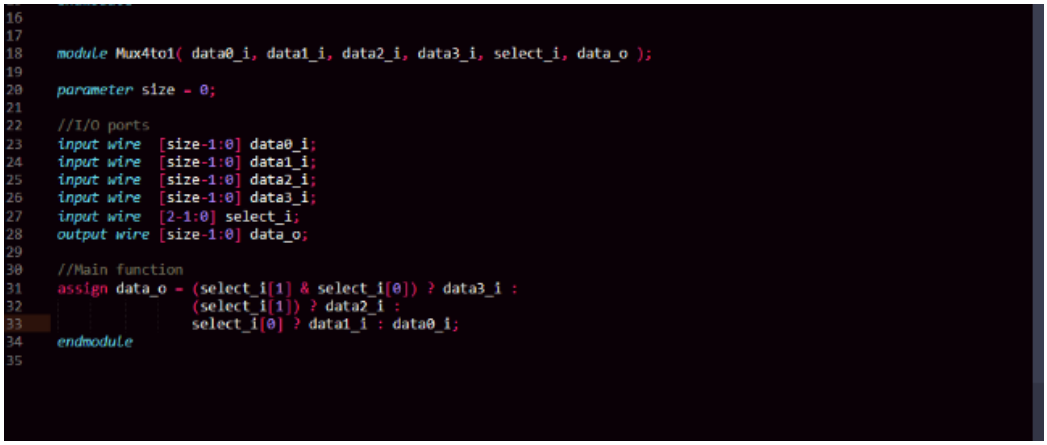
以上這三項指令 control line 與 Bne 差不多，需要更改的是在 ALU 內相對應的輸出。

BLT: 我在 ALU 內設了一個 sign bit，監看 result[31]是否為 1。若為 1 跳，反之亦然。

BNEZ 與 BNE 指令一樣，因此不需更改什麼。

BGEZ: 作法與 BLT 差不多，在 ALU 內設一個 sign bit 看 ALUsource1[31]是否為 1，為 1 不跳，反之亦然。

最後在 multiplexer 的部分我新增了一個 4 to 1 的 multiplexer，分別控制(Beq、Bne、Blt、Bgez)



```
16
17
18  module Mux4to1( data0_i, data1_i, data2_i, data3_i, select_i, data_o );
19
20  parameter size = 0;
21
22  //I/O ports
23  input wire  [size-1:0] data0_i;
24  input wire  [size-1:0] data1_i;
25  input wire  [size-1:0] data2_i;
26  input wire  [size-1:0] data3_i;
27  input wire  [2-1:0] select_i;
28  output wire [size-1:0] data_o;
29
30  //Main function
31  assign data_o = (select_i[1] & select_i[0]) ? data3_i :
32                  (select_i[1]) ? data2_i :
33                  select_i[0] ? data1_i : data0_i;
34  endmodule
35
```

我把此 module 放在 Mux3to1.V 中與 3to1MUX 並行。

結束!!!

最後心得:

這次作業看起來很難，我一開始也看檔案看到眼花。但仔細把每一份 code 掃過一遍竟然發現該做的都被寫完了。我只需要甜甜的把 control line 設定就好(感謝助教 carry)

Set B，Set C 的部分有些筱困難，在想怎麼接的時候都因為要大幅度更改 Set A 的內容搞得我有點不想做(但最後還是硬著頭皮趕在 Deadline 前兩小時做完)，呼~好險有趕完。