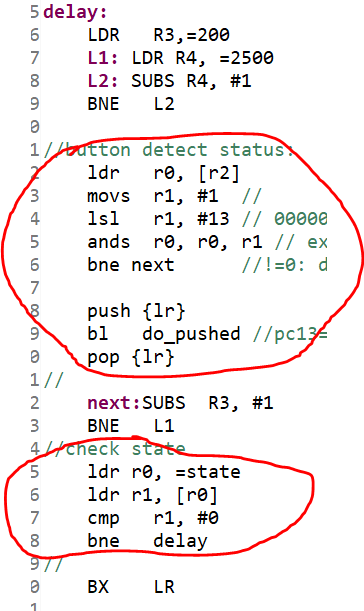
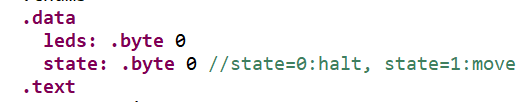
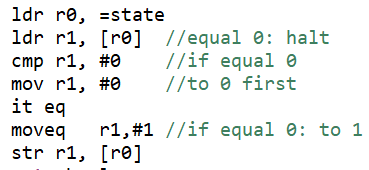
3.2. Lab3.2 : Push button (30%)

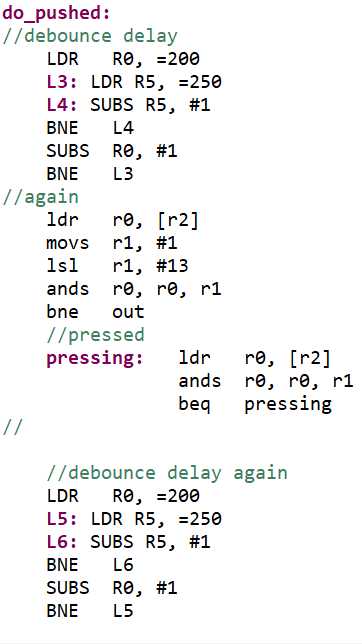
初始化GPIO PC13為pull-up input，並設計一程式輪詢 (polling) 實驗板上的 User button狀態，藉由按下按鈕去控制跑馬燈 (Lab3.1) 的停止與啟動。

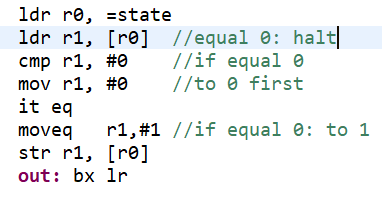
* 按按鈕**在delay迴圈內偵測**，不然放在計算led亮的loop的話，如果在按的時候剛好在delay loop，就讀不到按鈕。
* 在 memory放一個變數叫state (如圖):



* + State=0: 暫停
  + State=1: 繼續
  + 改變state方式：放在do\_pushed (如圖):



* + 因為state並不常改變，所以用記憶體的方式儲存，避免reg被占用。不然還需要一個reg去tracing 現在要做什麼，到後面reg可能不夠用
* 我把按按鈕偵測放在兩個迴圈之間，放最內圈應該也可以。
* 在delay做完後檢查state:
  + State=0: 暫停: 繼續跑delay迴圈
  + State=1: 繼續: 跳回原本led Loop迴圈，繼續計算led
* 按鈕有分3個階段:

1. Debouncing:
   * 在delay迴圈內第一次讀到pc13之後，經過一個小延遲(約0.01s)再讀一次，如果pc13還是一樣= 0代表真的有按，如果pc13 = 1就跳回去(out:bx lr)
   * 真的有按的話再小延遲一次，因為原本的按按鈕偵測放在delay，如果不延遲怕放開按紐後的bouncing會誤觸下次按鈕偵測
2. Pressing:
   * 讓按住的情況一值卡在這個小迴圈
   * 如果不鎖住，那按住按鈕會不停誤觸下次按鈕偵測
3. State changing: 按成功之後才觸發
   * 原本是0 =>變1
   * 原本是1 =>變0
   * 變更記得用str放回記憶體