112-1 組合語言與嵌入式系統 Midterm Project 報告

112 學年度第 1 學期

老師：朱守禮　老師
學生：
電資三 11020107 蘇伯勳
電資三 11020140 葉柏榆
電資二 11120115 林佩玟

一、背景

此次 Midterm Project 之目的為使用 ARM Assembly 撰寫能夠列印組別、組員姓名、組員學號與其加總的程式。由於為了符合 Project 的基本要求，因此會有些程式碼是單純為了滿足要求而寫的，如果刪掉對程式的目的與功能不會有影響。分工：蘇伯勳負責寫報告，葉柏榆負責 debug，林佩玟負責撰寫程式。

二、方法

按照作業說明中的要求，總共有三支程式：”name.s”、”id.s”、”main.s”。

- 對三支程式都有效的基本要求：
  1. 需要使用 Load/Store 指令中 10 種定址模式中的 3 種以上。
  2. 需要使用 Data Processing 指令中 13 種 Operand2 格式中的 4 種以上不同格式。
  3. 程式中需要包含 4 道以上不同的非 Branch 指令的 Conditional Execution，且不包含 al 條件。

以下分別針對三支程式說明其功能、要求與設計方法：

1. “name.s”（NAME 函數）：

   甲、函數功能：印出組別與組員名單。

   乙、要求：

   i. 組別與組員必須分行印出。

   ii. 若組員不足三位，重複填入已有的組員姓名，直到用完三個記憶體區塊為止。

   丙、設計方法：

   由 main.s 呼叫，先將組別與三位組員的資訊分別寫死在.data 段，其它諸如”*****Print Name*****”之類的提示訊息也寫在死區

塊。在.text 段為了讓 name.s 能被 linker 看見而被 main.s 呼叫，將 name 函數使用.global 定義在此處。

接下來便是 name 函數的真正程式碼。首先，stmfd sp!, {lr}先儲存 return address，接著使用 ldr r0, = to_print_msg 與 bl printf 互相搭配執行此函數的功能：印出組別與三位組員的英文名字（因為中文不適用）。在印出 end_msg 之後，接著有三個註解 basic requirements xx 的部分是單純為了滿足要求而且的程式碼：

@ basic requirements 02

為了滿足「需使用 Load/Store 指令中，10 種定址模式當中的 3 種以上不同模式。」要求。

```
ldr r3, [r2,#4]
ldr r5, [r2,r4]
```

@ basic requirements 03

為了滿足「需使用 Data Processing 指令中，13 種 Operand2 格式的當中 4 種以上不同格式。」要求。

```
sub r0, r1, #3
lsl r0, r1, #3
```

@ basic requirements 04

為了滿足「程式中須包含 4 道以上不同的非 Branch 指令的 Conditional Execution（不包含 al 條件）。」要求。

```
cmp r0, r1
addgt r2, r0, r1
cmp r0, r1
addne r0, r1, r2
cmp r0, r1
sbcgt r0, r1, r2
```

最後，mov r0, #0 將 r0 暫存器歸零，ldmfd sp!, {lr}將 lr 還原以 return 到 call function 的 address，mov pc, lr 是因為 name.s 被 main.s 呼叫，所以必須 return 到呼叫 name.s 的 address。在最後使用 swi 0 更新 cpu 狀態。

2. “id.s”（ID 函數）：

甲、函數功能：輸入組員的學號，並印出組員學號與學號加總。

乙、要求：

i. 規劃 4 個記憶體位址，作為之後輸入組員學號與總和的緩衝區。

ii. 請以輸入的方式，輸入三個組員的學號，並記錄於先前規劃的記憶體位址。

iii. 學號輸入完後，將 3 組輸入學號加總成一個值，紀錄餘地 4 個記憶體區塊。

iv. 輸入完 3 組學號後，按下 p 鍵，即分行印出完整的組員學號與組員總和。

v. 在此 id.s 程式中的第 6 道實體指令必為必須執行之指令：adds lr, pc, r0。

丙、設計方法：

由 main.s 呼叫，先在.data 段寫入各種 to_print_msg 與其中的必要小零件（如"\n"），再存五個.word：三位組員的學號 id1, id2, id3，總和 sum，與讀入指令 command，並初始化為 0。.text 段一樣為了要透過 linker 被 main.s 存取，將此函數 id 與上述的記憶體空間使用.global 定義在此。

接下來 id 函數的實體指令開始：

```
@ 備份 address
stmfd sp!, {lr} @ 第一道
@ 印出提示訊息
ldr r0, =start_msg @ 第二道
bl printf @ 第三道
@ 由於要求第六道實體指令會動到 pc 與 lr：
mov r4, lr @ 第四道，先將 lr 備份到 r4
mov r0, #0 @ 第五道，塞個無用的指令(類似 nop 的用途)
adds lr, pc, r0 @ 第六道，lr = pc + r0, set CPSR，要求的指令
mov lr, r4 @ 第七道，還原 lr 的值
@ 接下來使用 scanf 讀取 id1 ~ id3
ldr r0, =enter_msg1 @ get id1
bl printf
ldr r0, =InputInt
ldr r1, =id1
bl scanf
ldr r0, =enter_msg2 @ get id2
bl printf
ldr r0, =InputInt
ldr r1, =id2
bl scanf
ldr r0, =enter_msg3 @ get id3
bl printf
ldr r0, =InputInt
ldr r1, =id3
```

```
bl scanf
@ 接下來要持續讀指令直到'p'
@ 印出讀取指令的提示訊息
ldr r0, =enter_msg_command
bl printf
@ 直到讀入的 Command=='p'以前都 looping(使用.loop)
    ldr r0, =InputStr @ load InputStr 的 address 到 r0
    ldr r1, =command @ load command 的 address 到 r1
    bl scanf @ 讀入 string 到 r1（指的位置）
    ldr r1, =command @ 重抓 r1 的 address
    ldrb r1, [r1] @ 將讀入的 string 由間接定址變成直接定址
    cmp r1, #'p' @ 將讀入的 string 與'p'比較
    bne loop @ 不同就跳回 loop 標籤
@ 印出列印的提示訊息
ldr r0, =print_command
bl printf
@ 以下列印 id1~id3, OutputInt == "%d\n"
ldr r0, =OutputInt @ id1
ldr r1, =id1
ldr r1, [r1]
bl printf
ldr r0, =OutputInt @ id2
ldr r1, =id2
ldr r1, [r1]
bl printf
ldr r0, =OutputInt @ id3
ldr r1, =id3
ldr r1, [r1]
bl printf
ldr r0, =newline @ 換行
bl printf
@ 將三個 id 加總
ldr r3, =id1
ldr r3, [r3] @ 將 id1 存到 r3
ldr r4, =id2
ldr r4, [r4] @ 將 id2 存到 r4
ldr r5, =id3
ldr r5, [r5] @ 將 id3 存到 r5
```

adds r10, r3, r4 @ r10 = id1 + id2, set CPSR
adds r10, r10, r5 @ r10 = (id1 + id2) + id3, set CPSR
ldr r11, =sum @ 將 sum 的記憶體位址 load 到 r11
str r10, [r11] @ 將加總的值由 r10 store 到 r11 指向的位址
ldr r0, =result_msg @ 印出加總的提示訊息
ldr r1, =sum @將加總 load 到 r1
ldr r1, [r1] @ 間接定址 to 直接定址
bl printf @ printf(r0, r1)
@ 印出結束訊息
ldr r0, =end_msg
bl printf
@ reset r0, pc, lr, cpu
ldmfd sp!, {lr} @ 拿回 address
mov r0, #0 @ r0 歸零
mov pc, lr @ return 到 main.s 呼叫 id.s 的 address
swi 0 @ 更新 cpu

3. “main.s”（MAIN 函數）：

甲、函數功能：呼叫、整合前面兩個函數，並且印出組別、學號、姓名與總和等完整資料。

乙、要求：

i. 在 MAIN 中呼叫 NAME 與 ID 兩個函數，分別達成前兩之函數的分項功能，不可直接複製貼上。

ii. 應用 NAME 與 ID 函數所記錄的資料，輸出完整的組別、組員資訊、與組員學號數值計算結果。

丙、設計方法：

在.data 段存各種 to_print_msg 與小零件，在.text 段只有.global main，因為這是主程式，不需要被別人呼叫。

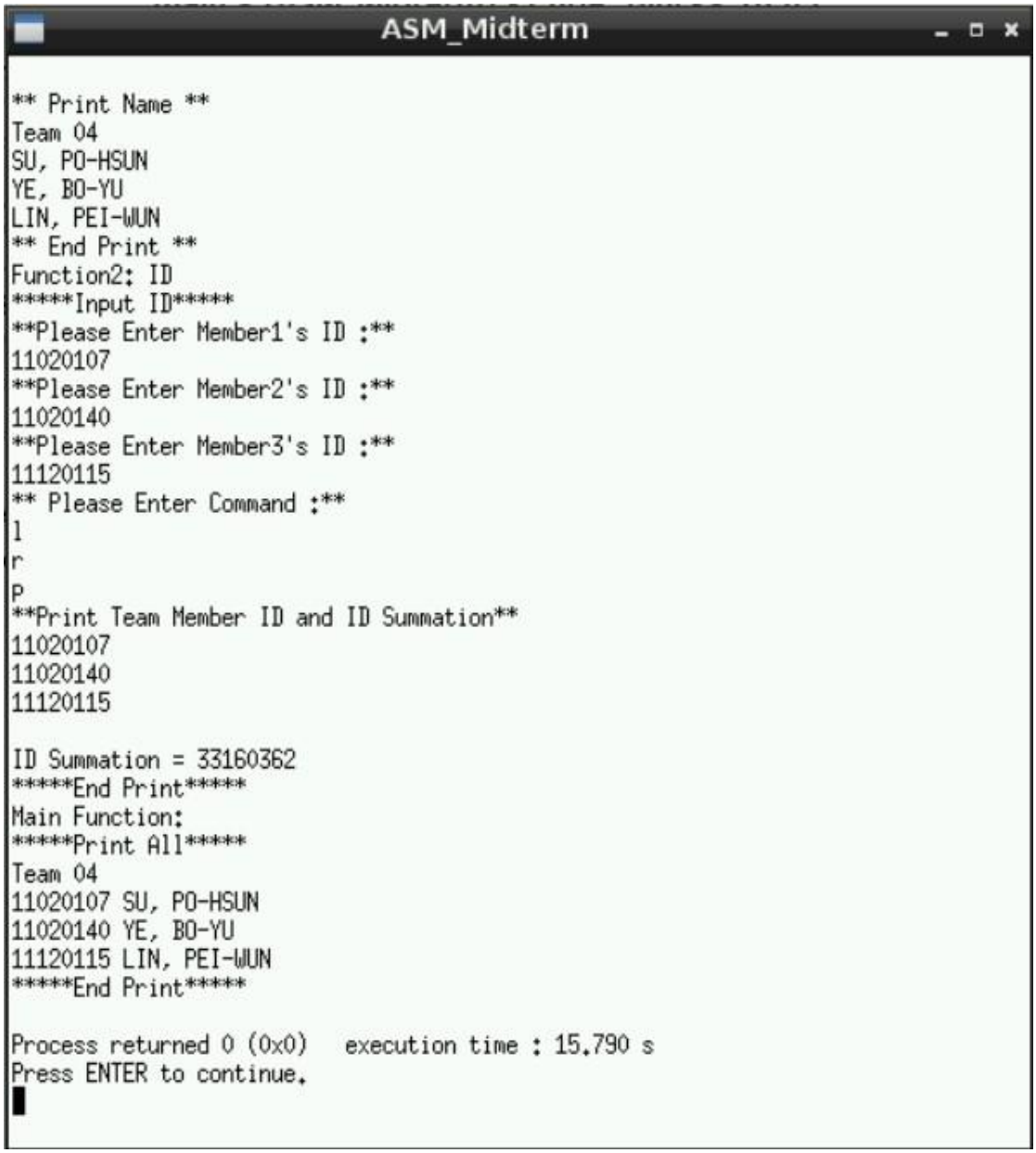
在主程式中：

stmfd sp!, {lr}
@ 呼叫執行 NAME、ID
ldr r0, =SubFunc1
bl printf
bl name @ call name.s
ldr r0, =SubFunc2
bl printf
bl id @ call id.s
@ 印出 main 的提示訊息
ldr r0, =MainFunc

```
bl printf
@ 印出組別，因為.global 所以使用 NAME 的 label
ldr r0, =team
bl printf
@ 印出 start_msg
ldr r0, =start_msg
bl printf
@ 印出 id1 ~ id3
ldr r0, =IDandNAME
ldr r1, =id1
ldr r1, [r1]
ldr r2, =member1
bl printf
ldr r0, =IDandNAME
ldr r1, =id2
ldr r1, [r1]
ldr r2, =member2
bl printf
ldr r0, =IDandNAME
ldr r1, =id3
ldr r1, [r1]
ldr r2, =member3
bl printf
@ 印出結束訊息
ldr r0, =end_msg
bl printf
@ 重設 pc, lr, cpu, r0
ldmfd sp!, {lr}
mov r0, #0
mov pc, lr
swi 0
```

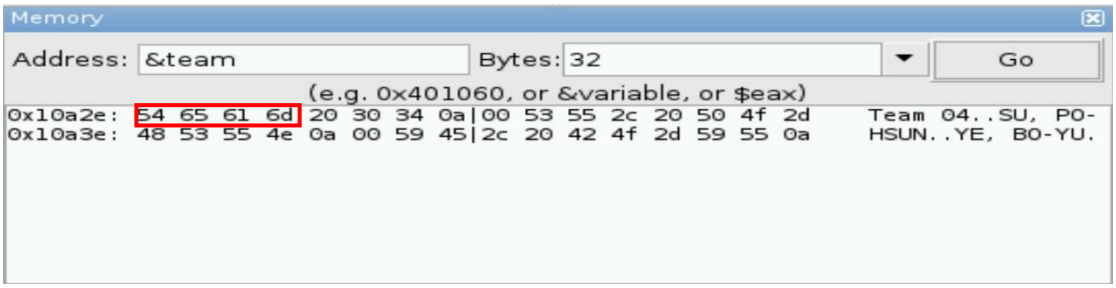
三、結果（截圖）

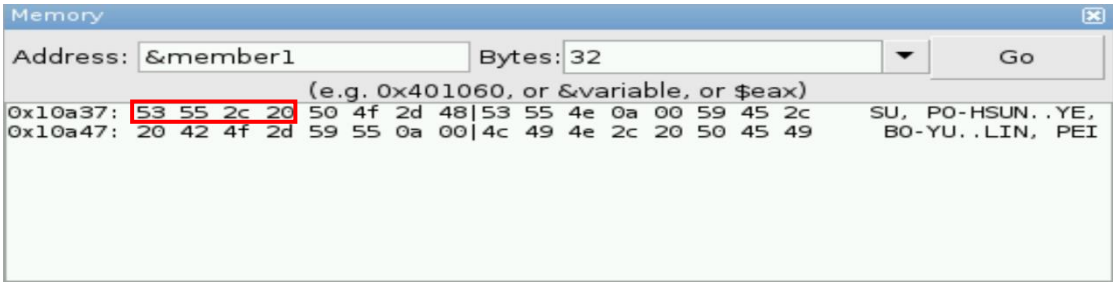
執行結果：

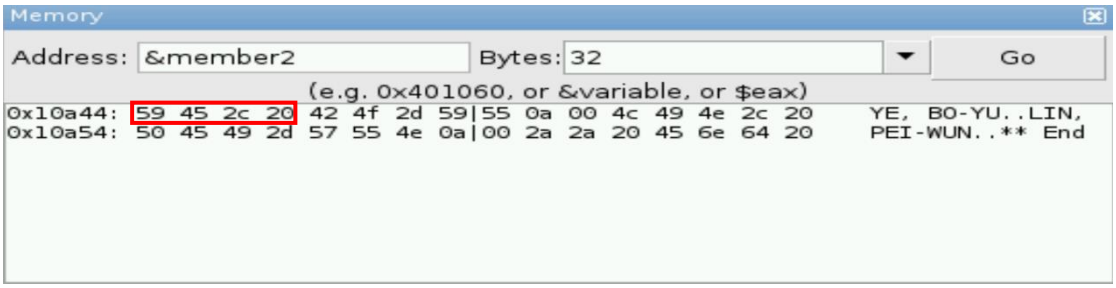


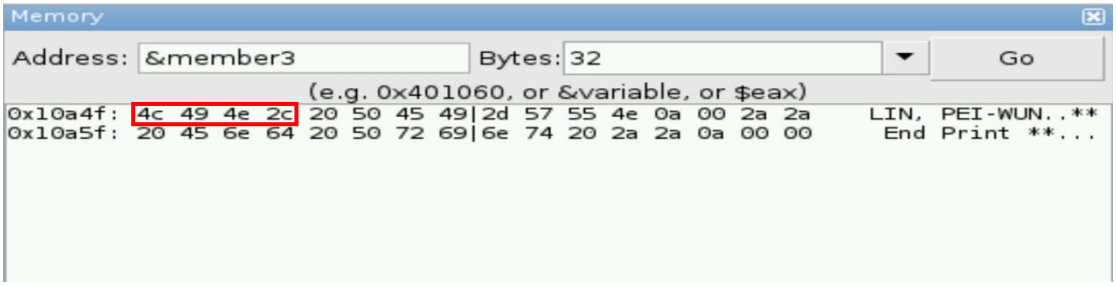
Memory dump：

- name.s

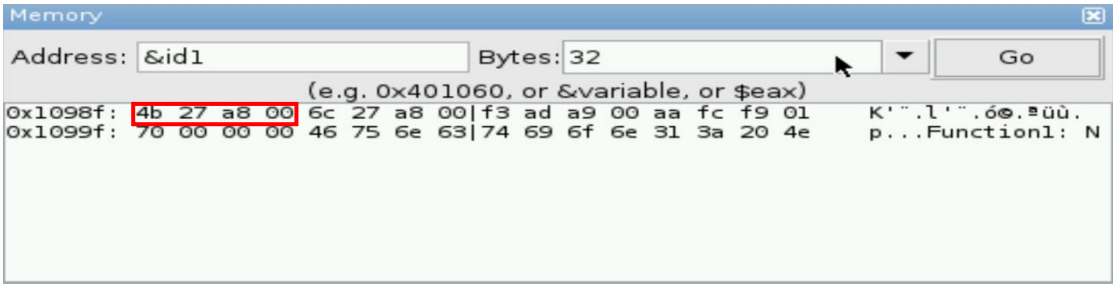


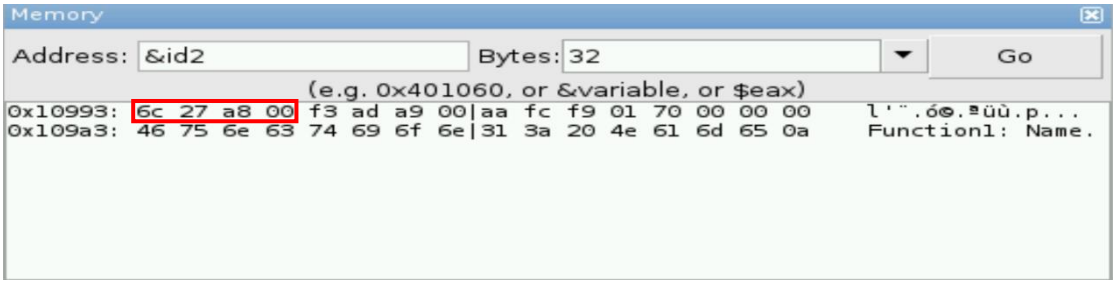

Memory
Address: &member1
Bytes: 32
Go
(e.g. 0x401060, or &variable, or $eax)
0x10a37: 53 55 2c 20 50 4f 2d 48|53 55 4e 0a 00 59 45 2c SU, PO-HSUN..YE,
0x10a47: 20 42 4f 2d 59 55 0a 00|4c 49 4e 2c 20 50 45 49 BO-YU..LIN, PEI

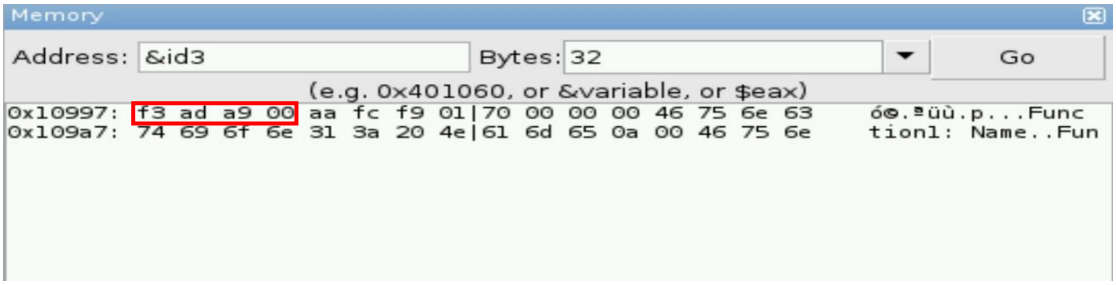

Memory
Address: &member2
Bytes: 32
Go
(e.g. 0x401060, or &variable, or $eax)
0x10a44: 59 45 2c 20 42 4f 2d 59|55 0a 00 4c 49 4e 2c 20 YE, BO-YU..LIN,
0x10a54: 50 45 49 2d 57 55 4e 0a|00 2a 2a 20 45 6e 64 20 PEI-WUN..** End

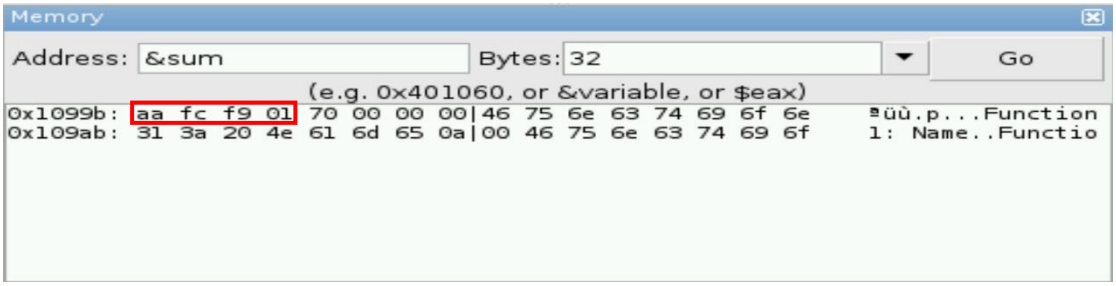

Memory
Address: &member3
Bytes: 32
Go
(e.g. 0x401060, or &variable, or $eax)
0x10a4f: 4c 49 4e 2c 20 50 45 49|2d 57 55 4e 0a 00 2a 2a LIN, PEI-WUN..**
0x10a5f: 20 45 6e 64 20 50 72 69|6e 74 20 2a 2a 0a 00 00 End Print **...

- id.s

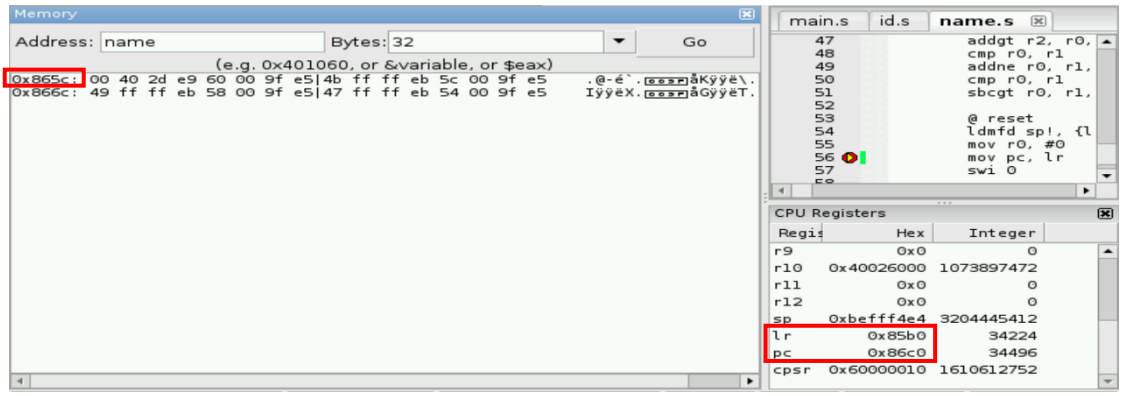

Memory
Address: &id1
Bytes: 32
Go
(e.g. 0x401060, or &variable, or $eax)
0x1098f: 4b 27 a8 00 6c 27 a8 00|f3 ad a9 00 aa fc f9 01 K'¨.l'¨.óꩪüù.
0x1099f: 70 00 00 00 46 75 6e 63|74 69 6f 6e 31 3a 20 4e p...Function1: N


Memory
Address: &id2
Bytes: 32
Go
(e.g. 0x401060, or &variable, or $eax)
0x10993: 6c 27 a8 00 f3 ad a9 00|aa fc f9 01 70 00 00 00 l'¨.óꩪüù.p...
0x109a3: 46 75 6e 63 74 69 6f 6e|31 3a 20 4e 61 6d 65 0a Function1: Name.

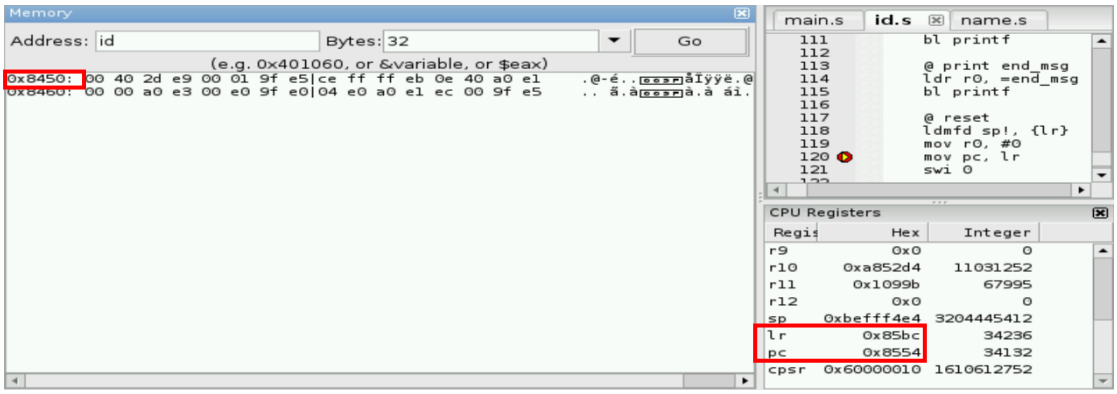

Memory
Address: &id3
Bytes: 32
Go
(e.g. 0x401060, or &variable, or $eax)
0x10997: f3 ad a9 00 aa fc f9 01|70 00 00 00 46 75 6e 63 óꩪüù.p...Func
0x109a7: 74 69 6f 6e 31 3a 20 4e|61 6d 65 0a 00 46 75 6e tion1: Name..Fun



- main.s



name.s 起始位址 0x865c 結束位址 0x86c0 返回位址 0x85b0



id.s 起始位址 0x8450 結束位址 0x8554 返回位址 0x85bc

四、討論

大部分人都會犯的錯我們也犯了，在建新 project 時沒把自帶的 hello world 的 cpp 檔案刪掉，導致 compile 時把原本的 main 蓋掉，要重寫一次，真的很扼腕。這次的題目很簡單，最大的問題反而是在處理前面這種小錯，與 Raspberry Pi 虛擬機設定等等要怎麼使用之類的。

比較麻煩的地方是為了要使用 printf 和 scanf，必須要記得寫 ldr r1, [r1]這種將間接定址轉為直接定址的先行步驟，不然印出時都是 address，常常忘記。並且，虛擬機很卡，希望之後能教怎麼在 VScode 中編譯執行.s 檔案。在截圖或查資料時，不只是要將游標來回切換，檔案傳輸也得使用 WinSCP，而使用 WinSCP 時虛擬機又必須保持登入，不只又慢又卡而且很麻煩。

五、結論

完成 project 要求的功能不難，比較多問題的是在虛擬機的使用與各種工具的使用上。下一次 project 應該可以寫得更好，不用為了要求而寫沒用的 code。

六、未來展望

希望下一次的內容不用為了完成要求而寫完全沒作用也沒必要的程式碼，必且希望下次的期末 Project 能在五天內解決；這個語言 debug 起來比較麻煩，雖然的確還蠻好玩的，但若講求效率還是高階語言寫起來快樂許多。