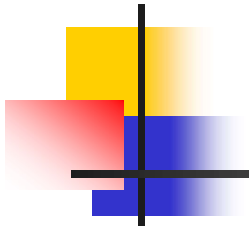
組合語言與微處理機
作業一說明

老師：張雲南



- 此份投影片包含:
  - 作業規定
  - 編譯執行方式
  - 簡易組語程式範例說明
  - template
  - 可能用到的函式
  - 參數傳遞方式

作業規定

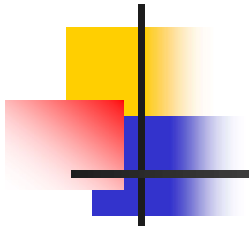
- I/O

```
D:\>arm-none-eabi-run a.out "hello world"
dlrow olleh
D:\>
```

- 請依照上圖方式接受參數
  - 參數將會依照APCS rule存到r0,r1,…etc
- 若是將輸入定義在程式裡會斟酌扣一點分數,EX:

```
Tex:
    .ascii  "Hello word\000"
```

  - 比如說現在的輸入資料不想要是Hello world, 想改成hi,那就必須要在程式碼裡面加入 “hi” 的字串,這樣是不完全的作法.



- 繳交方式：網大
  - Source code *.s
  - Executable file *.out
  - *檔名統一：reverse.s reverse.out
  - 做成壓縮檔: ID_HW#_Version#.(zip,rar…,etc)
    ex：b123456789_hw1_V1.rar
  - 評分將採用最新的版本
    - ex: b123456789_hw1_V1.rar與b123456789_hw1_V2.rar, 助教只會拿V2版本來評分

編譯執行

- 下圖包含了編譯的指令以及執行的指令
  - arm-none-eabi-gcc 檔名 -T generic-hosted.ld (編譯指令)
  - arm-none-eabi-run 執行檔名 (執行指令)

```
D:\>arm-none-eabi-gcc aaa.s -T generic-hosted.ld

D:\>arm-none-eabi-run a.out
Hello world
D:\>_
```

- 接著將以一個組合語言範例說明
  - 該程式相當於printf(“Hello world”);
  - 執行結果如上圖

組語程式範例說明

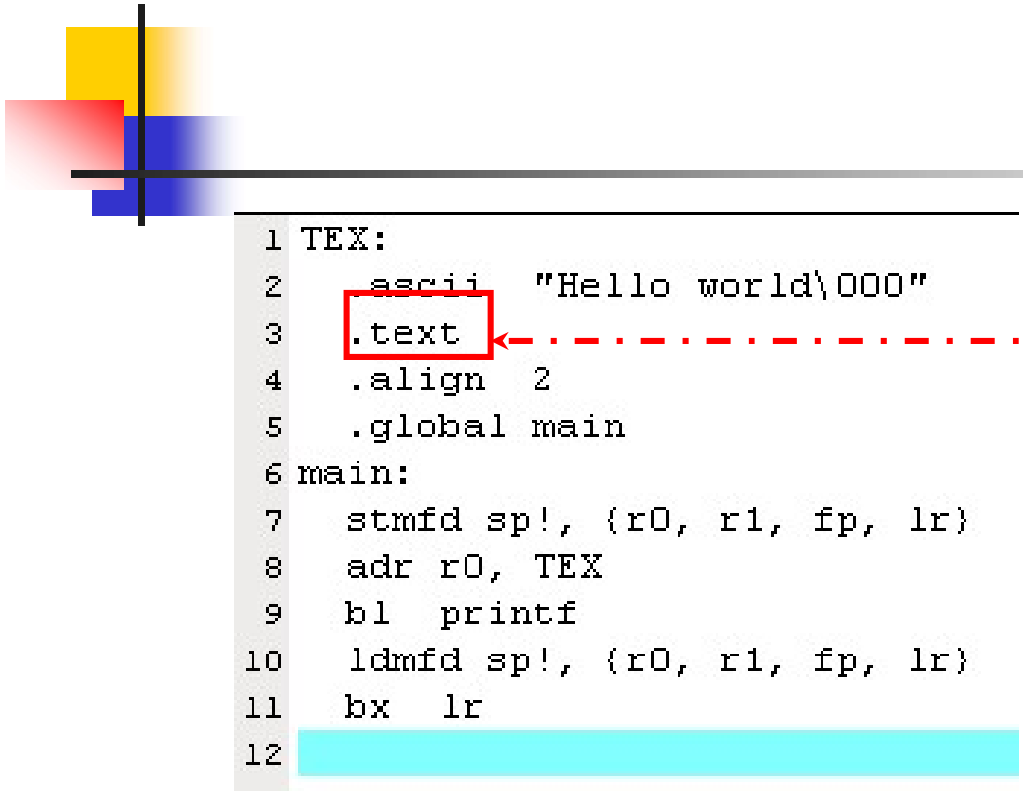
```
1 TEX:
2    .ascii  "Hello world\000"
3    .text
4    .align  2
5    .global main
6 main:
7    stmfd sp!, {r0, r1, fp, lr}
8    adr r0, TEX
9    bl  printf
10   ldmfd sp!, {r0, r1, fp, lr}
11   bx  lr
12
```

label—與課堂上不同的地方為，後面要加”：”

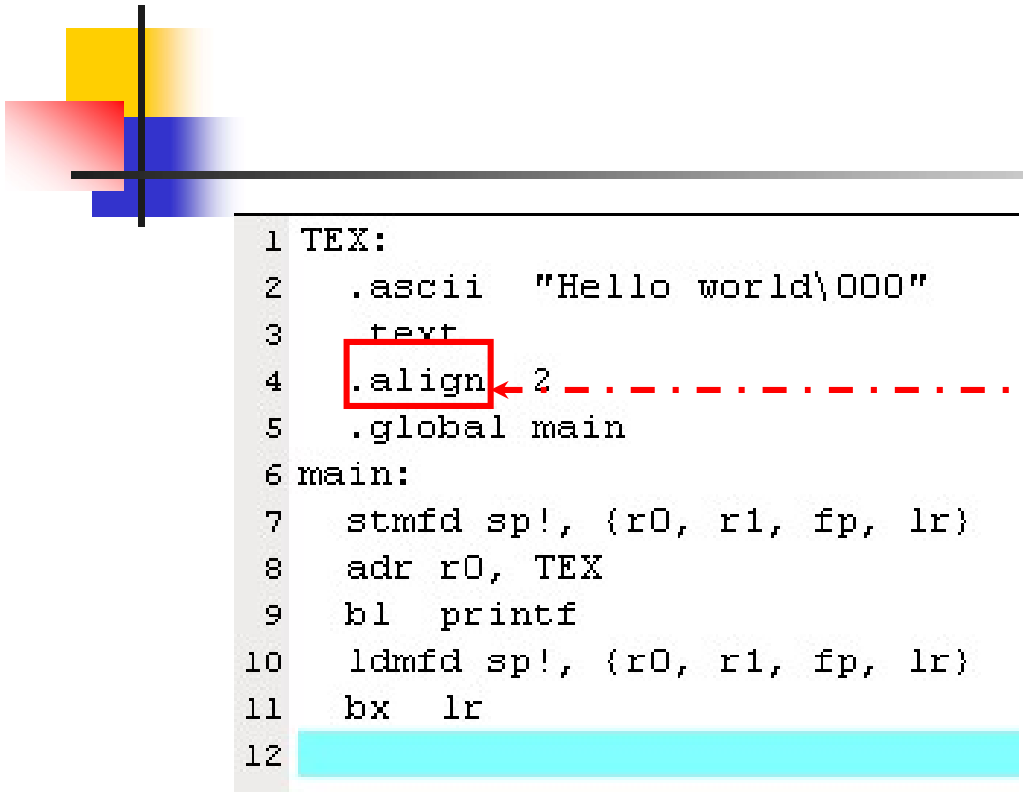
**部份說明來自組語實習講義

```
1  TEX:
2    .ascii  "Hello world\000"
3    .text
4    .align  2
5    .global main
6  main:
7    stmfd sp!, {r0, r1, fp, lr}
8    adr r0, TEX
9    bl  printf
10   ldmfd sp!, {r0, r1, fp, lr}
11   bx  lr
12
```

It assembles each string (with no automatic trailing zero byte) into consecutive address



表示以下開始為
程式碼主體
(optional)



Pad the location counter (in the current subsection) to a particular storage boundary

It is aligned power of 2

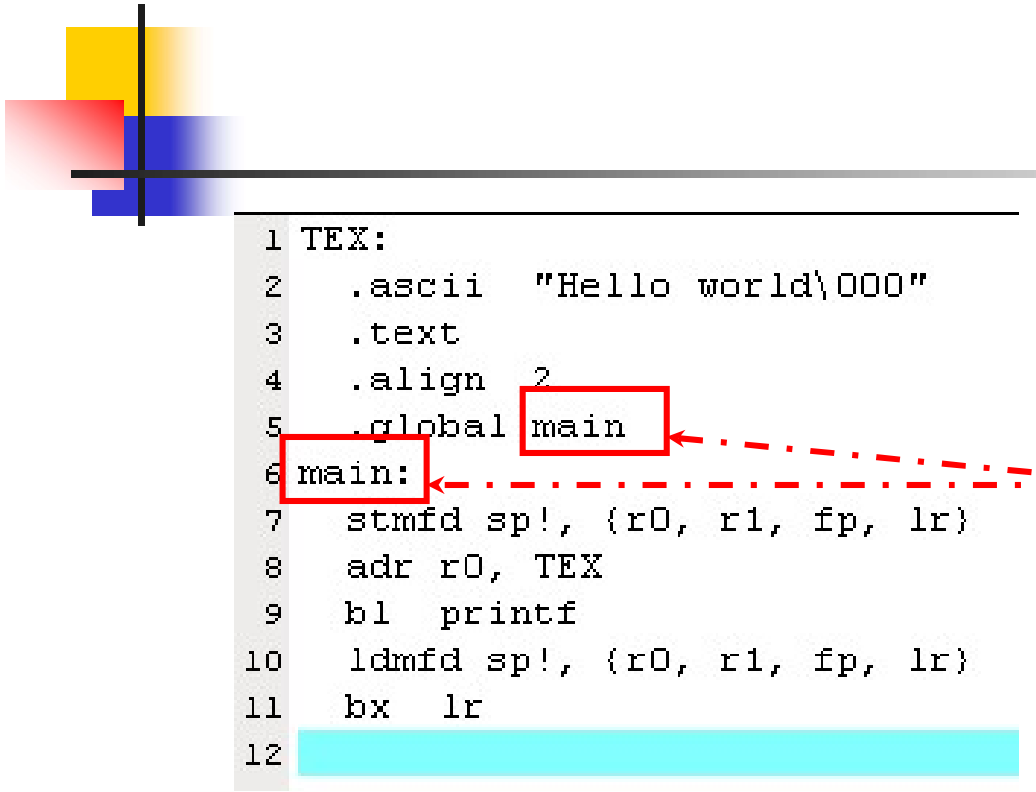
For example, aligned 4 byte:

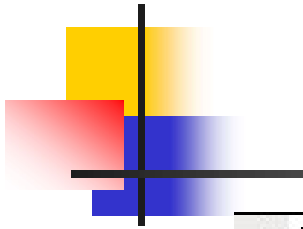
.align 2

```
1  TEX:
2    .ascii  "Hello world\000"
3    .text
4    .align  2
5    .global main
6  main:
7    stmfd sp!, {r0, r1, fp, lr}
8    adr r0, TEX
9    bl  printf
10   ldmfd sp!, {r0, r1, fp, lr}
11   bx  lr
12
```

Make the symbol visible to ld.

At least we must have a global symbol called “main” because we use the “generic-hosted.ld” for our linker script.


1 TEX:
2 .ascii "Hello world\000"
3 .text
4 .align 2
5 .global main
6 main:
7 stmfd sp!, {r0, r1, fp, lr}
8 adr r0, TEX
9 bl printf
10 ldmfd sp!, {r0, r1, fp, lr}
11 bx lr
12
一定要有”main”



```
1  TEX:
2    .ascii  "Hello world\000"
3    .text
4    .align  2
5    .global main
6  main:
7    stmfd sp!, {r0, r1, fp, lr}
8    adr r0, TEX
9    bl  printf
10   ldmfd sp!, {r0, r1, fp, lr}
11   bx  lr
12
```

呼叫已定義(內建)的函式

```
1  TEX:
2    .ascii  "Hello world\000"
3    .text
4    .align  2
5    .global main
6  main:
7    stmfd sp!, {r0, r1, fp, lr}
8    adr r0, TEX
9    bl  printf
10   ldmfd sp!, {r0, r1, fp, lr}
11   bx  lr
12
```

結束此部份程式,

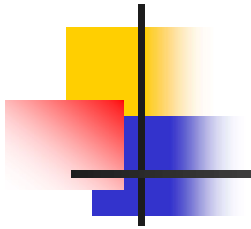
- 範例程式碼為正確可執行的,同學可以編譯執行看看

template

```
    .text
    .align  2
    .global main
main:
    stmfd sp!, {.....}
    @================================
    @code body
        :
        :
        :
    @end of code body
    @================================
    ldmfd sp!, {......}
    mov pc, lr @end
```

根據不同情況會需要存不同的 register

同學的程式碼主要就寫在這地方



函式

- 關於此次作業,有機會用到的函式
  - printf—類似C裡面的printf
    - 將字串位址放置於 r0 ,在呼叫此函式即可

參數傳遞方式

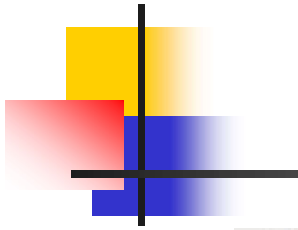
- 如何使用此種方式傳入參數

```
D:\arm-none-eabi-run a.out "hello world"
dlrow olleh
D:\>
```

  - 指令　　執行檔名　參數

```
Main(int argc, char** argv)
```

C程式執行時main將參數數量自動存到argc,參數位址存到argv,在組語則是將數量自動存到 ro ,位址存到 r1, 在寫作組與程式時就可以直接使用ro, r1來做參數相關的操作



```
 1   .text
 2   .align  2
 3   .global main
 4 main:
 5   stmfd sp!, {r0, r1, fp, lr}
 6   ldr r0, [r1,#4]
 7   bl  printf
 8   ldmfd sp!, {r0, r1, fp, lr}
 9   bx  lr
10
```

r1+4位置所存放的位址存到 r0，並呼叫printf.

** r1+4存的位址為 ”hello” 字串的位置

```
D:\>arm-none-eabi-gcc aaa.s -T generic-hosted.ld

D:\>arm-none-eabi-run a.out "hello"
hello
D:\>_
```