

組合語言 作業一

112403520李宥寬

一. 程式流程截圖、程式碼說明

```
32      push ebx
33
34      ; decrement ebx, call fact(n-1), so that eax becomes n-1!
35      dec ebx
36      call fact
37
38      ; restore n from stack to ebx, so that ebx = n
39      pop ebx
40
41      ; eax = (n-1)! * n
42      mul ebx
43
44      ret
45
46      L1:
47      mov eax, ebx
48      ret
49
50      fact ENDP
51
52      main PROC
53      mov ebx, MyID
54      call fact
55      INVOKE ExitProcess,0
56      main ENDP
57      END main
```

Registers

EAX = 004FFF60 EBX = 00000003 ECX = 007F1005
EDX = 007F1005 ESI = 007F1005 EDI = 007F1005
EIP = 007F1035 ESP = 004FFF08 EBP = 004FFF14
EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1
CY = 0

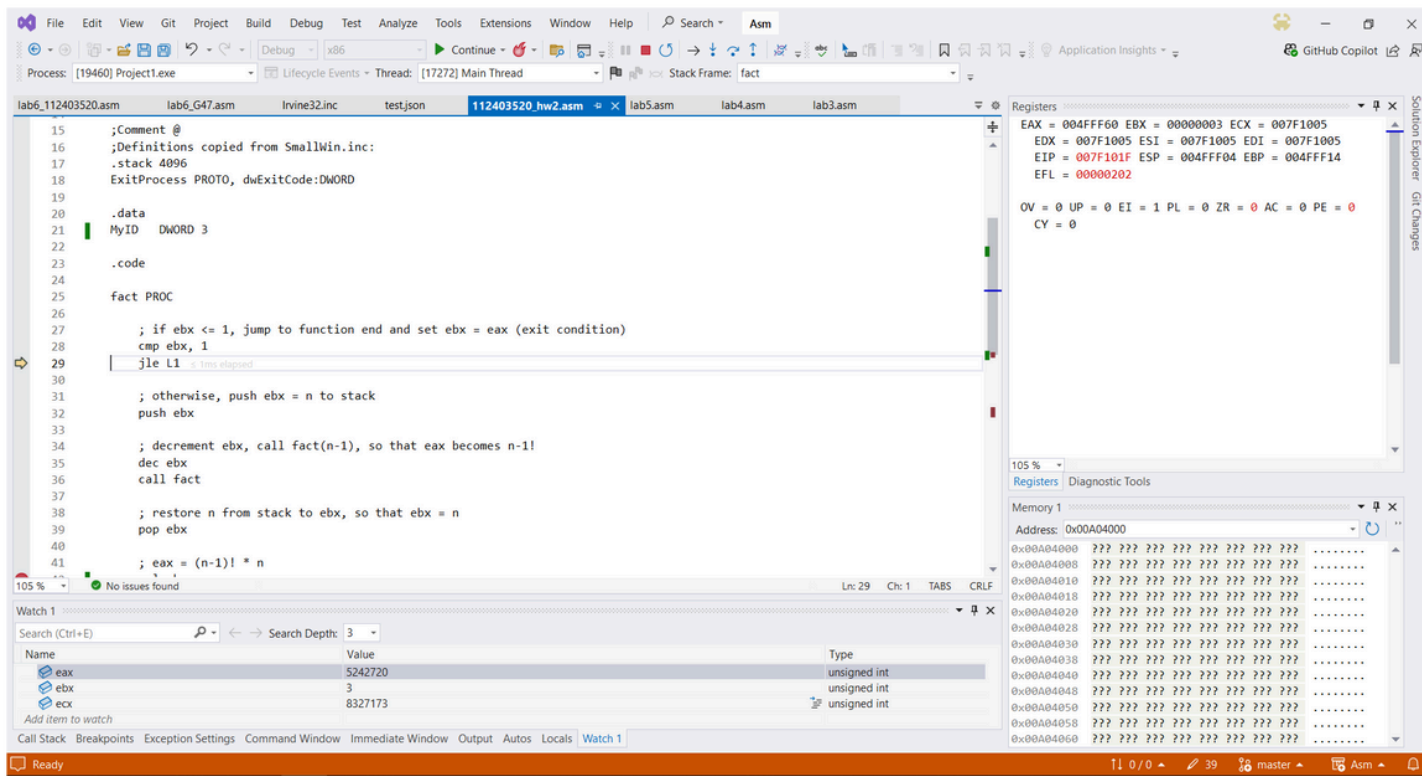
Memory 1

Address: 0x00A04000

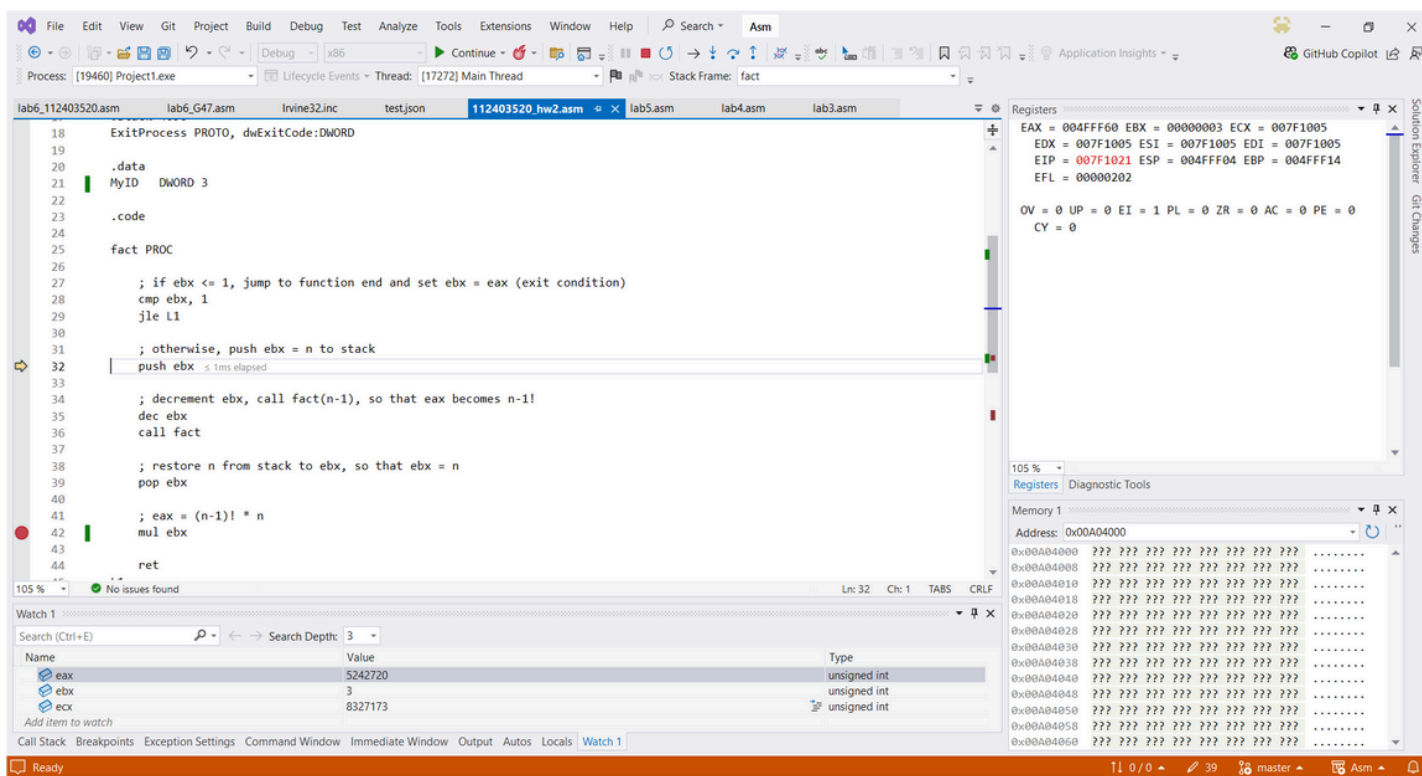
Name	Value	Type
eax	5242720	unsigned int
ebx	3	unsigned int
ecx	8327173	unsigned int

程式初始化，因為我的學號最後一碼是0，故使用MyID = 3來演示

將要計算的數字存入MyID後，呼叫fact

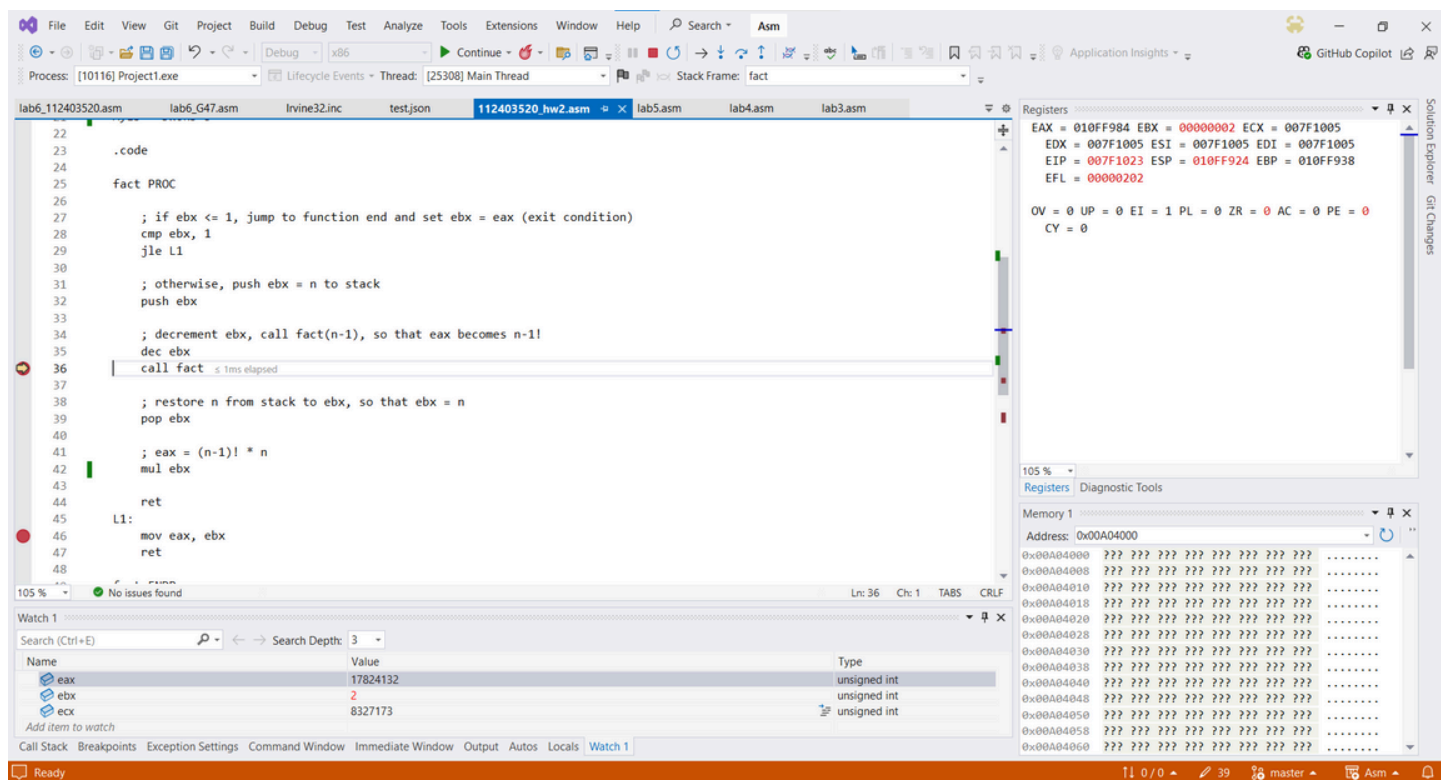


使用cmp比較ebx = n的值，若小於等於一則跳至L1



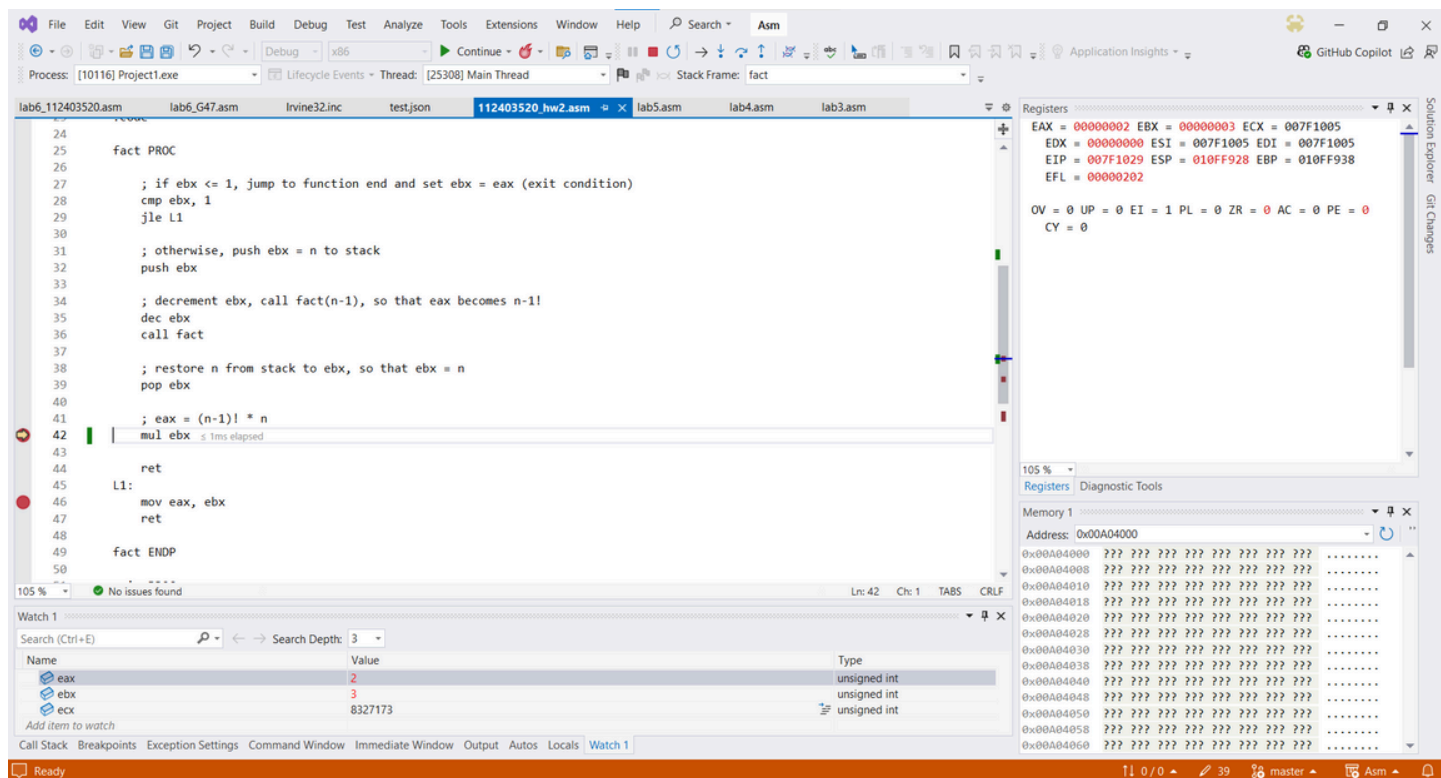
此時因為ebx還是3，程式繼續執行

因為等等會使用到ebx，將ebx先push到stack中儲存起來



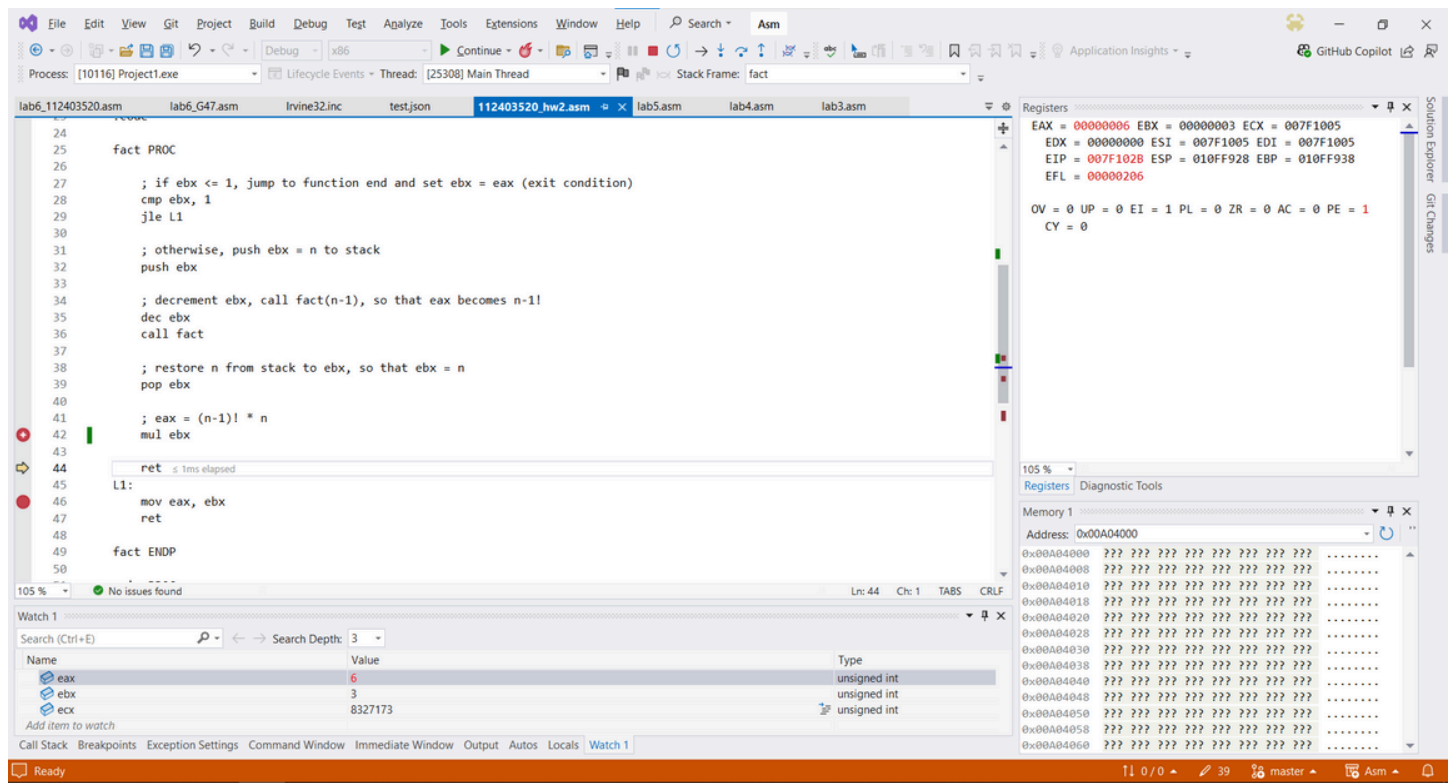
此時將ebx減一變成n-1，再call fact

得到(n - 1)!的值，並儲存在eax上



這時候再pop ebx，將剛剛儲存起來n的值放回ebx

所以此時eax = (n - 1)!, ebx = n



將eax乘以ebx，即得到 $(n - 1)! * n = n!$

所以程式的call stack可以視為:

fact(3)

fact(2)

fact(1)

→ ret 1

→ ret 2 * 1

→ ret 3 * 2 * 1

當執行到fact(1)時，會直接執行L1區段程式碼，回傳1，如下:

fact PROC

24

25

26

27 ; if ebx <= 1, jump to function end and set ebx = eax (exit condition)

28 cmp ebx, 1

29 jle L1

30

31 ; otherwise, push ebx = n to stack

32 push ebx

33

34 ; decrement ebx, call fact(n-1), so that eax becomes n-1!

35 dec ebx

36 call fact

37

38 ; restore n from stack to ebx, so that ebx = n

39 pop ebx

40

41 ; eax = (n-1)! * n

42 mul ebx

43

44 ret

45

46 L1: mov eax, ebx ; 1ms elapsed

47 ret

48

49 fact ENDP

50

Registers

EAX = 00B5FAC EBX = 00000001 ECX = 007F1005

EDX = 007F1005 ESI = 007F1005 EDI = 007F1005

EIP = 007F102C ESP = 00B5FA90 EBP = 00B5FAB0

EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1

CY = 0

Memory 1

Address: 0x00A04000

0x00A04000 60 0 0 0 232 86 1 0 <...?V...

0x00A04008 1 0 0 0 1 0 0 0 <...?V...

0x00A04010 252 86 1 0 64 0 0 0 ?V...@...

0x00A04018 60 0 0 0 60 87 1 0 <...?W...

0x00A04020 1 0 0 0 1 0 0 0 <...?W...

0x00A04028 80 87 1 0 66 0 0 0 PW...B...

0x00A04030 62 0 0 0 148 87 1 0 >...?W...

0x00A04038 1 0 0 0 1 0 0 0 <...?W...

0x00A04040 168 87 1 0 76 0 0 0 ?W...L...

0x00A04048 72 0 0 0 244 87 1 0 H...?W...

0x00A04050 1 0 0 0 1 0 0 0 <...?W...

0x00A04058 8 88 1 0 56 0 0 0 .X...8...

0x00A04060 52 0 0 0 64 88 1 0 4...@X...

Watch 1

Name Value Type

eax 11926268 unsigned int

ebx 1 unsigned int

ecx 8327173 unsigned int

L1區段將ebx的值複製到eax作為結果傳回

41 ; eax = (n-1)! * n

42 mul ebx

43

44 ret

45

46 L1: mov eax, ebx

47 ret

48

49 fact ENDP

50

51 main PROC

52 mov ebx, MyID

53 call fact

54 INVOKE ExitProcess,0 ; 1ms elapsed

55 main ENDP

56 END main

Registers

EAX = 00000006 EBX = 00000003 ECX = 007F1005

EDX = 00000000 ESI = 007F1005 EDI = 007F1005

EIP = 007F103A ESP = 004FF08 EBP = 004FF14

EFL = 00000206

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 0 AC = 0 PE = 1

CY = 0

Memory 1

Address: 0x00A04000

0x00A04000 ??? ??? ??? ??? ??? ??? ??? ???

0x00A04008 ??? ??? ??? ??? ??? ??? ??? ???

0x00A04010 ??? ??? ??? ??? ??? ??? ??? ???

0x00A04018 ??? ??? ??? ??? ??? ??? ??? ???

0x00A04020 ??? ??? ??? ??? ??? ??? ??? ???

0x00A04028 ??? ??? ??? ??? ??? ??? ??? ???

0x00A04030 ??? ??? ??? ??? ??? ??? ??? ???

0x00A04038 ??? ??? ??? ??? ??? ??? ??? ???

0x00A04040 ??? ??? ??? ??? ??? ??? ??? ???

0x00A04048 ??? ??? ??? ??? ??? ??? ??? ???

0x00A04050 ??? ??? ??? ??? ??? ??? ??? ???

0x00A04058 ??? ??? ??? ??? ??? ??? ??? ???

0x00A04060 ??? ??? ??? ??? ??? ??? ??? ???

Watch 1

Name Value Type

eax 6 unsigned int

ebx 3 unsigned int

ecx 8327173 unsigned int

得到最終結果eax = 3! = 6

二. 完成的程式畫面截圖

```
20      .data
21      MyID    DWORD 0
```

data 區段，將MyID設為學號最後一碼0

程式區段

```
23      .code
24
25      fact PROC
26
27          ; if ebx <= 1, jump to function end and set ebx = eax (exit condition)
28          cmp ebx, 1
29          jle L1
30
31          ; otherwise, push ebx = n to stack
32          push ebx
33
34          ; decrement ebx, call fact(n-1), so that eax becomes n-1!
35          dec ebx
36          call fact
37
38          ; restore n from stack to ebx, so that ebx = n
39          pop ebx
40
41          ; eax = (n-1)! * n
42          mul ebx
43
44          ret
45      L1:
46          mov eax, ebx
47          ret
48
49      fact ENDP
```

25~49 計算階乘的函式


```
51      main PROC
52          mov ebx, MyID
53          call fact
54          INVOKE ExitProcess,0
55      main ENDP
```

51~55 程式進入點

三. 學習心得

學習收穫:

- 組合語言的細節:
 - 這次作業讓我更加深入地了解組合語言的指令集和記憶體操作，還有函數呼叫真正的意義。
- 遞迴的原理:
 - 透過這個例子，我對遞迴的原理和應用有了更深刻的認識。
- 函數呼叫和堆疊:
 - 遞迴的實現離不開函數呼叫和堆疊的使用，這部分的知識也是非常重要的。
- 演算法效率:
 - 遞迴雖然寫法簡潔，但效率可能不是最佳的。在實際應用中，需要根據具體情況選擇合適的演算法。

遞迴的優缺點

- 優點:
 - 程式結構清晰: 遞迴的寫法通常比較直觀，符合數學上的遞迴定義。
 - 代碼簡潔: 相較於其他的方法，遞迴的程式碼往往更簡潔。
- 缺點:
 - 程式可讀性: 相較於線性或迴圈程式，遞迴較難分析及除錯
 - 容易發生堆疊溢位(stack overflow): 可以注意到每次呼叫下個遞迴，都要先把變數除存在 stack，並將return address也push到stack上。如果遞迴層次過深，可能會導致堆疊溢位。