Contents

1 hw 0x4

1.1 gcd inc

```
INCLUDE Irvine32.inc  ; include Irvine32.inc 進 main.asm 和 gcd.asm

gcd PROTO       ; 有一個在 gcd.asm 的 proc 叫做 gcd
```

1.2 main asm

```
; 409262449 吳家萱
.686
.model flat, stdcall
.stack 4096

INCLUDE gcd.inc

.code

main PROC

begin:
  push ebp
  mov ebp, esp
  sub esp, 8              ; 開 8 格存 m 和 n

  call ReadInt
  cmp eax, 0
  jle fin                 ; 有號數小於等於就跳轉 如果輸入小於 0 就判斷結束

  push eax                ; 把 n 存進 stack 為了傳給 gcd.asm
  mov DWORD PTR[ebp-4], eax     ; 把 n 存進這裡的 stack 為了下面求 lcm 要用

  call ReadInt
  push eax                ; 把 m 也存進 stack
  mov DWORD PTR[ebp-8], eax

  INVOKE gcd              ; call gcd.asm

  mov ebx, eax            ; 先把 gcd 傳回來的 eax(gcd) 存進沒有在用 ebx
  call WriteDec
  mov al, ' '             ; 輸出空白
  call WriteChar

  mov eax, DWORD PTR[ebp-8]
  mul DWORD PTR[ebp-4]        ; lcm = m*n / gcd

  cdq
  div ebx
  call WriteDec
  call Crlf

  jmp begin               ; 回來後繼續新的一趟吃輸入

fin:                  ; 函式結束
  mov esp, ebp            ; 最後也要記得清空
  pop ebp
  INVOKE ExitProcess, 0       ; 結束

main ENDP

END main
```

1.3 gcd asm

```
INCLUDE gcd.inc

.code

gcd PROC
  push ebp            ; 把 ebp 的 old address 存起來
  mov ebp, esp          ; 要把 ebp 拉到 esp 的位址
  sub esp, 4            ; 用 esp 向下開一個新空間 Rem

  mov ecx, DWORD PTR[ebp+12]    ; 現在 eax 裡面放的是 m
  mov ebx, DWORD PTR[ebp+8]   ; 現在 ebx 裡面放的是 n

  mov eax, ebx          ; div  的被除數 是 eax
  cdq               ; 清空 edx
  div ecx             ; eax/ecx = eax...edx
  cmp edx, 0            ; 如果不餘 0 就繼續遞迴
  jne Euclidean_Algorithm

  mov eax, ecx          ; 把最大公因數 (現在在 ecx 裡) 存給 eax, eax 個檔案共用 所以回去 main 的時候也還是最大公因數
  mov esp, ebp          ; 把 ebp 指到現在 esp 的位址
  pop ebp             ; 叫出 ret address
  ret 8             ; 清空 8 格 stack 房間 (清空舊的 n 和 m)

Euclidean_Algorithm:        ; 輾轉相除法 遞迴 ver.
  mov DWORD PTR[ebp-4], edx   ; 把現在的餘數 edx 放進 Rem (DWORD PTR[ebp-4])
  push DWORD PTR[ebp-4]     ; 把餘數(n%m) push 進去
  push ecx            ; 把 m 放進去遞迴
  call gcd            ; 遞迴 gcd

  mov esp, ebp          ; 最後他一層一層 ret 回來的時候繼續清空舊的 n 和 m
  pop ebp
  ret 8

gcd ENDP

END gcd
```

2 hw 0x5

2.1 header inc

```
INCLUDE Irvine32.inc  ; include Irvine32.inc 進 main.asm 和 gcd.asm

input PROTO,
  n: DWORD

sort PROTO,
  n: DWORD,
  arr: DWORD

```

```
output PROTO,
  n: DWORD,
  arr: DWORD
```

2.2 main asm

```
.686
.model flat, stdcall
.stack 4096

INCLUDE header.inc

.data
  flag DWORD ?
  N DWORD ?

.code

main PROC

  mov eax, 0
  mov flag, eax

begin:
  push ebp              ; 把 ebp 的 old address 存起來
  mov ebp, esp            ; 要把 ebp 拉到 esp 的位址

  call ReadInt
  cmp eax, 0
  jle fin               ; 如果輸入等於 0 就判斷結束

  push eax              ; 把 n 存進 stack
      為了傳給其他 .asm
  mov N, eax

  call ainput

  cmp flag, 0
  je noendl

  call Crlf

noendl:
  push edi              ; push 存 head
  push N                ; push N
  call sort

  mov eax, 1
  mov flag, eax

  mov eax, N
  mov ebx, 2
  cdq
  div ebx
  cmp edx, 1
  je odd

  mov esi,eax

  mov eax, [edi+esi*4]
  cmp eax, 0
  jg positive1

  call WriteInt
  jmp space

positive1:
  call WriteDec

space:
  mov al, ' '
  call WriteChar

  dec esi
  mov eax, [edi+esi*4]
  cmp eax, 0
  jg positive2

  call WriteInt
  jmp endl

positive2:
  call WriteDec

endl:
  call Crlf
  jmp begin

odd:
  mov esi,eax

  mov eax, [edi+esi*4]
  cmp eax, 0
  jg positive3

  call WriteInt
  jmp endl

positive3:
  call WriteDec
  jmp endl            ; 回來後繼續新的一趟吃輸入

fin:                  ; 函式結束
  mov esp, ebp            ; 最後也要記得清空
  pop ebp
  INVOKE ExitProcess, 0       ; 結束

main ENDP

END main
```

2.3 input asm

```
INCLUDE header.inc

.data
  array DWORD 1000 DUP(?)

.code

input PROC,             ; 在 PROTO 有宣告參數
    這裡也要宣告
  n: DWORD

  ;push ebp           ; 有 PROC, n:DWORD assembly
      就會自動 push ebp
  mov ebp, esp

  mov ecx, DWORD PTR[ebp+8]   ; ecx 存 N
      (記憶體位址在現在的 ebp 往上 +8)
  mov ebx, ecx          ; 因為往後陣列頭是固定的
      所以要先存進別的暫存器

  lea edi, array          ; 用 lea 不用 OFFSET
      array，一樣是直接傳 array 頭的記憶體"位址"
  mov esi, 0            ; 先將 esi 設為 0 (現在 esi
      相當於 c++ 程式中的 i)

L1:
  call ReadInt          ; 建構 array
  mov [edi+esi], eax        ; 在陣列頭 edi +
      esi(i*4格) 的位址存每一格的數值
  add esi, 4            ; 手動把 esi 每次都 +4
      開新空間

  dec ebx
  cmp ebx, 0
  jne L1
```

```
28
29 fin:                ; 函式結束
30   ;mov esp, ebp
31   ;pop ebp            ; 不用 pop 因為前面也沒有 push
32   ret 8              ; 要還 PROC, n:DWORD assembly
         借用的空間
33
34 input ENDP
35
36 END                  ; 這邊不用接 input
       因為只是副程式的結束而已
```

2.4 sort asm

```
1  INCLUDE header.inc
2  .data
3    min DWORD ?
4
5  .code
6  sort PROC,            ; 在 PROTO 有宣告參數
       這裡也要宣告
7    n: DWORD,
8    arr: DWORD
9
10   ;push ebp
11   mov ebp, esp
12
13   mov ecx, DWORD PTR[ebp+8]   ; ecx 存 N
14   mov edi, DWORD PTR[ebp+12]    ; edi 存 head
15
16   mov esi, 0            ; esi = i (as c++)
17
18 L1:
19   mov eax, ecx
20   dec eax
21   cmp eax, esi          ; if i == n, jmp
22   je fin
23
24   mov eax, [edi+esi*4]
25   mov min, esi          ; c++ 裡的 min = i
26
27   push esi              ; 先把 esi = i 存起來
28   add esi, 1
29 L2:
30                    ; j = i + 1 = esi + 4
31   cmp ecx, esi          ; if j == n , jmp
32   je swap
33
34   mov eax, [edi+esi*4]
35   mov edx, min
36   cmp eax, [edi+edx*4]      ; 比較 array[j] 和
         array[min]
37   jge conti             ; 大於等於就跳轉
38
39   mov min, esi          ; min = j
40
41 conti:
42
43   add esi, 1        ; j++
44   jmp L2
45
46 swap:
47   pop esi
48   mov eax, [edi+esi*4]      ; tmp = array[i]
49   mov edx, min
50   mov ebx, [edi+edx*4]      ; tmp = array[min]
51   xchg eax, ebx         ; swap array[i], array[min]
52   mov [edi+esi*4], eax
53   mov [edi+edx*4], ebx
54
55   mov ebx, esi
56   push edi
57   push ecx              ; 重新 push N
58   call output
59
60   mov esi, ebx
61   add esi, 1
62   jmp L1
63
64 fin:                ; 函式結束
65   mov esp, ebp          ; 最後也要記得清空
66   ;pop ebp
67   ret 8             ; 結束
68
69 sort ENDP
70
71 END
```

2.5 output asm

```
1  INCLUDE header.inc
2
3  .code
4
5  output PROC,              ; 在 PROTO 有宣告參數
       這裡也要宣告
6    n: DWORD,
7    arr: DWORD
8
9    ;push ebp
10   mov ebp, esp
11
12   mov edi, DWORD PTR[ebp+12]      ; edi 存 head
13   mov ecx, DWORD PTR[ebp+8]     ; 在 input 已經存過 N
         了
14
15   mov esi, 0               ; esi = i (as c++)
16
17 L1:
18   cmp ecx, esi
19   je fin
20
21   mov eax, [edi+esi*4]
22   cmp eax, 0
23   jge positive
24
25   call WriteInt
26   jmp space
27
28 positive:
29   call WriteDec
30
31 space:
32   mov al, ' '
33   call WriteChar
34
35   inc esi
36   jmp L1
37
38
39 fin:                   ; 函式結束
40   call Crlf
41   mov esp, ebp             ; 最後也要記得清空
42   ;pop ebp
43   ret 8
44
45 output ENDP
46
47 END
```