

## Contents

1	hw 0x4	44
1.1	gcd inc	45
1.2	main asm	46
1.3	gcd asm	47
2	hw 0x5	48
2.1	header inc	49
2.2	main asm	50
2.3	input asm	51
2.4	sort asm	52
2.5	output asm	53

## 1 hw 0x4

### 1.1 gcd inc

```
1 INCLUDE Irvine32.inc ; include Irvine32.inc 進
   main.asm 和 gcd.asm
2
3 gcd PROTO ; 有一個在 gcd.asm 的 proc 叫做 gcd
```

### 1.2 main asm

```
1 ; 409262449 吳家萱
2 .686
3 .model flat, stdcall
4 .stack 4096
5
6 INCLUDE gcd.inc
7
8 .code
9
10 main PROC
11
12 begin:
13     push ebp
14     mov ebp, esp
15     sub esp, 8 ; 開 8 格存 m 和 n
16
17     call ReadInt
18     cmp eax, 0
19     jle fin ; 有號數小於等於就跳轉
   如果輸入小於 0 就判斷結束
20
21     push eax ; 把 n 存進 stack 為了傳給
   gcd.asm
22     mov DWORD PTR[ebp-4], eax ; 把 n 存進這裡的
   stack 為了下面求 lcm 要用
23
24     call ReadInt
25     push eax ; 把 m 也存進 stack
26     mov DWORD PTR[ebp-8], eax
27
28     INVOKE gcd ; call gcd.asm
29
30     mov ebx, eax ; 先把 gcd 傳回來的
   eax(gcd) 存進沒有在用 ebx
31     call WriteDec
32     mov al, ' ' ; 輸出空白
33     call WriteChar
34
35     mov eax, DWORD PTR[ebp-8]
36     mul DWORD PTR[ebp-4] ; lcm = m*n / gcd
37
38     cdq
39     div ebx
40     call WriteDec
41     call Crlf
42
43     jmp begin ; 回來後繼續新的一趟吃輸入
```

```
44
45 fin: ; 函式結束
46     mov esp, ebp ; 最後也要記得清空
47     pop ebp
48     INVOKE ExitProcess, 0 ; 結束
49
50 main ENDP
51
52 END main
```

### 1.3 gcd asm

```
1 INCLUDE gcd.inc
2
3 .code
4
5 gcd PROC
6     push ebp ; 把 ebp 的 old address 存起來
7     mov ebp, esp ; 要把 ebp 拉到 esp 的位址
8     sub esp, 4 ; 用 esp 向下開一個新空間 Rem
9
10    mov ecx, DWORD PTR[ebp+12] ; 現在 eax 裡面放的是
   m
11    mov ebx, DWORD PTR[ebp+8] ; 現在 ebx 裡面放的是 n
12
13    mov eax, ebx ; div 的被除數 是 eax
14    cdq ; 清空 edx
15    div ecx ; eax/ecx = eax...edx
16    cmp edx, 0 ; 如果不餘 0 就繼續遞迴
17    jne Euclidean_Algorithm
18
19    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
20    mov esp, ebp ; 把 ebp 指到現在 esp 的位址
21    pop ebp ; 叫出 ret address
22    ret 8 ; 清空 8 格 stack 房間 (清空舊的
   n 和 m)
23
24 Euclidean_Algorithm: ; 輾轉相除法 遞迴 ver.
25     mov DWORD PTR[ebp-4], edx ; 把現在的餘數 edx 放進
   Rem (DWORD PTR[ebp-4])
26     push DWORD PTR[ebp-4] ; 把餘數(n%m) push 進去
27     push ecx ; 把 m 放進去遞迴
28     call gcd ; 遞迴 gcd
29
30     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
31     pop ebp
32     ret 8
33
34 gcd ENDP
35
36 END gcd
```

## 2 hw 0x5

### 2.1 header inc

```
1 INCLUDE Irvine32.inc ; include Irvine32.inc 進
   main.asm 和 gcd.asm
2
3 input PROTO,
4     n: DWORD
5
6 sort PROTO,
7     n: DWORD,
8     arr: DWORD
9
```

```

10 output PROTO,
11     n: DWORD,
12     arr: DWORD

```

## 2.2 main asm

```

1  .686
2  .model flat, stdcall
3  .stack 4096
4
5  INCLUDE header.inc
6
7  .data
8      flag DWORD ?
9      N DWORD ?
10
11 .code
12
13 main PROC
14
15     mov eax, 0
16     mov flag, eax
17
18 begin:
19     push ebp                ; 把 ebp 的 old address 存起來
20     mov ebp, esp           ; 要把 ebp 拉到 esp 的位址
21
22     call ReadInt
23     cmp eax, 0
24     jle fin                ; 如果輸入等於 0 就判斷結束
25
26     push eax                ; 把 n 存進 stack
27                             ; 為了傳給其他 .asm
28     mov N, eax
29
30     call ainput
31
32     cmp flag, 0
33     je noendl
34
35     call Crlf
36
37 noendl:
38     push edi                ; push 存 head
39     push N                  ; push N
40     call sort
41
42     mov eax, 1
43     mov flag, eax
44
45     mov eax, N
46     mov ebx, 2
47     cdq
48     div ebx
49     cmp edx, 1
50     je odd
51
52     mov esi, eax
53
54     mov eax, [edi+esi*4]
55     cmp eax, 0
56     jg positive1
57
58     call WriteInt
59     jmp space
60
61 positive1:
62     call WriteDec
63
64 space:
65     mov al, ' '
66     call WriteChar
67
68     dec esi
69     mov eax, [edi+esi*4]

```

```

69     cmp eax, 0
70     jg positive2
71
72     call WriteInt
73     jmp endl
74
75 positive2:
76     call WriteDec
77
78 endl:
79     call Crlf
80     jmp begin
81
82 odd:
83     mov esi, eax
84
85     mov eax, [edi+esi*4]
86     cmp eax, 0
87     jg positive3
88
89     call WriteInt
90     jmp endl
91
92 positive3:
93     call WriteDec
94     jmp endl                ; 回來後繼續新的一趟吃輸入
95
96 fin:                        ; 函式結束
97     mov esp, ebp           ; 最後也要記得清空
98     pop ebp
99     INVOKE ExitProcess, 0    ; 結束
100
101 main ENDP
102
103 END main

```

## 2.3 input asm

```

1  INCLUDE header.inc
2
3  .data
4      array DWORD 1000 DUP(?)
5
6  .code
7
8  input PROC,                ; 在 PROTO 有宣告參數
9      n: DWORD              ; 這裡也要宣告
10
11     ;push ebp                ; 有 PROC, n:DWORD assembly
12                             ; 就會自動 push ebp
13     mov ebp, esp
14
15     mov ecx, DWORD PTR[ebp+8] ; ecx 存 N
16                             ; (記憶體位址在現在的 ebp 往上 +8)
17     mov ebx, ecx             ; 因為往後陣列頭是固定的
18                             ; 所以要先進別的暫存器
19
20     lea edi, array           ; 用 lea 不用 OFFSET
21                             ; array, 一樣是直接傳 array 頭的記憶體 "位址"
22     mov esi, 0               ; 先將 esi 設為 0 (現在 esi
23                             ; 相當於 c++ 程式中的 i)
24
25 L1:
26     call ReadInt             ; 建構 array
27     mov [edi+esi], eax       ; 在陣列頭 edi +
28                             ; esi(i*4格) 的位址存每一格的數值
29     add esi, 4               ; 手動把 esi 每次都 +4
30                             ; 開新空間
31
32     dec ebx
33     cmp ebx, 0
34     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    mov esi, ebx
60    add esi, 1
61    jmp L1
62
63 fin:                ; 函式結束
64    mov esp, ebp        ; 最後也要記得清空
65    ;pop ebp
66    ret 8               ; 結束
67
68 sort ENDP
69
70 END
71

```

## 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

```