Contents

1 exams

1.1 star pyramid

```
.686                    ; 409262449 吳家萱 10/27 P1
.model flat, stdcall
.stack 4096

INCLUDE Irvine32.inc
ExitProcess PROTO, dwExitCode: DWORD

.data
n SDWORD ?                  ; 變數宣告
space SDWORD ?
b SDWORD ?
format1 BYTE " ", 0
format2 BYTE "*", 0

.code
main PROC

conti:
  call ReadDec              ; 用 Dec 讀取
  mov n, eax                ; 讀取n
  mov eax, n
  mov space, eax
  cmp n, 0                  ; 如果 n == 0 結束程式
  je finish
  mov b, 1                  ; 初始化 b

  mov eax, n                ; 要輸出的星星層數
  add eax, n
  mov n, eax
  mov ecx, n
  call printree

  jmp conti

finish:
  invoke ExitProcess, 0       ; 離開程式
  main ENDP

printree PROC

L1:
  push ecx
  mov eax, space
  sub eax, b
  cmp eax, 0                ; 如果 eax < 0
      停止迴圈繼續往下
  jbe cont
  mov ecx, eax
L2:                     ; 迴圈空白
  mov edx, OFFSET format1
  call WriteString
  loop L2

cont:
  mov eax, b
  add eax, b
  sub eax, 1
  mov ecx, eax
L3:                     ; 迴圈*
  mov edx, OFFSET format2
  call WriteString
  loop L3

  call Crlf

  pop ecx               ; 把 L1 的 ecx 拿出來 - 1
  mov eax, ecx
  sub eax, 1
  mov ecx, eax
  mov eax, b
  add eax, 1
  mov b, eax
  loop L1

  ret                   ; 要記得 ret QQ
      不然會超出組建範圍
  printree ENDP

END main
```

2 homeworks

2.1 print prime asm

```
.686p
.model flat, C
.stack 4096

includelib legacy_stdio_definitions.lib

ExitProcess PROTO, dwExitCode: DWORD

printf PROTO C,   ; Std C library function
  format: PTR BYTE, args: VARARG    ; (in libc.lib)

scanf PROTO C,      ; STd C library function
  format: PTR BYTE, args: VARARG    ; (in libc.lib)

AllPrimeNumbers PROTO C,
     dwInt: DWORD           ; in main.cpp

TAB = 9

.data
  ;  define your variables here
format1 BYTE "Give me an interger owo : ", 0

format2 BYTE "%d", 0

format3 BYTE "OK, so the integer N is... %d !",0dh,
    0ah, 0

format4 BYTE "%d",0dh, 0ah, 0

format5 BYTE "And all prime numbers less than N is :
    ", 0

newLine BYTE 0dh, 0ah, 0

space BYTE " ", 0

Number DWORD ?

.code

asmMain PROC C
    push ebp
  mov ebp, esp

  ;  write your assembly code here
  INVOKE printf, ADDR format1

  INVOKE printf, ADDR newLine ; 換行
```

```
48
49    INVOKE scanf, ADDR format2, ADDR Number    ;
          scanf("%d", &Number)
50
51    INVOKE printf, ADDR format3, Number      ; printf("N
          : %d\n", Number)
52
53    INVOKE printf, ADDR format5
54
55    INVOKE printf, ADDR newLine
56
57    INVOKE AllPrimeNumbers,  Number      ; return value
          in EAX
58
59    INVOKE printf, ADDR Format4, eax
60
61    mov eax, 0     ; return 0
62    pop ebp
63
64    ret
65
66  asmMain ENDP
67
68
69  END
```

2.2 print prime cpp

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <string.h>
4
5  using namespace std;
6
7  extern "C" int asmMain();
8
9  extern "C" int AllPrimeNumbers(int);
10
11 int main()
12 {
13   int num = asmMain();
14
15   system("pause");
16
17   return 0;
18 }
19
20 int AllPrimeNumbers(int num)
21 {
22   int cnt = 0;
23   bool prime[1000000];
24
25   memset(prime, 0, sizeof(prime));
26
27     for (int i = 2; i < num; i++) {
28
29         if (!prime[i]) {
30
31       printf("%d ", i);
32       cnt++;
33
34             for (int j = i + i; j < num; j += i) {
35
36                 prime[j] = true;
37
38             }
39         }
40     }
41   printf( "\nCnt is : " );
42
43   return cnt;
44 }
```

2.3 max and min

```
1  .686                    ; 指定微處理器模式
2  .model flat, stdcall           ; model ->
       定義記憶體模式, stdcall -> windows API
       的標準呼叫約定
3  .stack 4096                ; 堆疊 ->
       用來保存傳遞的參數 & 保存調用函數的程式位址
4
5  INCLUDE Irvine32.inc
6  ExitProcess PROTO, dwExitCode: DWORD  ; windos 服務,
       原型包含函數名, PROTO 關鍵字, 輸入參數列表
7
8  .data                   ; 組譯時配置
9  arr SDWORD 200 DUP(0)          ; 陣列宣告 SDWORD
10 n SDWORD ?                 ; 變數宣告
11 small SDWORD ?
12 large SDWORD ?
13 format1 BYTE "Min = ", 0         ; 輸出字串宣告
14 format2 BYTE "Max = ", 0
15
16 .code
17
18 main PROC
19   call ReadInt              ; Irvine32.lib 內建函式之一
       可用來讀取 Int
20   mov esi, OFFSET arr          ; 把 arr 的位址放進 esi
       暫存器
21   mov n, eax                ; ReadInt
       讀取的輸入會先暫存在 eax 中 所以要 mov 進 n 裡
22   mov ecx, n                ; 迴圈寫法, ecx 是 計數器,
       原理: ecx != 0 繼續跑 -> 總共跑 n 次
23
24 L1:                     ; 這邊在做讀取 input 的單層 for
     迴圈的工作
25   call ReadInt
26   mov [esi], eax              ; 把 eax 讀取輸入的值放入
       arr, esi 指向陣列位址的頭
27   add esi, TYPE arr           ; 往上指下一位 ( TYPE
       arry 大小 ) 的 esi
28   loop L1                 ; 迴圈循環
29
30   sub n, 1                ; n - 1 因為外層迴圈只要跑 n
       - 1 次就好
31   mov ecx, n                ; 在 call mini proc 前先把
       ecx 準備好
32   mov esi, OFFSET arr          ; 把 esi 歸 0
       回到記憶體位址的頭
33   call mini
34
35   mov edx, OFFSET format1        ; 因為 WriteString 吃
       edx 暫存器
36   call WriteString
37   cmp eax, 0                ; 如果 small 是正整數 or 0
38   jge positive1
39   call WriteInt             ; WriteInt 吃 eax,
       輸出負值的 small
40   call Crlf               ; 內建換行寫法
41   jmp conti               ; 如果這邊不跳過
       程式就會繼續往下執行 positive1
42
43 positive1:                  ; 如果 small 是正數
44   call WriteDec             ; 就輸出 decimal ( 沒有負數
       )
45   call Crlf
46
47 conti:
48   mov ecx, n
49   mov esi, OFFSET arr
50   call maxi
51
52   mov edx, OFFSET format2
53   call WriteString
```

```
54    cmp eax, 0
55    jge positive2
56    call WriteInt
57    call Crlf
58    jmp finish
59
60  positive2:
61    call WriteDec
62    call Crlf
63
64  finish:
65    invoke ExitProcess, 0       ; 離開程式
66    main ENDP             ; main 結束
67
68  mini PROC
69    push esi              ; 先把 esi 目前位址指向處
          push 放入堆疊中
70    push ecx              ; 同理, 放入堆疊是因為等等
          pop 出去後暫存器會回到 push 進去的狀態,
          所以內部就算有改動也沒關係
71    mov eax, [esi]
72    mov small, eax            ; 先預設 small 初始值是
          arr[0]
73    add esi, TYPE arr
74    mov ecx, n
75  L2:
76    mov eax, [esi]
77    add esi, TYPE arr
78    cmp eax, small            ;
          因為不能兩邊都是記憶體位址, 所以要先 mov 進 eax
          才去做比較
79    jge skip1             ; 如果大於等於就跳到 skip1
80    mov small, eax            ; 否則, 就重新紀錄當前
          small 是誰
81  skip1:
82    loop L2
83
84    mov eax, small            ; 等等回到 main proc 後
          WriteInt 要用到 eax, 現在先存好
85    pop ecx               ; 後進去 ecx 的要先出來
86    pop esi               ; 要 pop 是為了讓他們回到進來
          mini proc 前的狀態
87    ret                 ; return 回去
88
89    mini ENDP
90
91  maxi PROC               ; 所有理論皆與 mini 同理
92    push esi
93    push ecx
94    mov eax, [esi]
95    mov large, eax
96    add esi, TYPE arr
97    mov ecx, n
98  L3:
99    mov eax, [esi]
100   add esi, TYPE arr
101   cmp eax, large
102   jle skip2
103   mov large, eax
104 skip2:
105   loop L3
106
107   mov eax, large
108   pop ecx
109   pop esi
110   ret
111
112   maxi ENDP
113
114 END main
```