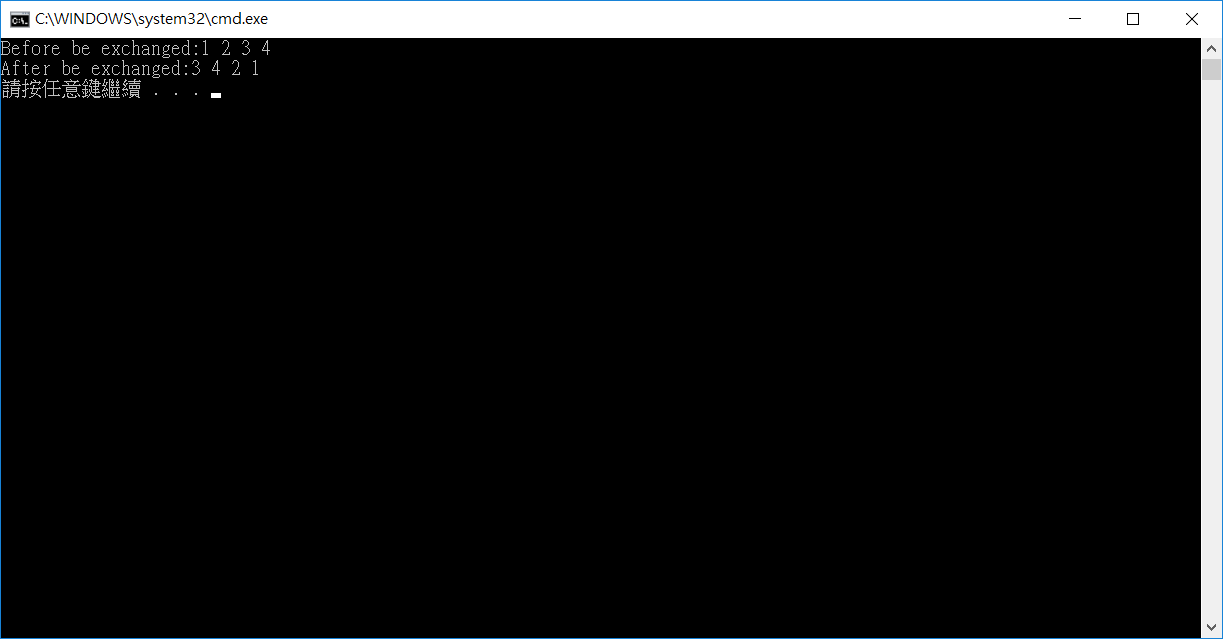
**Assembly Homework Ch 4**

**Your Name: 謝豐安 Student ID (學號):A1055548**

1. Print your executing results 20% (貼上執行結果20%)

Problem1



2. Source Code 20% and Comments 60% (程式原始碼20%與程式註解說明50%)

INCLUDE Irvine32.inc

.386

.model flat,stdcall

.stack 4096

ExitProcess PROTO,dwExitCode:DWORD

.data

arrayD DWORD 1,2,3,4

str1 BYTE "Before be exchanged:",0

str2 BYTE "After be exchanged:",0

str3 BYTE " ",0

A DWORD ?

B DWORD ?

.code

main PROC

mov edx , offset str1 ;move address of str1 to edx

call WriteString ;call str1

mov edx , offset str3 ;move address of str3 to edx

;----------move arrayD[1~4] to eax and show----------

mov eax , arrayD

call WriteDec

call WriteString

mov eax , arrayD+4

call WriteDec

call WriteString

mov eax , arrayD+8

call WriteDec

call WriteString

mov eax , arrayD+12

call WriteDec

call WriteString

call Crlf

;----------exchange segment----------

mov eax , arrayD+8 ;eax=3

mov A , eax ;A=3

mov eax , arrayD+12 ;eax=4

mov B , eax ;B=4

mov eax , arrayD ;eax=1

mov arrayD+12 , eax ;arrayD[4]=1

mov eax , arrayD+4 ;eax=2

mov arrayD+8 , eax ;arrayD[3]=2

mov eax , A ;eax=3

mov arrayD , eax ;arrayD[1]

mov eax , B ;eax=4

mov arrayD+4 , eax ; arrayD[2]=4

mov edx , offset str2 ;move address of str2 to edx

call WriteString ;call str1

mov edx , offset str3 ;move address of str3 to edx

;----------move arrayD[1~4] to eax and show----------

mov eax , arrayD

call WriteDec

call WriteString

mov eax , arrayD+4

call WriteDec

call WriteString

mov eax , arrayD+8

call WriteDec

call WriteString

mov eax , arrayD+12

call WriteDec

call WriteString

call Crlf

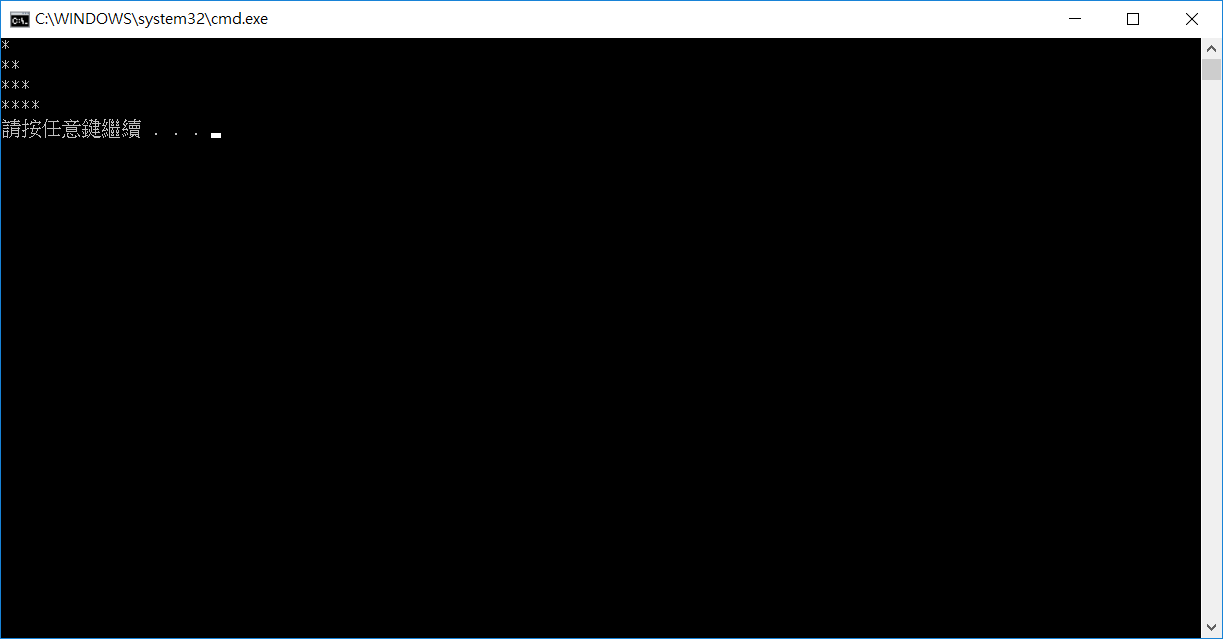
INVOKE ExitProcess,0

main ENDP

END main

1. Print your executing results 20% (貼上執行結果20%)

Problem2



2. Source Code 20% and Comments 60% (程式原始碼20%與程式註解說明50%)

INCLUDE Irvine32.inc

.386

.model flat,stdcall

.stack 4096

ExitProcess PROTO,dwExitCode:DWORD

; 程式說明:

; 使用 LoopTemp 變數暫存L1的counter , OurCount儲存L2的counter且每次L2執行完都讓OurCount+1

; 每次L1開始時需把ecx儲存下來 , 再讓L2需要跑多少次的值移動到ecx給L2做迴圈執行

; L2中就只是印星號 , L2結束後會回復L1所需的counter值給ecx,並換行

.data

LoopTemp DWORD ? ;store the value of counter of loop L1

OurCount DWORD 1 ;store the value of counter of loop L2

star BYTE "\*",0

.code

main PROC

mov ecx,4 ;move times of L1

mov edx,offset star ;move the address of star to edx

L1:

mov LoopTemp , ecx ;store ecx for L1

mov ecx , OurCount ;move OurCount to ecx for L2 that can run proper times

L2:

call WriteString ;Write the String star

loop L2 ;back to L2 if ecx equal zero

inc OurCount ;make the L2 can run 1,2,3,and 4 times, so need +1 to OurCount

call Crlf ;new line

mov ecx , LoopTemp ;back the value of L1 to ecx

loop L1 ;back to L1 if ecx equal zero

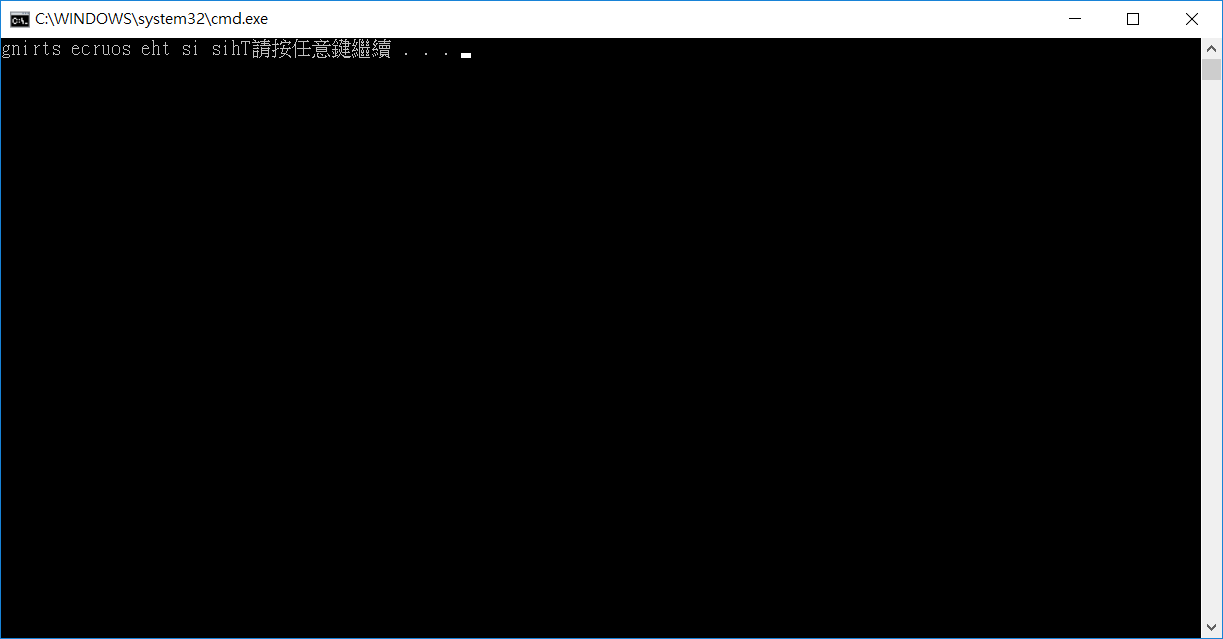
INVOKE ExitProcess,0

main ENDP

END main

1. Print your executing results 20% (貼上執行結果20%)

Problem3



2. Source Code 20% and Comments 60% (程式原始碼20%與程式註解說明50%)

INCLUDE Irvine32.inc

.386

.model flat,stdcall

.stack 4096

ExitProcess PROTO,dwExitCode:DWORD

;程式說明:

;因為每個字串結尾必須有一個空字元，所以必須先把source尾巴的空字元給target

;然後source字串從頭開始一個一個複製給dl

;把dl複製到target空字元的前一個位置，慢慢往前複製，用迴圈去處理

.data

source BYTE "This is the source string",0

target BYTE SIZEOF source DUP('#')

ttt BYTE ?

.code

main PROC

mov edx , offset target

mov eax , offset source ; eax as a pointer to string source

mov ebx , offset target ; ebx as a pointer to string target

mov ecx , SIZEOF source -1 ; the L1 counter is size of string source

add ebx , SIZEOF source -1 ; move ebx pointer to bottom of string target

add eax , SIZEOF source -1 ; move eax pointer to bottom of string source

mov dl , BYTE PTR [eax] ; move the null char of string source to dl

mov BYTE PTR [ebx] , dl ; move dl to last position of string target

sub eax , SIZEOF source-1 ; back the eax pointer to first position

dec ebx ; officially start reversing string

L1:

mov dl , BYTE PTR [eax] ; move the string source char to dl

mov BYTE PTR [ebx] , dl ; move dl to string target

inc eax ; add eax , 1

dec ebx ; sub ebx , 1

loop L1

mov edx , offset target

call WriteString

INVOKE ExitProcess,0

main ENDP

END main