```
                                ;Lab 4
                                ;Bee Cha
                                ;Declare 2 array of type DWORD with 5 elements
                                ;Perform simple arithmetic on each element in the array
                                ;Then swap and exchange the 2 array elements with each other
                                ;eax : DWORD SIZE
                                ;ax  : WORD SIZE
                                ;ah/al : BYTE SIZE
                                ;Using ECX as the accumulator

                                .586
                                .MODEL FLAT

                                INCLUDE io.h            ; header file for input/output
                               C ; IO.H -- header file for I/O macros (listing suppressed)
                               C .NOLIST      ; turn off listing
                               C .LIST        ; begin listing
                               C

                                .STACK 4096

 00000000                       .DATA

 00000000  00000005 [                  array1 DWORD  5 DUP (21H, 22H, 23H, 24H, 25H)
            00000021
            00000022
            00000023
            00000024
            00000025
           ]
 00000064  00000005 [                  array2 DWORD  5 DUP (31H, 32H, 33H, 34H, 35H)
            00000031
            00000032
            00000033
            00000034
            00000035
           ]
 000000C8 00000005              nbrElts       DWORD  5
 000000CC 41 72 72 61 79        aone   BYTE   "Array1 Contents", 0
          31 20 43 6F 6E
          74 65 6E 74 73
          00
 000000DC 41 72 72 61 79        atwo   BYTE   "Array2 Contents", 0
          32 20 43 6F 6E
          74 65 6E 74 73
          00
 000000EC 41 72 72 61 79        aone_  BYTE   "Array1 Contents + 2", 0
          31 20 43 6F 6E
          74 65 6E 74 73
          20 2B 20 32 00
 00000100 41 72 72 61 79        atwo_  BYTE   "Array2 Contents - 2", 0
          32 20 43 6F 6E
          74 65 6E 74 73
          20 2D 20 32 00
 00000114 41 72 72 61 79        aswap1 BYTE   "Array1 Swapped Contents", 0
          31 20 53 77 61
          70 70 65 64 20
          43 6F 6E 74 65
          6E 74 73 00
 0000012C 41 72 72 61 79    aswap2 BYTE   "Array2 Swapped Contents", 0
          32 20 53 77 61
          70 70 65 64 20
          43 6F 6E 74 65
          6E 74 73 00

 00000000                   .CODE
 00000000                   _MainProc PROC


                            ;----------Output the intial values of array1----------

 00000000  8D 1D 00000000 R              lea           ebx, array1                      ;get
the address of array1
 00000006  8B 0D 000000C8 R              mov           ecx, nbrElts              ;cout :=
nbrElts
                                         ;jecxz quit                             ;quit if
there are no elements
 0000000C                   forCount1:
                                         output aone, [ebx]          ;output what ebx points to
 00000021  83 C3 04                      add           ebx, 4                           ;add
4 for the next DWORD memory address
 00000024  E2 E6                         loop   forCount1                        ;repeat for
nbrElts times

                            ;---------------------------------------------------



                            ;----------Output the intial values of array2----------

 00000026  8D 1D 00000064 R              lea           ebx, array2                      ;get
the address of array2
 0000002C  8B 0D 000000C8 R              mov           ecx, nbrElts              ;cout :=
nbrElts
                            ;            jecxz  quit                             ;quit if
there are no elements
 00000032                   forCount2:
                                         output atwo, [ebx]                      ;output what
ebx points to
 00000047  83 C3 04                      add           ebx, 4                           ;add
4 for the next DWORD memory address
 0000004A  E2 E6                         loop   forCount2                        ;repeat for
nbrElts times


                            ;----------Add 2 to each element in array1----------

 0000004C  8D 1D 00000000 R              lea           ebx, array1                      ;get
the address of array1
 00000052  8B 0D 000000C8 R              mov           ecx, nbrElts              ;cout :=
nbrElts
                            ;            jecxz  quit                             ;quit if
there are no elements
 00000058                   forCount3:
 00000058  8B 03                         mov           eax, [ebx]                       ;move
contents of ebx to eax
```

```
 0000005A  83 C0 02                     add           eax, 2                           ;add
2 to eax then move it back to ebx
 0000005D  89 03                        mov           [ebx], eax
                                        output aone_, [ebx]             ;output what ebx
points to
 00000074  83 C3 04                     add           ebx, 4                           ;add
4 for the next DWORD memory address
 00000077  E2 DF                        loop   forCount3                        ;repeat for
nbrElts times

                            ;-----------------------------------------------------



                            ;----------Minus 2 to each element in array2---------

 00000079  8D 1D 00000064 R             lea           ebx, array2                      ;get
the address of array2
 0000007F  8B 0D 000000C8 R             mov           ecx, nbrElts              ;cout :=
nbrElts
                            ;             jecxz  quit                             ;quit if
there are no elements
 00000085                   forCount4:
 00000085  8B 03                        mov           eax, [ebx]                       ;move
contents of ebx to eax
 00000087  83 E8 02                     sub           eax, 2
       ;subtract 2 to eax then move it back to ebx
 0000008A  89 03                        mov           [ebx], eax
                                        output atwo_, [ebx]             ;output what ebx
points to
 000000A1  83 C3 04                     add           ebx, 4                           ;add
4 for the next DWORD memory address
 000000A4  E2 DF                        loop   forCount4                        ;repeat for
nbrElts times

                            ;------------------------------------------------------



                            ;---------Exchange/Swap array1 with array2----------

 000000A6  8D 1D 00000000 R             lea           ebx, array1                      ;get
the address of array1
 000000AC  8D 15 00000064 R             lea           edx, array2                      ;get
the address of array2
 000000B2  8B 0D 000000C8 R             mov           ecx, nbrElts              ;cout :=
nbrElts
                                        ;jecxz quit                              ;quit if
there are no elements
 000000B8                   forCount5:
 000000B8  87 DA                        xchg   ebx, edx                         ;move
contents of edx to ebx
 000000BA  83 C3 04                     add           ebx, 4                           ;add
4 for the next DWORD memory address
 000000BD  83 C2 04                     add           edx, 4
 000000C0  E2 F6                        loop   forCount5                        ;repeat for
nbrElts times

                            ;-----------------------------------------------------
```

```
                           ;----------Output the arrays with swapped conents----------

 000000C2  8D 1D 00000000 R             lea           ebx, array1                       ;get
the address of array1
 000000C8  8D 15 00000064 R             lea           edx, array2                       ;get
the address of array2
 000000CE  8B 0D 000000C8 R             mov           ecx, nbrElts
 000000D4                  forCount6:
                                        output aswap1, [ebx]             ;output what ebx
points to
 000000E9  83 C3 04                     add           ebx, 4
 000000EC  E2 E6                        loop   forCount6

 000000EE  8B 0D 000000C8 R             mov           ecx, nbrElts
 000000F4                  forCount7:
                                        output aswap2, [edx]
 00000109  83 C2 04                     add           edx, 4
 0000010C  E2 E6                        loop   forCount7

                           ;---------------------------------------------------------
 0000010E                  quit:
 0000010E  B8 00000000                   mov     eax, 0  ; exit with return code 0
 00000113  C3                    ret
 00000114                  _MainProc ENDP
                           END                            ; end of source code

Microsoft (R) Macro Assembler Version 12.00.30501.0        02/13/15 11:10:24
lab4_a.asm                                           Symbols 2 - 1




Macros:

                N a m e                 Type

atod . . . . . . . . . . . . . .  Proc
atow . . . . . . . . . . . . . .  Proc
dtoa . . . . . . . . . . . . . .  Proc
input  . . . . . . . . . . . . .  Proc
output . . . . . . . . . . . . .  Proc
wtoa . . . . . . . . . . . . . .  Proc


Segments and Groups:

                N a m e                 Size     Length   Align   Combine Class

FLAT . . . . . . . . . . . . . .  GROUP
STACK  . . . . . . . . . . . . .  32 Bit  00001000 Para        Stack         'STACK'
_DATA  . . . . . . . . . . . . .  32 Bit  00000144 Para        Public  'DATA'
_TEXT  . . . . . . . . . . . . .  32 Bit  00000114 Para        Public  'CODE'


Procedures, parameters, and locals:

                N a m e                 Type     Value    Attr
```

```
_MainProc  . . . . . . . . . . .  P Near  00000000 _TEXT      Length= 00000114 Public
  forCount1  . . . . . . . . . .  L Near  0000000C _TEXT
  forCount2  . . . . . . . . . .  L Near  00000032 _TEXT
  forCount3  . . . . . . . . . .  L Near  00000058 _TEXT
  forCount4  . . . . . . . . . .  L Near  00000085 _TEXT
  forCount5  . . . . . . . . . .  L Near  000000B8 _TEXT
  forCount6  . . . . . . . . . .  L Near  000000D4 _TEXT
  forCount7  . . . . . . . . . .  L Near  000000F4 _TEXT
  quit . . . . . . . . . . . . .  L Near  0000010E _TEXT


Symbols:

                N a m e                 Type     Value    Attr

@CodeSize  . . . . . . . . . . .  Number  00000000h
@DataSize  . . . . . . . . . . .  Number  00000000h
@Interface . . . . . . . . . . .  Number  00000000h
@Model . . . . . . . . . . . . .  Number  00000007h
@code  . . . . . . . . . . . . .  Text          _TEXT
@data  . . . . . . . . . . . . .  Text          FLAT
@fardata?  . . . . . . . . . . .  Text          FLAT
@fardata . . . . . . . . . . . .  Text          FLAT
@stack . . . . . . . . . . . . .  Text          FLAT
_getInput  . . . . . . . . . . .  L Near  00000000 FLAT       External
_showOutput  . . . . . . . . . .  L Near  00000000 FLAT       External
aone_  . . . . . . . . . . . . .  Byte    000000EC _DATA
aone . . . . . . . . . . . . . .  Byte    000000CC _DATA
array1 . . . . . . . . . . . . .  DWord   00000000 _DATA
array2 . . . . . . . . . . . . .  DWord   00000064 _DATA
aswap1 . . . . . . . . . . . . .  Byte    00000114 _DATA
aswap2 . . . . . . . . . . . . .  Byte    0000012C _DATA
atodproc . . . . . . . . . . . .  L Near  00000000 FLAT       External
atowproc . . . . . . . . . . . .  L Near  00000000 FLAT       External
atwo_  . . . . . . . . . . . . .  Byte    00000100 _DATA
atwo . . . . . . . . . . . . . .  Byte    000000DC _DATA
dtoaproc . . . . . . . . . . . .  L Near  00000000 FLAT       External
nbrElts  . . . . . . . . . . . .  DWord   000000C8 _DATA
wtoaproc . . . . . . . . . . . .  L Near  00000000 FLAT       External

          0 Warnings
          0 Errors
```