```
                        ;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
                        ;Without using ECX register or loops

                        .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
                                         output aone, [ebx]                ;output each memory
DWORD size address
                                         output aone, [ebx+4]
                                         output aone, [ebx+8]
                                         output aone, [ebx+12]
                                         output aone, [ebx+16]

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



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

 00000073  8D 1D 00000064 R              lea           ebx, array2                        ;get
the address of array2
                                         output atwo, [ebx]
                                         output atwo, [ebx+4]
                                         output atwo, [ebx+8]
                                         output atwo, [ebx+12]
                                         output atwo, [ebx+16]

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

 000000E6  8D 1D 00000000 R              lea           ebx, array1                        ;get
the address of array1
 000000EC  8B 03                         mov           eax, [ebx]                         ;copy
ebx to eax first to add
 000000EE  83 C0 02                      add           eax, 2                             ;add
2 to each element
 000000F1  89 03                         mov           [ebx], eax                         ;then
move it back to memory address
 000000F3  8B 43 04                      mov           eax, [ebx+4]                ;repeat for
each element
 000000F6  83 C0 02                      add           eax, 2
 000000F9  89 43 04                      mov           [ebx+4], eax
 000000FC  8B 43 08                      mov           eax, [ebx+8]
 000000FF  83 C0 02                      add           eax, 2
 00000102  89 43 0C                      mov           [ebx+12], eax
 00000105  8B 43 10                      mov           eax, [ebx+16]
 00000108  83 C0 02                      add           eax, 2
 0000010B  89 43 10                      mov           [ebx+16], eax

                                         output aone_, [ebx]               ;output what ebx
```

```
points to
                                        output aone_, [ebx+4]
                                        output aone_, [ebx+8]
                                        output aone_, [ebx+12]
                                        output aone_, [ebx+16]

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



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

 0000017B  8D 1D 00000064 R             lea           ebx, array2                       ;get
the address of array2

 00000181  8B 03                        mov           eax, [ebx]                        ;move
contents of ebx to eax
 00000183  83 E8 02                     sub           eax, 2
       ;subtract 2 to eax then move it back to ebx
 00000186  89 03                        mov           [ebx], eax
 00000188  8B 43 04                     mov           eax, [ebx+4]               ;repeat for
each element
 0000018B  83 E8 02                     sub           eax, 2
 0000018E  89 43 04                     mov           [ebx+4], eax
 00000191  8B 43 08                     mov           eax, [ebx+8]
 00000194  83 E8 02                     sub           eax, 2
 00000197  89 43 08                     mov           [ebx+8], eax
 0000019A  8B 43 0C                     mov           eax, [ebx+12]
 0000019D  83 E8 02                     sub           eax, 2
 000001A0  89 43 0C                     mov           [ebx+12], eax
 000001A3  8B 43 10                     mov           eax, [ebx+16]
 000001A6  83 E8 02                     sub           eax, 2
 000001A9  89 43 10                     mov           [ebx+16], eax

                                        output atwo_, [ebx]               ;output what ebx
points to
                                        output atwo_, [ebx+4]
                                        output atwo_, [ebx+8]
                                        output atwo_, [ebx+12]
                                        output atwo_, [ebx+16]

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



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

 00000219  8D 1D 00000000 R             lea           ebx, array1                       ;get
the address of array1
 0000021F  8D 15 00000064 R             lea           edx, array2                       ;get
the address of array2

                                        ;xchg  [ebx], [edx+16]                  ;move
contents of edx to ebx
 00000225  8B 43 10                     mov           eax, [ebx+16]
 00000228  87 02                        xchg   eax, [edx]
 0000022A  87 43 10                     xchg   [ebx+16], eax

 0000022D  8B 43 0C                     mov           eax, [ebx+12]
 00000230  87 42 04                     xchg   eax, [edx+4]
```

```
 00000233  87 43 0C                      xchg   [ebx+12], eax

 00000236  8B 43 08                      mov           eax, [ebx+8]
 00000239  87 42 08                      xchg   eax, [edx+8]
 0000023C  87 43 08                      xchg   [ebx+8], eax

 0000023F  8B 43 04                      mov           eax, [ebx+4]
 00000242  87 42 0C                      xchg   eax, [edx+12]
 00000245  87 43 04                      xchg   [ebx+4], eax

 00000248  8B 03                         mov           eax, [ebx]
 0000024A  87 42 10                      xchg   eax, [edx+16]
 0000024D  87 03                         xchg   [ebx], eax

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



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

 0000024F  8D 1D 00000000 R              lea           ebx, array1                       ;get
the address of array1
 00000255  8D 15 00000064 R              lea           edx, array2                       ;get
the address of array2

                                        output aswap1, [ebx]              ;output what ebx
points to
                                        output aswap1, [ebx+4]
                                        output aswap1, [ebx+8]
                                        output aswap1, [ebx+12]
                                        output aswap1, [ebx+16]

                                        output aswap2, [edx]              ;output what edx
points to
                                        output aswap2, [edx+4]
                                        output aswap2,       [edx+8]
                                        output aswap2, [edx+12]
                                        output aswap2, [edx+16]

                          ;------------------------------------------------------------
 00000335                 quit:
 00000335  B8 00000000                    mov     eax, 0  ; exit with return code 0
 0000033A  C3                     ret
 0000033B                 _MainProc ENDP
                          END                            ; end of source code

Microsoft (R) Macro Assembler Version 12.00.30501.0        02/17/15 08:28:06
lab4_b.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  0000033B Para        Public  'CODE'


Procedures, parameters, and locals:

                N a m e                 Type     Value    Attr

_MainProc  . . . . . .  P Near  00000000 _TEXT      Length= 0000033B Public
  quit . . . . . .  L Near  00000335 _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
```