```
COMMENT !

Description: Convert big endian value to little endian

Author name: Koichi Nakata

Author email: kanakta595@insite.4cd.edu

Last modified date: February 24, 2024

Creation date: February 24, 2024

!


INCLUDE Irvine32.inc


.386

.model flat, stdcall

.stack 4096

ExitProcess PROTO, dwExitCode: dword


.data

bigEndian word 12h, 34h, 56h, 78h

littleEndian dword ?


.code

main PROC

        mov ah, bigEndian

        mov al, bigEndian + 1

        mov word ptr littleEndian + 2, ax


        mov ah, byte ptr bigEndian + 2

        mov al, byte ptr bigEndian + 3

        mov word ptr littleEndian, ax
        CALL DumpRegs


        INVOKE ExitProcess, 0
main endp
end main
```

```
COMMENT !

Description: Exchanges the upper and lowers words in a double word variable

Author name: Koichi Nakata

Author email: kanakta595@insite.4cd.edu

Last modified date: February 25, 2024

Creation date: February 25, 2024

!


INCLUDE Irvine32.inc


.386

.model flat, stdcall

.stack 4096

ExitProcess PROTO, dwExitCode: dword


.data

three dword 12345678h                          ; This is stored as 78, 56, 34, 12 in the memory

temp word 2 DUP(?)


.code

main PROC


        mov ax, word ptr three          ; ax = 5678h (in memory, 78, 56)

        mov bx, word ptr three + 2      ; bx = 1234h

        mov temp, bx                    ; temp = {1234h, ?}

        mov temp + 2, ax                ; temp = {1234h, 5678h}


        mov eax, dword ptr temp         ; The first word is copied to the lower of eax, and the
second word is copied to the upper of eax, so eax = 56781234hs
```

```
        mov three, eax                          ; Move back the value to three


        CALL DumpRegs


        INVOKE ExitProcess, 0
main endp
end main
```

```
COMMENT !
Description: Calculates the first seven values of the Fibonacci Sequence, using a loop
Author name: Koichi Nakata
Author email: kanakta595@insite.4cd.edu
Last modified date: February 25, 2024
Creation date: February 25, 2024
!

INCLUDE Irvine32.inc

.386
.model flat, stdcall
.stack 4096
ExitProcess PROTO, dwExitCode: dword

.data
fibonacci word 7 DUP(0)                         ; Create an array with 7 zeros

.code
main PROC
        mov fibonacci, 1                        ; Fill 1 in the first element
        mov fibonacci + 2, 1                ; Fill 1 in the second element

        mov ecx, 5                                  ; We want to loop 5 times
        mov esi, offset fibonacci + 4   ; Make an iterator starting from the third element

L1:
        mov ax, [esi - 4]                   ; Move the previous element to ax
        add ax, [esi - 2]                   ; Add the previous previous element to ax
```

```
	mov [esi], ax				; [esi] deferences the third element
	add esi, 2					; Don't forget increment the iterator by
2 bytes
	loop L1

	CALL DumpRegs

	INVOKE ExitProcess, 0
main endp
end main
```

```
COMMENT !

Description: Reorders the values in four 8-bit registers, using xchg no more than 3 times

Author name: Koichi Nakata

Author email: kanakta595@insite.4cd.edu

Last modified date: February 26, 2024

Creation date: February 26, 2024

!


INCLUDE Irvine32.inc


.386

.model flat, stdcall

.stack 4096

ExitProcess PROTO, dwExitCode: dword


.data


.code

main PROC

        mov al, "A"

        mov bl, "B"

        mov cl, "C"

        mov dl, "D"             ; Now (al, bl, cl, dl) = {A, B, C, D}


        xchg al, bl             ; (al, bl, cl, dl) = {B, A, C, D}

        xchg bl, cl             ; (al, bl, cl, dl) = {B, C, A, D}

        xchg cl, dl             ; (al, bl, cl, dl) = {B, C, D, A}


        CALL DumpRegs
```

```
	INVOKE ExitProcess, 0
main endp
end main
```

```
COMMENT !

Description: Reverses an array with any data type and size, using a loop

Author name: Koichi Nakata

Author email: kanakta595@insite.4cd.edu

Last modified date: February 25, 2024

Creation date: February 25, 2024

!


INCLUDE Irvine32.inc


.386

.model flat, stdcall

.stack 4096

ExitProcess PROTO, dwExitCode: dword


.data

array dword 1, 5, 6, 8, 0Ah, 1Eh, 22h, 2Ah, 32h


.code

main PROC

        mov esi, 0                                          ; Index of the first element

        mov edi, sizeof array - type array       ; Index of the last element (use sizeof, not lengthof)

        mov ecx, lengthof array / 2                   ; Counter


L1:

        mov eax, array[esi]

        xchg eax, array[edi]

        mov array[esi], eax
        add esi, type array                                  ; Increment the first index by type bytes
        sub edi, type array                                  ; Decrement the second index by type
bytes
        loop L1



        CALL DumpRegs


        INVOKE ExitProcess, 0
main endp
end mainp
```