1. mov

```
.data
var1 BYTE  100
var2 BYTE  ?
var3 BYTE  2
var4 BYTE  1

.code          ; True/False
 mov ds, 45  ;
 mov esi, var3  ;
 mov eip, var4  ;
 mov 25, var2  ;
 mov var1, var2  ;
```

```
.data
bVal   BYTE    100
bVal2  BYTE    ?
wVal   WORD    2
dVal   DWORD   5
.code
    mov ds,45        immediate move to DS not permitted
    mov esi,wVal     size mismatch
    mov eip,dVal     EIP cannot be the destination
    mov 25,bVal      immediate value cannot be destination
    mov bVal2,bVal   memory-to-memory move not permitted
```

2. xchg

Write a program that rearranges the values of three doubleword values in the following array as: 3, 1, 2.

Definition:

```
.data
 array DWORD 1, 2, 3
```

Step1: copy the first value into E A X and exchange it with the value in the second position.

```
mov eax,arrayD
xchg eax,[arrayD+4]
```

Step 2: Exchange E A X with the third array value and copy the value in E A X to the first array position.

```
xchg eax,[arrayD+8]
mov  arrayD,eax
```

3. INC/DEC

```
.data
        myByte BYTE 0FFh, 0
.code
        mov al,myByte       ; AL =
        mov ah,[myByte+1]           ; AH =
        dec ah      ; AH =
        inc al      ; AL =
        dec ax      ; AX =
```

FFh

00h

FFh

00h

FEFF

4. flag

```
mov al,-128
neg al      ; CF =1     OF =1
```

The carry flag, on a subtraction, represents a borrow. If you negate x, you (virtually) subtract x from 0, which needs a borrow, unless x is 0.

```
mov ax,8000h
add ax,2   ; CF =0    OF =0

mov ax,0
sub ax,2   ; CF =1    OF =0

mov al,-5
sub al,+125            ; OF =1
```

5. PTR

```
 .data
varB BYTE 65h,31h,02h,05h
varW WORD 6543h,1202h
varD DWORD 12345678h

.code
```

```
mov ax,WORD PTR [varB+2]        ; a.
mov bl,BYTE PTR varD     ; b.
mov bl,BYTE PTR [varW+2]; c.
mov ax,WORD PTR [varD+2]        ; d.
mov eax,DWORD PTR varW; e.
```

0502h

78h

02h

1234h

12026543h

6. LOOP

What will be the final value of AX? 10

```
        mov ax,6
        mov ecx,4
L1:
        inc ax
        loop L1
```

7. OFFSET

Please finish the program below for an array sum.

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

```
.data
arrayW WORD 1000h,2000h,3000h
.code
mov esi,OFFSET arrayW
mov ax,[esi]
add esi,2     ; or: add esi,TYPE arrayW
add ax,[esi]
add esi,2
add ax,[esi]  ; AX = sum of the array
```