a db 7
dw, dd, dq

a resb 2 => 2 bytes uninitialised

three equ 3 => declaring constants

mov<dest>,<source>
//// (memory, resisters) , (memory, registers, constant immediate) ///
move EAX, 1 ; (=> EAX = 1) destination and source have to have the same size
mov EAX, BL => ERROR (not the same size)

mov EAX, [a] => it takes the size of EAX, in this case a doubleword => Logical error, doesn't crash, but it places [a]'s value in AL.
FIX: mov AL, byte [a]

When we write [a] => address of a, so it doesn't know the size of it, we SHOULD SPECIFY THE SIZE.

WE CANNOT HAVE TWO OPPERANDS THAT ARE ADDRESES like: mov [a], [b] => ERROR
But we can do:
    mov bl, ah

eax => memory location, like the pointer
[eax] => accesses the value of eax

INSTRUCTIONS:
ADD, SUB
Same rules apply as MOV

ADD [b], AL => add the value stored in AL to the value stored in b
ADD [a], [b] => BOTH MEMORY LOCATION => ERROR

MUL
1byte * 1 byte => 2bytes
2bytes * 2bytes => 4bytes
4bytes * 4bytes => 8bytes

MUL <explicit opperand>
MUL BL => BL * AL = AX
MUL CX(2 bytes) =>CX * AX = DX : AX (16bits : 16bits stored in each)
MUL ECX => ECX * EAX = EDX : EAX

ALWAYS SPECIFY THE SIZE IN MULTIPLICATION
mov al, 2
mul byte [b]
=> ax = 6

mov bx, 2
mov ax, 100h
mul bx
=> bx = 2 and dx = 00 00 and ax = 02 00

mov ax, 5
mul 2

=> ERROR, cannot put immediates in mul

DIV
2bytes/1byte => 1byte result, remainder 1byte
4bytes/2bytes => 2bytes result, remainder 2 bytes
8/4 => 4 r 4

DIV BL(1 byte) => AX/BL => result AL remainder AH
DIV CX (2 bytes)=> DX: AX / CX => res AX remainder DX
DIV ECX(4 bytes) => EDX: EAX / ECX => res EAX remainder EDX

EXAMPLES:

```
mov ax, 10
mov bl, 2
div bl
```
=> al = 5 , ah = 0

```
mov ax, 200h
mov bl, 2
div bl
```
=> division error 100h bigger than al
FIX:
```
mov dx, 0
mov ax, 200h
```
=> YOU HAVE TO PUT BOTH IN DX AND AX because that's your divident
```
mov bl, 2
div bx

mov ax, 10
mov bl, 0
```

div bl
=> division error
1111 => 4 bits => baza 16 pe 4 bits
FF => 15 + 16*15 = 15 * 17 = 255 max in baza 10 reprezentat pe AL, maxim FF reprezentat baza 16 pe AL
200h => 2*16^2 = 512

INC/DEC/NEG
INC EAX; => EAX ++
INC [a] => ERROR => INC byte[a] => correct, specify the size

DEC EAX; => EAX--
DEC byte[a]

NEG EAX; EAX = -EAX(complement by 2)
NEG byte [a]

GENERAL EXERCISES:
9. (2 * d + e)/a
data
a db 5
d db 4
e db 2

code:
start:
mov al, [d]
mov bl, 2
mul bl; => ax = 2*d
add ax, [e]
div byte [a]