Course

Location counter and pointer arithmetic

```
segment data
a db 1,2,3,4 ; four bytes generated, correct 01 02 03 04
lg db $-a ; = 4
// lg0 db $-data ; in nasm is a SyntaxError
// lg1 db a-data ; same thing, SyntaxError
// lg2 dw data-a ; ok syntactically talking, LinkingError
  db a-$ ; = -5 in base 10 = FB
c equ a-$ ; correct, -6 = FA
d equ a-$ ; correct, -6 = FA, they are constants, not defining any bytes (doesn't have db)
e db a-$ ; correct, -6 = FA even though it has a db, next one would've been -7
x dw x ; correct -> 07 00 (should be)
                 -> 07 10 (if i put it in ollydebugger)
I want x to be initialised with the offset of itself,
x1 db x1 ; wrong, syntax error because on 8 bits you can't represent an address
x2 dw x2 ; correct -> 09 00
                   -> 09 10
(address is the number of bytes)
  db lg-a ; correct, scalar = 4, in memory 04
  db a-lg ; = -4 = FC
  db [$-a] ; SyntaxError, $-a is a scalar
; you can't have in [] a register or a memory operator
; in assembly time both of them can't be calculated
; you can only put constants
  db [lg-a] ; same thing
lg11 EQU lg11 ; it works? but it shouldn't, you want to define a thing by itself, blasfemie
; you obtain 0 apparently, BUG NASM!!
lg12 EQU lg12-a ; it works? again 0?? because a is 0 as an offset by nasm at assembly time,
; it would've been a syntax error if a wasn't 0 at assembly time, like having something befora a
g12 dw c-2 ; = -8 = FFF8

b dd a-start ; SyntaxError, they are from two different segments
; a is defined in the data segment, and you want to subtract something somewhere else
; here-somewhere else - always a SyntaxError, but not the other way around
  dd start-a ; it works,
; somewhere else - here - OK!!, but it is not a scalar, it's a pointer data type
  dd start-start1 ; test test 123

segment code
start:
  mov ah, lg11 ; = 0
  mov bh, c ; = -6 = FA
  mov ch, lg ;  SyntaxError, lg without square brackets is an address, the offset, and it doesn't fit in a byte
  mov ch, lg - a ; SyntaxError, same
  mov ch, [lg - a] ; wrong, at runtime it works, (what happens in the code segment), it is a correct formula, but you get a memory vio
  mov cx, lg - a ; CX = 0004
  mov cx, [lg-a] ; idk nu eram atent, nu cred ca era bun
  mov cx, $-a ; here - somewhere else. syntax error
  mov cx, $$-a ; $$ - starting address of the current section, still syntax error
  mov cx, a - $ ; it's okay, somewhere else - here
  mov ch, $-a ; Syntax Error, invalid operand type
  mov ch, a - $ ; the exprssion is ok, but it is a SyntaxError because it is a pointer type and can't fit in a byte
  mov cx, $-start ; it's okay, here-here
  mov cx, start - $ ; okay
  mov ch, $-start ; okay, it's a scalar
  mov ch, start - $ ; okay, it's a scalar  not a pointer
  mov cx, a-start ; okay, somewhere else - here
  mov cx, start-a  ; SyntaxError, here - somewhere else




start1: ; we suppose b is good
  mov ah, a+b ; will accept espression, a+b = (a-$$) + (b-$$). Addition of scalars
  mov ax, b+a ; same thing
  mov ax,[a+b] ; SyntaxError
  var1 dd a+b ; cred????? nu eram atent SyntaxError
```