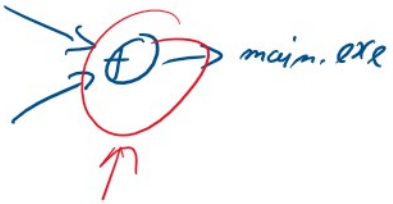
Seminar VI 212

Tuesday, December 8, 2020 12:04 PM

modul1.asm → modul1.obj

:

modulN.asm → modulN.obj



nasm -fobj modul1.asm

alink -oPE -subsys console -entry start modul1.obj ... modulN.obj

global, import

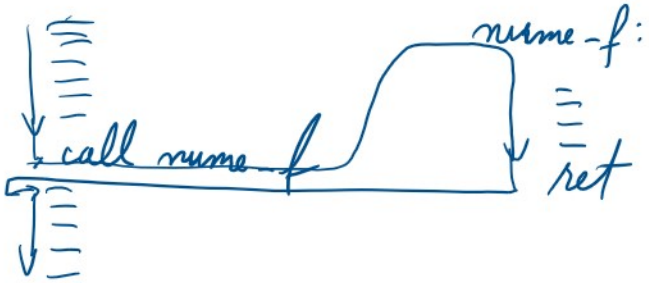
global = export un simbol

import =

---

call nume_f

ret



segment code

nume_f:

=

=

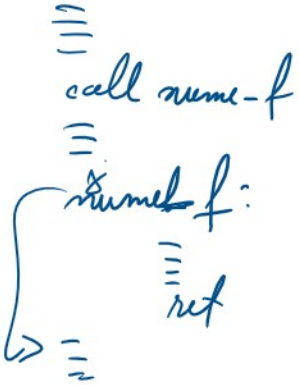
ret

stat:

↓

.eticheta:

eticheta:



---

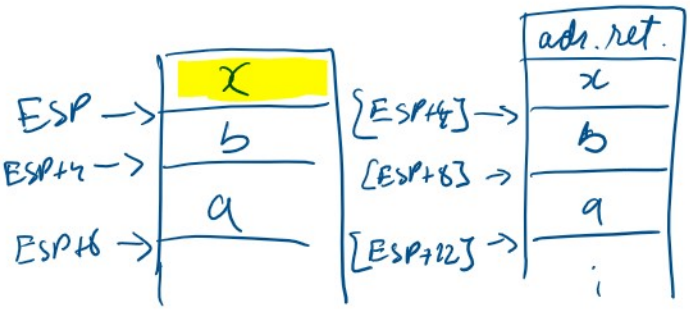
pasare param.

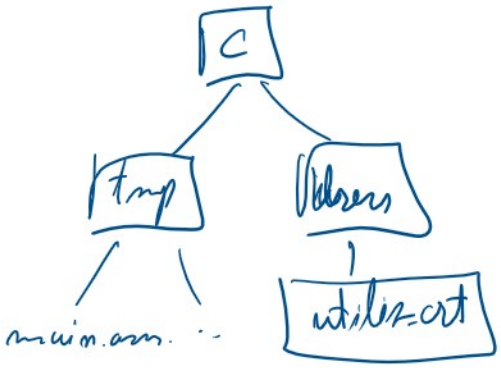
↳ stack

↳ registrii

↳ var. globale

$x = a + b$





---

hex [0, 0xFFFF]

in.txt 16 octeți

ex. F2 A1

in.txt 0 1 2 3 4 5 6 7 8 9 A B C D E F

1 0 0 0 0 1 0 1 0 1 0 0 1 1 1 1 F2A1

0 5 7 9 C D E F

$(F2A1)_{16}$

1111 0010 1010 0001

1.asm

```
bits 32

global start
extern f_afisare

extern exit, scanf, fopen, fread, fclose, printf
import exit msvcrt.dll
import printf msvcrt.dll
import scanf msvcrt.dll
import fopen msvcrt.dll
import fread msvcrt.dll
import fclose msvcrt.dll

segment data use32 class=data
   format_h    db  '%x',0
   numar       dd  0
   fisier      db  'in.txt',0
   mod_r       db  'r',0
   handle_r    dd  -1
```

```
    buffer     resb    16

segment code use32 class=code
start:
    ;citesc numarul in reprezentare hexa
    ;int scanf(const char * format, variabila_1, constanta_2, ...);
    push numar
    push format_h
    call [scanf]
    add esp, 8

    ;FILE * fopen(const char* nume_fisier, const char * mod_acces)
    push mod_r
    push fisier
    call [fopen]
    add esp,8
    mov [handle_r], eax

    ;int fread(void * buffer, int size, int count, FILE * stream)
    push dword [handle_r]
    push dword 16
    push dword 1
    push buffer
    call [fread]
    add esp, 16

    ;int fclose(FILE * descriptor)
    push dword [handle_r]
    call [fclose]
    add esp, 4

    ;pun param
    push dword [numar]
    push buffer
    call f_afisare


    push    dword 0
    call    [exit]
```

2.asm

```
bits 32

global f_afisare

extern printf
import printf msvcrt.dll

segment data use32 class=data

   format_c    db  '%c',0

segment code use32 class=code
f_afisare:
   mov ebx, [esp+8]
   mov ecx, 16
   mov esi, [esp+4]
   cld
   repeta:
      lodsb
      rcr bx, 1
      jnc mai_departe
         ;afisez caracterul din registrul al
         pushad
         mov edx, 0
         mov dl, al
         push edx
         push format_c
         call [printf]
         add esp, 8


         popad
      mai_departe:
      loop repeta

   ret
```