Programare multi-modul în limbajul de asamblare

* Avem mai multe fisiere sursa cu extensia .asm care se vor asambla separat  
  nasm –fobj modul1.asm  
  nasm –fobj modul2.asm  
  …  
  nasm –fobj modulN.asm
* Se va crea un singur fisier executabil, prin link-editarea fisierelor cu extensia .obj care au rezultat in urma asamblarii:  
  alink –oPE –subsys console –entry start modul1.obj modul2.obj … modulN.obj
* Va rezulta fisierul executabil modul1.exe
* Important!:
  + Doar un modul va contine un program principal
  + Celelalte modul vor descrie functii/proceduri care vor fi apelate de programul principal
* Folosind cuvantul cheie **global** se poate exporta o variabila, o procedura care este definita in modulul curent pentru a putea fi utilizat in celelalte module.
* Celelalte module vor importa variabila sau procedura folosind cuvantul cheie **import**

Transmiterea parametrilor unei functii / proceduri

1. Folosind registrii

* Dezavantaj: exista un numar limitat de registrii care pot fi utilizati

1. Folosind variabile

* Se vor declara variabilele care se doreste a fi transmise ca parametri ca fiind **global**
* Dezavantaj: se incalca principiul modularitatii (un program este mai bine intretinut daca este compus din module INDEPENDENTE)

1. Folosind stiva

* Este solutia preferata datorita flexibilitatii

Exemple: Sa se rezolva x=a+b

1. Folosind registrii pentru transmiterea parametrilor

|  |  |
| --- | --- |
| MAIN.asm (modulul principal) | FUNCTION.asm (modulul secundar) |
| bits32 global start extern exit import exit msvcrt.dll  **extern adunare**  segment data use32 class=data **public** a db 2 b db 3 x db 0  segment code use32 class=code **public** start:  mov BL,[a] mov BH,[b] **call adunare**  mov [x],al  push dword 0 call [exit] | bits32 **global adunare**  segment code use32 class=code **public**  adunare: mov al,bl  add al,bh ret |

1. Folosind variabile pentru transmiterea parametrilor

|  |  |
| --- | --- |
| MAIN.asm (modulul principal) | FUNCTION.asm (modulul secundar) |
| bits32 global start extern exit import exit msvcrt.dll  **extern adunare**  **global a** **global b** **global x**  segment data use32 class=data **public** a db 2 b db 3 x db 0  segment code use32 class=code **public** start:  **call adunare**  push dword 0 call [exit] | bits32 **global adunare**  **extern a,b,x**  segment code use32 class=code **public**  adunare: mov al,[a] add al,[b] mov [x],al ret |

1. Folosind stiva pentru transmiterea parametrilor

|  |  |
| --- | --- |
| MAIN.asm (modulul principal) | FUNCTION.asm (modulul secundar) |
| bits32 global start extern exit import exit msvcrt.dll  **extern adunare**  segment data use32 class=data **public** a db 2 b db 3 x db 0  segment code use32 class=code **public** start:  **mov eax,0** **mov al,[a]** **push eax**  **mov eax,0** **mov al,[b]** **push eax**  **mov eax,0** **mov al,[x]** **push eax**  **call adunare**  **pop eax**  **mov [x],al**  **add esp,4\*2**  push dword 0 call [exit] | bits32 **global adunare**  segment code use32 class=code **public**  adunare:   |  |  | | --- | --- | | STIVA | | | Adresa de revenire | [ESP] | | x | [ESP+4] | | b | [ESP+8] | | a | [ESP+12] |   **mov EAX, [ESP+12]; EAX=[a]** **add EAX, [ESP+8]; EAX=[a]+[b]**  **mov [ESP+4],EAX** ret |

Probleme:

1. Scrieti un program care concateneaza doua siruri de octeti prin apelul unei functii scrise intr-un modul secundar si apoi afiseaza rezultatul pe ecran.

|  |  |
| --- | --- |
| MAIN.asm | MODUL.asm |
| bits32 global start import exit msvcrt.dll  Import printf msvcrt.dll  extern exit  extern printf  extern concatenare  segment data use32 class=data public  sir1 db “acest sir”,0  sir2 db “ este lung”,0  rez times $-sir1-1 db 0  segment code use32 class=code public  start:  push dword sir1  push dword sir2  push dword rez  call concatenare  add esp,4\*3  Push rez  Call [printf]  add esp,4  push dword 0 call [exit] | bits32 global concatenare  segment code use32 class=code public  concatenare:   |  |  | | --- | --- | | STIVA | | | Adresa de revenire | [ESP] | | rez | [ESP+4] | | sir2 | [ESP+8] | | sir1 | [ESP+12] |   mov esi,[ESP+12]  mov edi,[ESP+4]  Repeta:  Lodsb  stosb  Cmp al,0  Jnz Repeta  Dec edi  mov esi,[ESP+8]  Repeta2:  Lodsb  stosb  Cmp al,0  Jnz Repeta2  ret |

1. Se da un numar a reprezentat pe 32 de biti fara semn. Sa se afiseze suma cifrelor numarului

|  |  |
| --- | --- |
| MAIN.asm | MODUL.asm |
| bits32 global start extern exit import exit msvcrt.dll  Import printf msvcrt.dll  Extern printf  Extern sumacifrelor  segment data use32 class=data public  a dd 1234  rez dd 0  Format\_afisare db “Suma cifrelor este %d”,0 segment code use32 class=code public  start:  Push dword [a]  Push dword rez  Call sumacifrelor  Add esp,4\*2  Push dword [rez]  Push dword Format\_afisare  Call [printf]  Add esp,4\*2  push dword 0 call [exit] | bits32 global sumacifrelor  segment code use32 class=code public  adunare:   |  |  | | --- | --- | | STIVA | | | Adresa de revenire | [ESP] | | Rez | [esp+4] | | A | [esp+8] |   sumacifrelor:  Mov eax,[esp+8]  Mov esi,[esp+4]  Mov edx,0  Mov ebx,10  Repeta:  Div ebx;edx-restul  Add [esi],edx  Mov edx,0  Cmp eax,0  Jne Repeta  ret |
|  |  |

1. Se citeste un cuvant de la tastatura. Sa se cripteze cuvantul citit adunand 20 la fiecare caracter. Sa se afiseze pe ecran rezultatul

|  |  |
| --- | --- |
| MAIN.asm | MODUL.asm |
| bits32 global start extern exit import exit msvcrt.dll  Import printf msvcrt.dll  Import scanf msvcrt.dll  Extern printf,scanf,criptare  segment data use32 class=data public  Sir times 100 db 0  Format db “%s”,0  Rez times 100 db 0 segment code use32 class=code public start:  Push dword sir  Push dword Format  Call [scanf]  Add esp,4\*2  Push dword sir  Push dword rez  Call criptare  Add esp,4\*2  Push dword rez  Call [printf]  Add esp,4  Push dword 0  Call [exit] | bits32  Global criptare  segment code use32 class=code public  criptare:   |  |  | | --- | --- | | STIVA | | | Adresare de revenire [esp+0] | | | Rez [esp+4] | | | Sir [esp+8] | |   criptare:  Mov esi,[esp+8]  Mov edi,[esp+4]  CLD;DF=0  Repeta:  Lodsb  ;al=caracter  Cmp al,0  Je end  Add al,20  Stosb  Jmp Repeta  end:  ret |
|  |  |

Tema: Se dau 2 siruri de caractere. Sa se afiseze sirul care contine numarul maxim de caractere speciale (caracter special = orice caracter in afara de litera sau cifra)

3 module: main, modulul care calculeaza maximul dintre 2 numere, modulu care calculeaza numarul de caractere speciale dintr-un sir

Cum arata stiva dupa ce se executa urmatoarea secventa de cod:

Push dword [a]

Push dword [b]

Push eax

Push dword [c]

Push dword [d]

Call modul

De la ce adrese se pot prelua valorile variabilelor a,b,c,d in modul?