Seminar 6 – Programarea multi modul

* Exista mai multe fisiere 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 care va fi compus din mai multe fisiere cu extensia .obj, prin link-editare  
  alink –oPE –subsys console –entry start modul1.obj modul2.obj … modulN.obj
* Va rezulta modul1.exe
* Observatii
  + Un singur modul poate contine programul principal
  + Celelalte module vor descrie proceduri care vor fi apelate din programul principal
  + Folosind **global** se poate exporta un simbol (variabila, procedura) definit in modulul curent cu scopul de afi folosit in alte module
  + Folosind **extern** pentu a importa simbolul extern

Transmiterea parametrilor unei proceduri

1. Prin intermediul registrilor (numarul de registrii este limitat)
2. Prin intermediul variabilelor globale (variabilele vor fi vizibile in toate modulele => modulele nu o sa fie INDEPENDENTE)
3. Prin intermediul stivei (solutia preferata!)

Exemple:

Scrieti un program in limbaj de asamblare care rezolva expresia x=a+b prin intermediul unui modul care primeste la intrare 2 numere si furnizeaza la iesire suma lor.

1. Transmiterea parametrilor prin intermediul registrilor

|  |  |
| --- | --- |
| MAIN.ASM | MODUL.ASM |
| bits 32  global start    extern exit import exit msvcrt.dll  **extern suma**  segment data use32 class=data **public**  A db 12h  B db 34  X resb 1  segment code use32 class=code **public**  start:  Mov AL, [A]  Mov AH, [B]  **CALL suma**  Mov [X],AL  push dword 0  call [exit] | bits 32  **global suma**  segment code use32 class=code **public**  suma:  ADD AL, AH  ret |

1. Transmiterea parametrilor prin intermediul variabilelor

|  |  |
| --- | --- |
| MAIN.ASM | MODUL.ASM |
| bits 32  global start    extern exit import exit msvcrt.dll  **extern suma**  **global A** **global B** **global X**  segment data use32 class=data **public**  A db 12h  B db 34  X resb 1  segment code use32 class=code **public**  start:  **CALL suma**  push dword 0  call [exit] | bits 32  **global suma** **extern A, B, X**  segment code use32 class=code **public**  suma:  MOV AL, [A] ADD AL, [B}  MOV [X], AL  ret |

1. Transmiterea parametrilor prin intermediul stivei

|  |  |
| --- | --- |
| MAIN.ASM | MODUL.ASM |
| bits 32  global start    extern exit import exit msvcrt.dll  **extern suma**  segment data use32 class=data **public**  A db 12h  B db 34  X resb 1  segment code use32 class=code **public**  start:  ; suma(A,B,X)  PUSH dword X  MOV EAX,0 MOV AL,[B] PUSH EAX  MOV EAX,0 MOV AL,[A] PUSH EAX  **CALL suma** ADD ESP, 4\*3  push dword 0  call [exit] | bits 32  **global suma**  segment code use32 class=code **public**  suma:  ; stiva  ; [ESP] <- adresa de revenire ...  ; [ESP+4] <- valoarea lui A …. 00000012h ; [ESP+8] <- valoarea lui B …. 00000022h  ; [ESP+12] <- adresa lui X  MOV EAX, [ESP+4] ADD EAX, [ESP+8]  MOV ESI, [ESP+12]  MOV [ESI], EAX  ret |

Probleme

1. Scrieti un program multi modul in limbaj de asamblare care concateneaza doua siruri prin apelul unei proceduri scrise intr-un modul secundar si care apoi afiseaza rezultatul pe ecran

|  |  |
| --- | --- |
| MAIN.ASM | MODUL.ASM |
| bits 32  global start    extern exit  Extern printf import exit msvcrt.dll  Import printf msvcrt.dll  **extern \_proc**  segment data use32 class=data **public**  A db ‘abcde’  Dim\_a equ $-A  B db ‘efghijkl’  Dim\_b equ $-B  C times dim\_a + dim\_b+1 db 0  Format db ‘%s’, 0  segment code use32 class=code **public**  start:  ; proc(c, a, b, dim\_a, dim\_b)  **Push dword dim\_b**  **Push dword dim\_a**  **Push dword b**  **Push dword a**  **Push dword c**  **Call \_proc**  **Add esp, 4\*5**  **;printf(format, c)**  **Push dword c**  **Push dword format**  **Call [printf]**  **Add esp, 4\*2**  push dword 0  call [exit] | bits 32  **global \_proc**  segment code use32 class=code **public**  \_proc:  ; stiva  ; [ESP] <- adresa de revenire ...  ; [ESP+4] <- offset c  ; [ESP+8] <- offset a  ; [ESP+12] <- offset b  ; [ESP+16] <- dim\_a  ; [ESP+20] <- dim\_b  Cld ; DF = 0  Mov esi, [esp+8]  Mov edi, [esp+4]  Mov ecx, [esp+16]  Jecxz final  Repeta:  movsb  Loop repeta  Final:  Mov esi, [esp+12]  Mov ecx, [esp+20]  Jecxz final2  Rep Movsb  Final2:  ret |
|  |  |

1. Se da un sir de numere reprezentate pe 32 de biti cu semn. Sa se afiseze numarul care are cea mai mare suma a cifrelor.

|  |  |
| --- | --- |
| MAIN.ASM | MODUL.ASM |
| bits 32  global start    extern exit  extern \_suma\_cifre  extern \_max  extern printf import exit msvcrt.dll  Import printf msvcrt.dll  segment data use32 class=data **public**  S dd 123, 421, 642  Len\_s EQU $-S  Suma db 0  M db 0  El dd 0  Format db “%d”, 0  segment code use32 class=code **public**  start:  cld  Mov esi,S  Mov ecx, Len\_s  Jecxz final  Bucla:  Lodsd  Push ecx  Push eax  ;\_suma\_cifre(s,suma)  Push dword Suma  Push dword eax  Call \_suma\_cifre  Add esp, 4\*2  ;[Suma] = suma cifrelor dublucuvantului curent  ;\_max(a,b,m)  Push dword M  Push dword [M]  Push dword [suma]  Call \_max  Add esp, 4\*3  Pop eax  Pop ecx    Mov edx, [M]  CMP edx, [suma]  JNE next  Mov [el], eax  Next:  Loop bucla  ; printf(format, el)  Push dword [el]  Push format  Call [printf]  Add esp, 4\*2  Final:  push dword 0  call [exit] | bits 32  **global \_suma\_cifre**  segment code use32 class=code **public**  \_suma\_cifre:  ; stiva  ; [ESP] <- adresa de revenire  ; [ESP+4] <- dublucuvantul curent  ; [ESP+8] <- adresa lui suma  Mov esi,[esp+8]  Mov ecx,0; se calculeaza suma    Mov eax, [esp+4]  Repeta:  CDQ  Mov ebx, 10  Idiv ebx ; eax = catul edx = restul  Add ecx, edx  CMP eax, 0  JNE Repeta  Mov [esi],ecx  ret |
| MAX.ASM |  |
| bits 32  global \_max  segment code use32 class=code **public**  \_max:  ; stiva  ; [ESP] <- adresa de revenire  ; [ESP+4] <- valoare suma  ; [ESP+8] <- valoare M  ; [ESP+12] <- adresa lui M  Mov esi, [ESP+12]  Mov eax, [ESP+4]  Mov ebx, [ESP+8]  CMP eax, ebx  JG Caz1  Mov esi, ebx  Jmp Final  Caz1:  Mov [esi], eax  Final:  ret |  |

1. Sa se scrie un program multi modul in limbaj de asamblare in care se citeste un cuvant (max 20 de caractere) de la tastatura. Sa se formeze criptarea acestui cuvant prin adaugarea dupa fiecare vocala mica a literei “p” urmata de vocala.

Ex: aluna=> apalupunapa

|  |  |
| --- | --- |
| MAIN.ASM | MODUL.ASM |
| bits 32  global start    extern exit import exit msvcrt.dll  Extern scanf  Import scanf msvcrt.dll  Extern printf  Import printf msvcrt.dll  **extern ?**  segment data use32 class=data **public**  Cuvant TIMES 21 DB 0  Result TIMES 61 DB 0  Format DB “%s”, 0  segment code use32 class=code **public**  start:  Push dword Cuvant  Push dword format  Call [scanf]  Add esp, 4\*2  ; criptare(cuvant,dest)  Push dword Result  Push dword Cuvant  Call Criptare  Add esp, 4\*2  Push dword result  Push dword format  Call [printf]  Add esp, 4\*2  push dword 0  call [exit] | bits 32  **global ?**  segment code use32 class=code **public**  Criptare:  ; cuv = [ESP+4]  ; res = [ESP+8]  Mov ESI, [ESP+4]  Mov EDI, [ESP+8]  .iterate  Lodsb ; AL = [ESI], ESI++  Cmp AL, 0; Or AL, AL ; AL != 0  Jz, .exit  Stosb  Cmp AL, ‘a’  Jz .execute  Cmp AL, ‘e’  Jz .execute  Cmp AL, ‘i’  Jz .execute  Cmp AL, ‘o’  Jz .execute  Cmp AL, ‘u’  Jz .execute  Jmp .skip  .execute  Mov BL, AL  Mov AL, ‘p’  Stosb  Mov AL, BL  Stosb  .skip:  Jmp .iterate  .exit:  ret |
|  |  |

1. Sa se scrie un program in limbaj de asamblare care afiseaza pe ecran cuvantul criptat (cf. 3) de lungime maxima dintr-un sir de n cuvinte citite de la tastatura (n se da)

|  |  |
| --- | --- |
| MAIN.ASM | MODUL.ASM |
| bits 32  global start    extern exit import exit msvcrt.dll  Extern scanf  Import scanf msvcrt.dll  Extern printf  Import printf msvcrt.dll  **extern ?**  segment data use32 class=data **public**  Cuvant TIMES 21 DB 0  Result\_max TIMES 21 DB 0  Result TIMES 61 DB 0  Format DB “%s”, 0  N db 9  Lungime db 0  Maxim db 0  segment code use32 class=code **public**  start:  Mov ecx, [n]  Jecxz end  Repeta:  Push ecx  Push dword Cuvant  Push dword format  Call [scanf]  Add esp, 4\*2  ; criptare(cuvant,dest)  Push dword Result  Push dword Cuvant  Call Criptare  Add esp, 4\*2    ;lungime(Cuvant, lungime)  Push Lungime  Push Cuvant  Call \_lungime  Add esp 4\*2   ;TEMA  ;comparare lungime cu max si retinere cuvant de lungime max pentru afisare  Pop ecx  Loop repeta  ;afisare cuvant de lungime max  Push dword result\_max  Push dword format  Call [printf]  Add esp, 4\*2  end:  push dword 0  call [exit] | bits 32  **global ?**  segment code use32 class=code **public**  Criptare:  ; cuv = [ESP+4]  ; res = [ESP+8]  Mov ESI, [ESP+4]  Mov EDI, [ESP+8]  .iterate  Lodsb ; AL = [ESI], ESI++  Cmp AL, 0; Or AL, AL ; AL != 0  Jz, .exit  Stosb  Cmp AL, ‘a’  Jz .execute  Cmp AL, ‘e’  Jz .execute  Cmp AL, ‘i’  Jz .execute  Cmp AL, ‘o’  Jz .execute  Cmp AL, ‘u’  Jz .execute  Jmp .skip  .execute  Mov BL, AL  Mov AL, ‘p’  Stosb  Mov AL, BL  Stosb  .skip:  Jmp .iterate  .exit:  ret |
| MAX.ASM | LUNGIME.ASM |
| bits 32  global \_max  segment code use32 class=code **public**  \_max:  ; stiva  ; [ESP] <- adresa de revenire  ; [ESP+4] <- valoare suma  ; [ESP+8] <- valoare M  ; [ESP+12] <- adresa lui M  Mov esi, [ESP+12]  Mov eax, [ESP+4]  Mov ebx, [ESP+8]  CMP eax, ebx  JG Caz1  Mov esi, ebx  Jmp Final  Caz1:  Mov [esi], eax  Final:  ret | Bits 32 global \_lungime  segment code use32 class=code **public**  \_lungime:  ;adresa cuvant = [esp+4]  ;adresa lungime = [esp+8]  Mov esi, [ESP+4]  Mov ebx, 0  cld  Numar:  Lodsb  Cmp al, 0  Jz out\_Numar  Inc ebx  Jmp Numar  Out\_Numar:  Mov esi, [esp+8]  Mov [esi],ebx  ret |
|  |  |