SUPORT SEMINAR 4 - ASSEMBLER

PROCEDURI

Atenţie: La procesoarele 80x86 următoarea instrucţiune de executat este determinată de conţinutul registrelor CS:IP (registrul asociat segmentului de cod CS şi pointerul la instrucţiune IP) ce reprezintă adresa fizică a segmentului de cod şi offset-ul în cadrul acestuia.

Instrucţiunile de salt necondiţionat (Ex. JMP) permit salturi în program însă întoarcere la punctul de salt este gestionată în întregime de utilizator care trebuie fie să indice punctul respectiv printr-o etichetă care ulterior să fie de o instrucţiune de salt, fie să salveze conţinutul registrului IP.
Apelurile de proceduri, reprezintă salturi la secvenţe de cod (reutilizabile) declarate de utilizator în corpul procedurii, care permit reîntoarcerea la punctul de start. Acest lucru este posibil prin implementarea instrucţiunilor CALL (apel procedură) şi RET (ieşire din procedură şi întoarcere în programul apelator).
Diferenţa dintre CALL şi JMP este dată de faptul că prima instrucţiune salvează pe stiva înainte de a face saltul adresa instrucţiunii următoare.
Tipuri de CALL

- near; salvează pe stivă valoarea pe 16 biţi din IP şi face salt la prima instrucţiune din corpul procedurii; are formă internă pe 3 bytes: 1 byte codul operaţiei (E8h), 2 bytes reprezentând distanţa ca număr de bytes până la codul procedurii;
- far; salvează pe stivă valoarea din CS şi apoi din IP; are formă internă pe 5 bytes: 1 byte codul operaţiei (9Ah), 2 bytes offset şi 2 bytes adresa segemntului.

Sintaxa declarării unei proceduri:

```
Nume_procedură PROC [FAR | NEAR]
..............
[Nume_procedura] ENDP
```

Structura generală a unei proceduri scrisă în limbaje de asamblare este:

Nume_procedură PROC [FAR | NEAR]
Salvare registre
Bloc prelucrare date cu referirea parametrilor formalii
Stocare rezultate
Restaurarea registre
RET [nr_octeţi]
[Nume_procedură] ENDP

Apelul şi ieşirea din proceduri

Apelul unei proceduri se face prin instrucțiunea CALL:

- sintaxă:

```
CALL operand
```

- aceasta salvează pe stivă adresa instrucțiunii următoare (dacă saltul este de tip intersegment – far se salvează înainte şi adresa din CS) şi execută salt la locația indicată de operand;
- tipul operandului este:

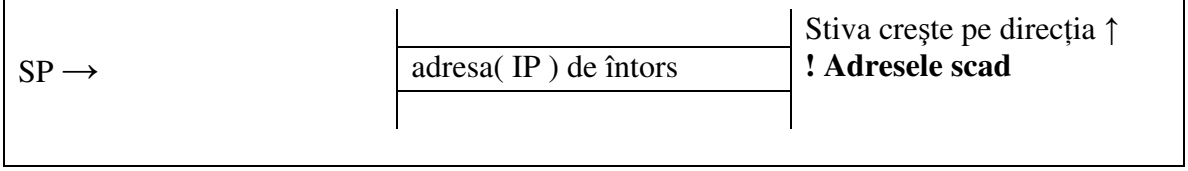
Tip operand	Descriere
CALL nume_procedură	Numele procedurii apelate reprezintă în esență o etichetă.
CALL etichetă	Procesorul presupune în această situați că eticheta este locală şi atunci face un salt de tip NEAR. Aceasta trebuie să fie la o distanță cuprinsă în intervalul [-32768,32676] de bytes. A se vedea exemplul de mai jos.
CALL FAR PTR eticheta	Eticheta este în alt segment. Înlocuieşte CS şi IP cu segmentul şi offsetul etichetei.
CALL registru sau variabilă	Conținutul din registru sau variabilă este copiat în IP după ce acesta a fost salvat pe stivă. Deci variabilasau registrul respectiv reprezintă de fapt un pointer near (ține doar offsetul unei proceduri sau etichete)
CALL WORD PTR variabilă	Conținutul variabilei reprezintă un offset şi are loc un salt de tip NEAR. De obicei variabila este reprezentată de o adresare indirectă cu registru index (SI,DI), [SI] sau registru de bază (BX). [BX] şi atunci trebuie specificat câti bytes trebuie citiți (trebuie indicat procesorului tipul de salt – near, se citesc 2 octeți; far se citesc 4 octeți). De văzut exemplul cu apeluri indirecte de proceduri.
CALL DWORD PTR adresă	Conținutul variabilei reprezintă un segment + offset şi are loc un salt de tip FAR. De obicei variabila este reprezentată de o adresare indirectă cu registru index (SI,DI), [SI] sau registru de bază (BX). [BX]. De văzut exemplul cu apeluri indirecte de proceduri.

- instrucțiunea CALL se traduce prin secvențele de instrucțiuni:

<u>pentru NEAR</u>

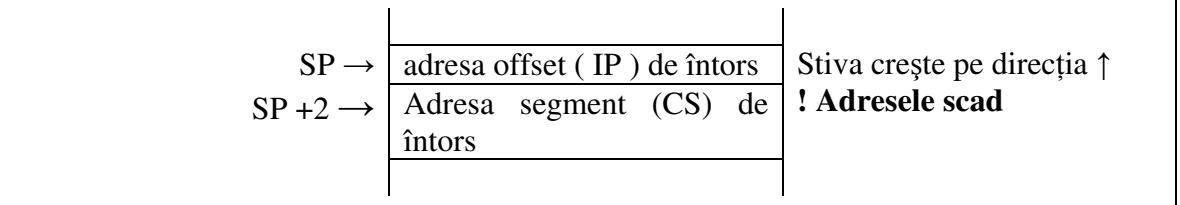
PUSH IP ; salvare adresa instrucțiunii următoare lui CALL
JMP adresa_salt ;adresa_salt reprezintă un offset şi este indicată de operand

Imaginea stivei după CALL şi la poziționarea pe prima instrucțiune din corpul procedurii



pentru FAR
PUSH CS ;salvare adresa segment de cod curent
PUSH IP ; salvare adresa instrucțiunii următoare lui CALL
JMP adresa_salt ;adresa_salt reprezintă un sehment+offset şi este indicată de operand

Imaginea stivei după CALL şi la poziționarea pe prima instrucțiune din corpul procedurii



Ieşirea din proceduri se realizează prin instrucțiunea RET:

- sintaxă:

```
RET [valoare numerică]
```

- aceasta scoate de pe stivă adresa instrucțiunii următoare (dacă saltul este de tip intersegment – far se restaurează după IP şi CS salvat anterior) şi inițializează registrele care controlează executarea instrucțiunilor, IP şi CS (ambele dacă procedura a fost de tip FAR);
- dacă se dă şi valoarea numerică opțională are loc curățarea stivei prin modificarea poziției curente din stivă (indicată de SP) adunând la SP atâția bytes câți indică valoarea numerică;
- instrucțiunea RET se traduce prin secvențele de instrucțiuni:

pentru NEAR
POP IP ;reiniționalizează registrul IP cu valoarea salvată la intrarea în procedură
;cum acest registru conține offsetul instrucțiunii de executat, programul se
;reia de la instrucțiunea următoare saltului
[add SP, valoare_numerica];instrucțiune opțională care se execută doar dacă se dă
;instrucțiunea RET + valoare numerică
pentru FAR
POP IP
POP CS ;reiniționalizează registrul CS cu adresa segmentului de cod

[add SP,valoare_numerica]

- instrucțiunea ştie ce ieşire să facă în funcție de modul în care a fost declarată procedura sau mai exact în funcție de tipul apelului FAR sau NEAR; la asamblare fiecare instrucțiune de tip *return* este înlocuită cu RETN sau RETF (instrucțiunile pot fi utilizate – a se vedea exemplul următor) care reprezintă forma explicită a instrucțiunii de ieşire.

Realizarea unei proceduri fără PROC şi ENDP

Exemplul următor subliniază modul de interpretare internă a procedurii asemenea unei secvențe de cod la care se realizează salturi şi de la care programul ştie să se întoarcă gestionând adresa punctului de start.
Procedura realizează suma a două numere de tip word ale căror valori sunt puse pe stivă iar rezultatul este returnat în registrul CX.

```
.model small
.286
.stack 100h
.data
      a dw 5
      b dw 3
      s dw ?
.code

      mov AX,@data
      mov DS,AX

      push a      ;pun valoarea lui a pe stiva
      push b;     ;pun valoarea lui b pe stiva

      call NEAR PTR start_procedura ;apel explicit de procedura NEAR

      mov s, CX   ;pun rezultatul în s

      mov AX, 4c00h
      int 21h

start_procedura:  ;etichetă ce indică începutul procedurii
      push BP     ;salvez valoarea din BP
      mov BP,SP   ;inițializez BP cu valoarea lui SP pentru a-l utiliza
                  ;ca reper în citirea datelor de pe stivă
      push AX     ;salvez valoarea lui AX
      mov AX,[BP+6]     ;copiez în AX valoarea lui a primită pe stivă
;a se vedea figura de mai jos care descrie stiva in acest moment
      add AX,[BP+4]     ;adun în AX valoarea lui b primită pe stivă
      mov CX,AX         ;pun rezultatul în CX
      pop AX      ;restaurez AX
      mov SP,BP   ;copiez în SP valoarea din BP
                  ;aduc SP la poziția reperului
      pop BP      ;restaurez SP
      retn        ;return de tip NEAR - scote doar IP de pe stiva
end
```

Imaginea stivei la momentul adunării celor două valori de pe stivă

Poziție în stivă relativă față de BP	**SP**		
BP-2→	SP →	valoare din AX	
BP→	SP + 2 →	valoare din BP	Stiva creşte pe direcția ↑
BP+2→	SP + 4 →	adresa offset (IP) de întors	**! Adresele scad**
BP+4→	SP + 6 →	valoare b	
BP+6→	SP + 8 →	valoare a	

Acelaşi program se scrie utilizând directivele PROC şi ENDP:

```
.model small
.286
.stack 100h
.data
      a dw 5
      b dw 3
      s dw ?
.code
      mov AX,@data
      mov DS,AX
      push a
      push b
      call suma
      mov s, CX
      mov AX, 4c00h
      int 21h
suma PROC
      push BP
      mov BP,SP
      push AX
      mov AX,[BP+6]
      add AX,[BP+4]
      mov CX,AX
      pop AX
      mov SP,BP
      pop BP
      ret
ENDP
end
```

Proceduri de tip FAR

Procedurile FAR sunt proceduri care se găsesc în alt segment decât cel curent. Pentru a exemplifica o astfel de situație se definesc explicit segmentele şi se renunță la definirea simplificată a unui program utilizând directivele `.model` `.code` `.stack` `.data`.

De exemplu, se scrie programul care adună două numere de tip *double* trimițând parametrii prin referință pe stivă.

```
Variabile SEGMENT ;declar un segment numit Variabile in care declar
                  ;datele
a     dd    1234677
b     dd    3456123
sum   dd    ?
Variabile ENDS    ;sfârşit segment de date

Stiva SEGMENT     ;declar segment numit Stiva în care rezerv 512 octeți
                  ;pentru a-i utiliza pe post de stivă
      dw    100h  dup(?)
baza  label word  ;declar o variabila simbolica de tip word pentru a
                  ;lua offset-ul in stiva
Stiva ENDS        ;sfârşit segment de stivă

Principal SEGMENT ;declar segment numit Parincipal în care scriu codul
                  ;programului principal
      ASSUME CS:Principal,DS:Variabile,SS:Stiva
                  ;se realizează asocieri logice între registrele de
                  ;segment şi segmentele utilizate - NU înseamnă ca
                  ;sunt şi inițializate registrele
start:
      mov AX, Variabile ;inițializez DS
      mov DS,AX
      mov AX,Stiva      ;inițializez SS
      mov SS,AX
      mov SP,offset baza      ;inițializez SP

      mov AX,offset a   ;pun pe stivă adresa lui a
      push AX
      mov AX,offset b   ;pun pe stivă offset b
      push AX
      mov AX,offset sum ;pun pe stivă offset sum
      push AX

      call FAR PTR suma ;apel procedură FAR

      mov AX,4c00h
      int 21h
Principal ENDS    ;sfârşit segment de cod principal

Procedura SEGMENT ;definesc un alt segment de cod în care scriu
                  ;procedura
      ASSUME CS:Procedura
                  ;asociere logică
suma PROC FAR     ;definesc procedura
      push BP     ;salvez BP
      mov BP,SP   ;inițializez BP cu valoarea lui SP
      push AX     ;salvez AX
      mov SI,[BP+10]    ;copiez în SI offset a
      mov DI,[BP+8]     ;copiez în DI offset b
      mov AX,[SI]       ;copiez în AX cuvânt inferior din a
      add AX,[DI]       ;adun la AX cuvânt inferior din b
      push SI           ;salvez conținut SI - adică affset a
```

```
      mov SI,[BP+6]      ;copiez în SI offset sum
;a se vedea figura următoare ce descrie stiva în acest punct
      mov [SI],AX        ;copiez în sum suma cuvintelor inferioare
      pop SI             ;restaurez SI - adică offset a
      mov AX,[Si+2]      ;adun cu carry cuvinte superioare din a şi b
      adc AX,[DI+2]
      mov SI,[BP+6]
      mov [SI+2],AX      ;copiez în partea superioara a sum rezultatul
      pop AX             ;restaurez AX
      mov SP,BP
      pop BP
      ret 6              ;curăț stiva ştergând logic cele 3 offset-uri
suma ENDP
Procedura ENDS
end start
```

Imaginea stivei la momentul copierii în sum a rezultatului adunării cuvintelor inferioare din *a* şi *b*

Poziție în stivă relativă față de

BP	**SP**		
BP-4→	SP →	valoare din SI	
BP-2→	SP + 2 →	valoare din AX	
BP→	SP + 4 →	valoare din BP	
BP+2→	SP + 6 →	adresa offset (IP) de întors	
BP+4→	SP + 8 →	adresa segment (CS) de întors	Stiva creşte pe direcția ↑ **! Adresele scad**
BP+6→	SP + 10 →	offset sum	
BP+8→	SP + 12 →	offset b	
BP+10→	SP + 14 →	offset a	

Reguli la scriere procedurilor:

- secvențele de cod ce compun corpul procedurii realizează operații cu caracter general;
- utilizarea registrelor în proceduri implică salvarea conținutului acestora pe stivă înainte de efectuarea prelucrărilor şi restaurarea lor de pe stivă înainte de a ieşi; pentru acest lucru se utilizează instrucțiunile PUSH (pune pe stivă) şi POP (scoate de pe stivă). De exemplu:

```
ProcTest PROC
push AX
push BX
push CX
……
Pop CX
Pop BX
Pop AX
ret
```

```
                    ENDP
```

Atenţie: Restaurarea registrelor cu POP se face în ordine inversă salvării cu PUSH.

- citirea datelor de pe stivă se realizează **numai** cu registrul **BP** (**Atenţie** pentru că chestia asta mi-a scăpat şi mie)

Apeluri indirecte de proceduri

Se realizează prin intermediul pointerilor la funcţii. De exemplu programul care adună două numere de tip word prin intermediul unei proceduri.

```
.model small
.286
.stack 100h
.data
      pointer_functie         dw ?  ;pointer la funcţie de tip NEAR
      pointer_functie_far     DD ?  ;pointer la funcţie de tip FAR
      a dw 5
      b dw 3
      s dw ?
      dif dw ?
.code

      mov AX,@data
      mov DS,AX

      mov AX, offset suma           ;incarc in AX offset proecdură
      mov pointer_functie,AX        ;iniţializez pointerul la procedură

      mov AX,a    ;trimit parametrii prin regişţrii
      mov BX,b
      call pointer_functie    ;apelez procedura NEAR

      mov s,CX
                  ; incarc pointerul cu offsetul si adresa segmentului
      mov WORD PTR pointer_functie_far,offset diferenta
      mov WORD PTR pointer_functie_far+2,seg diferenta
      mov AX,a    ;trimit parametrii prin regişţrii

      call pointer_functie_far      ;apelez procedura FAR
      ; sau call FAR PTR diferenta  ;apel prin nume

      mov dif,CX
      mov AX, 4c00h
      int 21h
suma PROC
      add AX,BX
      mov CX,AX
      ret
ENDP
diferenta PROC FAR      ;declar explicit procedura de tip FAR
      sub AX,BX
      mov CX,AX
```

```
      ret
ENDP
end
```

În exemplul anterior procesorul ştie ce tip de procedură să apeleze pentru că pointerii sunt declaraţi ca fiind de tip word, respectiv, double, ceea ce înseamnă că poate să conţină doar offsetul procedurii apelate (în cazul near) sau adresa completă, segment de cod + offset (pentru far).
În caz ca există un vector de variabile de tip pointeri la funcţii, la apelul procedurii trebuie indicat explicit tipul acesteia prin WORD PTR (pentru NEAR) sau DWORD PTR (pentru FAR).
Dacă în exemplul anterior încarc în SI offsetul de început al variabilei *pointer_functie* (simulez utilizarea unui vector) şi atunci apelurile echivalente sunt:

```
call WORD PTR [SI]   ;apel de tip near pentru suma
call DWORD PTR [SI+2] ;apel de tip far pentru diferenţă
```

Dacă se dă apelul call [SI] atunci implicit se consideră că apelul este de tip NEAR şi se vor citi 2 octeţi de la adresa [SI] reprezentând offsetul saltului. De aceea este important (mai ales în cazul salturilor de tip FAR) să se indice explicit tipul saltului prin WORD sau DWORD.

Transmiterea parametrilor de intrare în proceduri

[urmează]

Transmiterea parametrilor de ieşire din proceduri

[urmează]

Utilizarea procedurilor din fişiere incluse

[urmează]

MACRODEFINIŢII

[urmează]