

LUCRAREA Nr. 2

DEZASAMBLORUL-DEPANATOR ("DEBBUGER") "AFD"

1. Scopul lucrării

Lucrarea de față își propune familiarizarea cu posibilitățile de dezasamblare, lansare în execuție și depanare a programelor executabile scrise în limbajul de asamblare al microprocesoarelor Intel funcționând "în modul real".

2. Prezentare generală

După lansarea programului AFD, pe ecranul calculatorului apare sigla de prezentare a acestuia. După apăsarea oricărei taste se intră în ecranul principal, prezentat schematic mai jos.

REGISTRE PROCESOR	STIVA	FANIOANE
ZONA INTRODUCERE COMENZI	ZONA 1 MEMORIE	
ZONA PROGRAM		
ZONA 2 MEMORIE (HEX)		
ZONA 2 MEMORIE (ASCII)		
FUNCTII PREDEFINITE PE TASTELE F1-F10		

Zona "registre" afișează conținutul registrelor procesorului. În plus mai apar registrele simbolice **HS** și **FS**, folosite de obicei în adresarea unor zone de memorie fixe, independente de registrele segment ale procesorului.

În zona "stivă" sunt afișate ultimele 4 cuvinte (8 octeți) introduse în stivă.

Zona "fanioane" arată atât conținutul registrului de fanioane (16 biți) cât și valoarea fiecărui fanion în parte.

Accesul în aceste zone se poate face cu ajutorul tastelor funcționale de direcție (**F7 - F10**). Deplasarea în interiorul zonelor se face cu săgeți sau cu tasta **TAB**. Atât registrele cât și fanioanele sunt modificabile direct prin suprascriere. Stiva nu poate fi modificată direct.

În zona "introducere comenzi" se face introducerea comenzilor date debugger-ului. Editarea unei comenzi se face cu comenzile uzuale de editare. Ștergerea comenzii se poate face cu **ESC**. Tasta **F3** permite readucerea ultimelor 6 comenzi introduse.

Zona "program" afișează 8-9 instrucțiuni dezasamblate, începând de la adresa curentă. Instrucțiunea curentă este afișată în mod video invers. Accesul în această zonă se poate face cu comanda **A** (asamblare), fiind posibilă modificarea directă a liniilor de program prin suprascriere.

Zonele "memorie" 1 și 2 afișează conținutul a două zone de câte 256 octeți. Zona 2 este afișată și cu caractere ASCII. Accesul în zonele de memorie se poate face cu tastele funcționale de direcție. În interiorul zonelor, deplasarea se poate face cu săgeți sau cu **TAB**, modificările făcându-se prin suprascriere.

Din oricare dintre zonele de mai sus se poate reveni în linia de introducere comenzi cu **ENTER**.

Pe ultima linie a ecranului sunt afișate comenzile corespunzătoare tastelor funcționale **F1 - F10**.

3. Comenzile disponibile în linia de comandă

Acestea se introduc în zona de introducere comenzi urmate de **ENTER**. Caracterele folosite pot fi majuscule sau minuscule. Ștergerea unei comenzi se poate face cu **ESC**. La introducerea incorectă a unei comenzi sau a unui parametru, debugger-ul răspunde cu mesaje de eroare corespunzătoare și un semnal sonor. Aceste mesaje rămân pe ecran până la apăsarea unei taste.

În comenzile de mai jos s-au folosit următoarele denumiri :

- **nume_fisier** - numele unui fișier specificat complet conform sistemului de operare DOS (**nume.extensie**). Dacă fișierul nu se află în directorul curent, trebuie introdusă și calea până la acesta;
- **adresa** - adresa fizică formată din adresa segment și adresa efectivă, conform formulei

$$AF = AS \uparrow 0H + AE, \text{ unde}$$

$$AS = (CS) \text{ sau } (DS) \text{ sau } (SS) \text{ sau } (ES)$$

$$AE = (IP) \text{ sau } (SI) \text{ sau } (DI) \text{ sau } (SP) \text{ sau } (BP) \text{ sau}$$

(BX) sau **adr**.

Conform notațiilor folosite de asamblare și dezasamblare, există următoarea echivalență de notație:

$$(CS) \uparrow 0H + (IP) \equiv CS:IP.$$

Pentru adresarea indirectă se folosește notația:

$$(\text{adresa}) \equiv [\text{adresa}]$$

Dacă nu se specifică registrul segment, la adresare se folosește registrul segment implicit al fiecărei comenzi;

- **registru** - oricare dintre registrele procesorului precum și **HS, FS**;

- **valoare** - o valoare numerică sau conținutul unui registru. Spre deosebire de alte programe utilitare, la acest debugger valorile numerice sunt considerate implicit hexazecimale. Valorile zecimale se introduc cu prefixul "%". Sunt permise și expresii aritmetice simple;
- **sir** - listă de valori sau caractere ASCII separate prin virgulă. Caracterele ASCII trebuie introduse între apostrofuri (' ').

L nume_fisier [parametri][,adresa] - încarcă fișierul specificat în memorie, începând de la adresa dată. Segmentul implicit este **CS**. Dacă adresa lipsește, fișierul se încarcă de la adresa **CS:100**. Dacă încărcarea s-a făcut corect, numărul de octeți încărcați este afișat pe 32 biți în registrele **BX,CX**.

Exemplu: **L PROG1.COM**

W nume_fisier, adresa, lungime - scrie conținutul unui bloc de memorie pe disc. Segmentul implicit este **DS**. Lungimea maximă a fișierului este **FFFF** sau **%65535**.

Exemplu: **W PROG2.DAT,CS:100,200**

registru = valoare - modifică conținutul registrului specificat. Registrul de fanioane este denumit **FL**. Fanioanele se pot modifica și individual, putând lua valorile **0** sau **1**.

Exemple: **AX = F0F0**
SI = %1234
CF = 1

D adresa - afișează instrucțiunile dezasmblate în zona de program, începând de la adresa dată. Segmentul implicit folosit la adresare este **CS**. Linia curentă devine cea specificată de adresă.

Exemplu: **D 200**

M n adresa - modifică adresa de început a zonei de memorie **n** (1 sau 2). Segmentul implicit este cel afișat în zona respectivă. Se poate folosi și adresarea indirectă **[registru]**.

Exemple: **M 1 400**
M 2 ES:[BX]

G [adresa_start][,adresa_breakpoint] - lansează în execuție un program de la adresa specificată. Segmentul implicit este **CS**. Dacă nu se specifică adresa de start, execuția se lansează de la linia curentă (în mod video invers pe ecran). Un "breakpoint" reprezintă o adresă la care se dorește întreruperea execuției programului. La oprirea prin breakpoint se pot examina efectele programului executat (registrele procesorului și diverse zone de memorie) după care programul poate fi rulat în continuare. Rularea unui program poate fi întreruptă cu **CTRL/ESC**.

Exemplu: **G 100,110**

A [adresa] - intră în modul asamblare. În acest mod se pot modifica instrucțiunile din zona program. Dacă nu se specifică adresa, modul asamblare începe de la linia curentă. Modificările se fac prin suprascriere urmată de **ENTER**. Dacă nu este urmată de **ENTER**, modificarea nu este luată în considerare. Deplasarea în zona program se face cu săgeți. Ieșirea din modul asamblare se face cu **CTRL/HOME**.

Exemplu: **A 100**

F adresa, contor, sir - umple un bloc de memorie începând de la adresa specificată, cu șirul dat, de un număr de ori indicat de contor. Segmentul implicit este **DS**.

Exemple: **F 0,10,55**
F ES:20,8,'ABCD'

CO adresa_sursa, adresa_destinatie, lungime - copiază un bloc de date începând de la adresa sursă, cu lungimea specificată, la adresa destinație. Segmentul implicit este **DS**.

Exemple: **CO 0,100,20**
CO CS:100,ES:100,10

286 ON | OFF - comută operațiile de dezasamblare și asamblare în mod 80286 sau 8086. În mod 8086 asamblorul nu va recunoaște instrucțiunile speciale ale procesorului 80286. Setarea modului procesor se face automat la lansarea debugger-ului. Introducerea comenzii 286 fără parametru afișează modul procesor curent.

MODE M - comută afișarea pe ecran pentru monitor monocrom.

MODE C - comută afișarea pe ecran pentru monitor color.

MODE A[LTERNATE] ON | OFF - modul **ON** permite utilizatorului urmărirea pe ecran a efectelor rulării programului. Trecerea de la ecranul debugger-ului la cel al aplicației și invers se face cu **F6**. Introducerea comenzii **MODE** fără parametru afișează modul de lucru video curent.

QUIT - părăsește debugger-ul.

4. Funcții predefinite pe tastele F1 - F10

Acestea se introduc cu ajutorul tastelor **F1 - F10**, fără a mai fi urmate de **ENTER**. Semnificația lor este indicată pe ultima linie a ecranului.

F1 (Step) - permite executarea unui program, instrucțiune cu instrucțiune, începând de la linia curentă. După executarea unei instrucțiuni se pot examina efectele acestora în registrele procesorului, memorie etc.

F2 (Step Procedure) - la fel ca mai sus, cu deosebirea că instrucțiunile de tip procedură se execută într-un singur pas. Este recomandabilă la executarea programelor în care nu se dorește rularea pas cu pas a instrucțiunilor de tip **CALL**, **INT**, **LOOP** etc.

- F3 (Retrieve)** - readuce în linia de introducere comenzi ultima comandă introdusă. Sunt memorate ultimele 6 comenzi introduse.
- F4 (Help)** - oferă utilizatorului o descriere a comenzilor disponibile. Trecerea de la o pagină la alta se face cu **SPACE**. Se revine în ecranul principal cu **ENTER**.
- F5 (Set Breakpoint)** - intră în meniul de fixare a adreselor și a condițiilor de breakpoint. De asemenea, permite dezasamblarea de la o adresă dată. Acest submeniu are propriile sale comenzi funcționale. Revenirea în ecranul principal se face tot cu **F5**.
- F6 (Toggle Screen)** - permite trecerea de la ecranul principal la cel al aplicației și invers. Această tastă devine activă numai după folosirea comenzii **MODE A ON**.
- F7 (Up)** - tastă de direcție. Trece în zona de sus.
- F8 (Down)** - tastă de direcție. Trece în zona de jos.
- F9 (Left)** - tastă de direcție. Trece în zona din stânga.
- F10 (Right)** - tastă de direcție. Trece în zona din dreapta.

Revenirea din orice zonă a ecranului în linia de introducere comenzi se poate face cu **ENTER**.

5. Desfășurarea lucrării

- 5.1. Se lansează asamblorul TASMB.
- 5.2. Se editează (**E**) programul sursă din anexă, se salvează pe disc (**W**) cu numele **P2.ASM**, se activează (**O**) opțiunea **F8 (COM)** și se assemblează (**A**).
- 5.3. Se părăsește asamblorul (**Q**).
- 5.4. Se lansează debugger-ul AFD. Cu orice tastă se intră în ecranul principal.
- 5.5. Folosind comenzile de direcție **F7 - F10**, săgețile și **TAB**, se verifică modificarea prin suprascriere a registrelor **AX, BX, CX, DX, SI, DI** și a fanioanelor.
- 5.6. Se revine cu **ENTER** în linia de introducere comenzi și se modifică aceleași registre și fanioane folosind comenzile:

registru = valoare
fanion = valoare
- 5.7. Folosind comanda **M**, se fixează zonele de memorie 1 și 2 la adresele de început **DS:100**, respectiv **ES:200**.
- 5.8. Cu ajutorul comenzii **F**, se umple un bloc de memorie începând de la adresa **DS:100**, pe o lungime de 32 octeți, cu valoarea 64. Se observă efectul comenzii în zona de memorie 1.
- 5.9. Se umple un bloc de memorie începând de la adresa **ES:200**, pe o lungime de 48 octeți, cu șirul **'AB'**. Se observă efectul comenzii în zona de memorie 2.
- 5.10. Folosind comanda **CO**, se copiază un bloc de memorie de la adresa **DS:100**, la adresa **ES:200**, pe o lungime de 32 octeți. Se observă efectul în zona de memorie 2.
- 5.11. Se setează modul video alternat (**MODE A ON**).
- 5.12. Cu comanda **L** se încarcă programul **P2.COM** la adresa **CS:100**. Se observă efectul în zona de program.

5.13. Folosind comanda **G**, se lansează în execuție programul încărcat, de la adresa **CS:100**, cu breakpoint la adresa **CS:143**. Se observă ecranul aplicației cu ajutorul comenzii **F6**.

5.14. Se lansează în execuție programul în continuare (**G**), fără breakpoint. Se observă din nou ecranul aplicației.

5.15. Folosind comanda de asamblare (**A**), se modifică instrucțiunea aflată la adresa **CS:119** astfel:

MOV BX,000F devine **MOV BX,00F0** (urmată de **ENTER**)

Se revine în linia de introducere comenzi cu **CTRL/HOME**.

5.16. Cu ajutorul comenzii **D** se fixează zona program de la adresa **CS:100**.

5.17. Se refac punctele 5.13 și 5.14 observându-se efectul modificării de la pasul 5.15.

5.18. Folosind comanda **F2**, se rulează programul pas cu pas, observându-se efectele asupra registrelor procesorului și ecranului aplicației.

5.19. Se părăsește debugger-ul cu comanda **QUIT**, revenindu-se în sistemul de operare.

ANEXA

Programul sursă P2.ASM

```
org 100h
mov ax,cs
mov ds,ax
mov ax,3
int 10h           ; sterge ecran
mov di,0b1ch
mov ah,2
mov bh,0
mov dx,di
int 10h           ; pozitioneaza cursor
adr1 mov si,offset mesaj
mov al,[si]
mov bx,0fh
mov cx,1
mov ah,9
int 10h           ; scrie un caracter
inc di
mov ah,2
mov bh,0
mov dx,di
int 10h           ; avanseaza cursor
inc si
cmp si,offset endms
jne adr1
int 20h           ; terminare program
mesaj db 'TEST VIDEO'
endms equ $
end
```