24. JEDNOSTAVNI PRIMJERI PROGRAMA NA JEZIKU ASEMBLERA. `primjera`

Svaki DOS ili Windows računar sadrži program `debug.exe`. Taj program služi za četiri primjera koji su sada na redu.

1. Aritmetički program. Sastaviti program za računanje četiri broja y=a·b·2+1, y=y−2, y=y−2 i y=y−16. Neka se na kraju odštampaju ta četiri broja.

Rješenje. Umjesto 100 na ekranu piše 0F6E: 0100. Umjesto 102 na ekranu piše 0F6E:0102. Itd.

```
    A 100 loaduj na 100,101,... u tekućem s.
100 JMP 108       goto 108
102 DB E          a=14 (1 bajt)
103 DB 3          b=3
104 DB 0          mjesto za y1
105 DB 0          y2
106 DB 0          y3
107 DB 0          y4
108 MOV AL,[102]  y=a=14
10B MOV CL,[103]
10F MUL CL        AX←AL·CL y=a·b=42
111 SHL AX,1      shift left AX y=2y=84
113 INC AX        increment y=y+1=85
114 MOV [104],AL
117 SUB AX,2      y=y−2=83
11A MOV [105],AL
11D SUB AX,2      y=y−2=81
120 MOV [106],AL
123 SUB AX,10     y=y−16=65
126 MOV [107],AL
129 MOV AH,2      INT 21 & AH=2: štam-
12B MOV DL,[104]  panje slova (bajta) DL
12F INT 21        poziva se rutina prekida
131 MOV DL,[105]
135 INT 21        štampa se drugi broj
137 MOV DL,[106]
13B INT 21        štampa se treći broj
13D MOV DL,[107]
141 INT 21
143 INT 20        rutina koja okončava rad
145 ␣             prazan red (izađi iz l.)
    G             ajde ili run executable
```

Na kraju će se odštampati USQA. Znamo da slovo U ima ASCII kod 85 S 83 Q 81 A 65. Još će se odštampati Program terminated normally. Ovo je bio program `primjera`.

25. `primjerb`.

2. Rad sa stekom. Sastaviti program za računanje jednog broja y po nizu formula y=0, y=(y+c)·2−1, y=(y+b)·2−1, y=(y+a)·2−1. Neka se na kraju programa dvaput odštampa broj y. Staviti a=4, b=5 i c=6, tako da treba da se dobije y=69.

Rješenje. Ovo je program `primjerb`. Na početku rada programa `primjerb` stek je prazan. Tokom izvršavanja programa `primjerb` stanje u steku se mijenja i to (kada se izvrši naredba):

(push) 4; (push) 4,5; (push) 4,5,6; (pop) 4,5; (pop) 4; (pop) prazan

```
A 100 loaduj na 100,... u tekućem segmentu
MOV AX,4    a=4
PUSH AX     stavi AX na stek
MOV AX,5    b=5
PUSH AX     stavi AX na stek
MOV AX,6    c=6
PUSH AX     stavi AX na stek
MOV AX,0    y=0
POP CX      skini sa steka i stavi u CX
ADD AX,CX   AX←AX+CX (y=y+6=6)
SHL AX,1    left shift 1 place AX (y=2y=12)
DEC AX      decrement AX (y=y−1=11)
POP CX      skini sa steka i stavi u CX
ADD AX,CX   y=y+5=16
SHL AX,1    y=2y=32
DEC AX      y=y−1=31
POP CX      skini sa steka i stavi u CX
ADD AX,CX   y=y+4=35
SHL AX,1    y=2y=70
DEC AX      y=y−1=69='E'
MOV DX,AX   DX←y, da bi se odštampalo y
MOV AH,2    priprema za štampanje jednog slova
INT 21      štampaj
INT 21      štampaj
INT 20      halt
␣           izađi iz loadovanja
G           run executable
```

Kada sve ovo otkucamo (otkucamo pošto smo pozvali `debug.exe`) onda će na ekranu pisati EE.

26. `primjerc`.

3 `primjerc`. Učitavanje pojedinačnih slova i štampanje niza slova.

Sastaviti program koji će na ekranu prikazati poruku `NIZ␣SLOVA`. Koristiti prekid broj $(21)_{16}$ tj. koristiti naredbu INT 21 i to njegovu funkciju

$(9)_{16}$ – štampanje niza slova. Neka na ekranu piše u prvom redu `NIZ␣SLO` a u drugom redu `VA`. Prva dva slova N i I učitati preko tastature tokom izvršavanja programa. Koristiti prekid broj $(21)_{16}$ i to njegovu funkciju $(1)_{16}$ – učitavanje jednog slova sa odjekom. Ostala slova zadati programski.

Znamo da se funkcija prekida INT 21 zadaje upisivanjem odgovarajućeg broja u AH. Znamo da definicija funkcije AH=9 glasi: štampaj redom slova počev od mjesta DX (počev od mjesta DS:DX) pa naprijed, sve dok se ne naiđe na mjesto u kome piše slovo $. Funkcija AH=1: slovo koje se učitava dolazi u AL. Funkcija AH=8: učitavanje jednog slova bez odjeka, slovo koje se učitava dolazi u AL.

Naš raspored upotrebe memorije: na 100,101, ... naredbe, a na 200,201,... podaci (u tekućem segmentu). Znamo da `debug.exe` koristi heksadekadni brojni sistem.

Rješenje.

```
A 100           load point = 100
MOV AH,1        funkcija prekida
INT 21          prekid
MOV [200],AL    prvo slovo ide na svoje mjesto
INT 21          prekid
MOV [201],AL    drugo slovo ide na svoje mjesto
MOV AH,9        funkcija prekida
MOV DX,200      gdje počinje niz slova
INT 21          prekid
INT 20          halt
␣               nema više
A 202           load point = 202
DB 5A           'Z'
DB 20           '␣'
DB 53           'S'
DB 4C           'L'
DB 4F           'O'
DB D            CR carriage return
DB A            LF line feed
DB 56           'V'
DB 41           'A'
DB 24           '$'
␣               nema više
G               run executable (go)
```

Otkucati NI. Sada na ektanu piše

`NINIZ␣SLO`

`VA`

Promijenimo prvu naredbu: neka umjesto MOV AH,1 sada bude MOV AH,8. Otkucati NI. Sada na ekranu piše

`NIZ␣SLO`

`VA`

27. `primjerd`.

4 `primjerd`. Rad sa nizom. Ovaj primjer služi da se ilustruje mogućnost indirektnog adresiranja.

Sastaviti program za sabiranje svih članova jednog niza brojeva (za računanje zbira četiri broja).

Plan. Zbir se drži na adresi 200 a sabirci se drže na adresama 202, 204, 206 i 208. Sabirci se zadaju programski, a zbir se ostavlja u AX. Registar BX služi kao indeks–registar.

Imamo da je BX=202, BX=204, BX=206 i BX= 208 tokom rada programa. Tako da je [BX]= [202], ..., [BX]=[208]. Prema tome, glavnu ulogu u programu ima naredba ADD AX,[BX].

U programu se ponavlja: dodaj sabirak i ispitaj treba li još.

Rješenje.

```
      A 100           load point = 100
      MOV BX,202      BX←202
→ 103 MOV AX,[200]    AX←c(200)
      ADD AX,[BX]     AX←AX+c(BX)
      MOV [200],AX    c(200)←AX
      CMP BX,208      compare
      JE 115          jump if equal
      INC BX          BX←BX+1
      INC BX          BX←BX+1
      JMP 103         jump
→ 115 INT 20          halt
      ␣               nema više
      A 200           load point = 200
      DW 0      y=0, y=a1+a2+a3+a4=3CA
      DW F1           a1=F1
      DW F2           a2=F2
      DW F3           a3=F3
      DW F4           a4=F4
      ␣               nema više
      T ponovo T itd. trace
```

Na svako T (izvršavanje korak po korak) imamo na ekranu sadržaj procesorovih registara.

Vidimo da je na kraju AX=03CA.

Poslije izvršavanja naredbe INT vidimo adresu CS:IP na kojoj počinje kod rutine.