PRAKTIČNA VJEŽBA SA PROGRAMOM DEBUG.EXE (8086/PENTIUM ASEMBLER)

Neobavezni materijal. Sadržaj: 1. uvodni primjer, 2. obično adresiranje, 3. indirektno adresiranje, 4. rad sa stekom, 5. rad sa sistemskom rutinom i 6. uslovni skok.

1. Uvodni primjer – dešava se u procesoru (ne idemo u memoriju)

Pozvati ms–dos prompt odnosno command prompt i otkucati:

debug	poziva se program (svaki put treba enter)
a	unošenje, assemble (ili svejedno a 100)
mov ax, 6789	ax ← 6789 (hex), move
mov bx, 2345	bx ← 2345
mov cl, 2	cl ← 2
mov ch, 1	ch ← 1
inc dx	dx ← dx + 1, increment
add ax, dx	ax ← ax + dx, add
add ax, dx	ax ← ax + dx
shl dx, 1	left shift dx 1 place
shl dx, 1	(praktično dx ← 2 · dx)
shl dx, 1	(isto)
sub ax, dx	ax ← ax − dx, subtract
xor ax, ax	ax ← ax xor ax bit–po–bit (praktično ax ← 0)
not bx	bx ← not bx bit–po–bit (praktično prvi komplement)
mov ax, bx	ax ← bx
mov ah, ff	ah ← ff, znači da se u ah upisuje konkretna brojna vrijednost
neg ax	ax ← −ax, negacija, promjena znaka, praktično drugi komplement
(prazan red)	izlazi se iz "a" time što se pritisne dugme enter
t =100	trace, izvršavanje korak po korak od 100
t	jedan korak `Pratimo sadržaj procesorovih registara`
t itd.	idući korak
(stalno t)	sve dok se ne izvrši i instrukcija neg ax
q	quit, izlazi se iz programa

2. Uvodni primjer – dešava se u procesoru i memoriji, obično adresiranje

debug	poziva se program
a	unošenje
mov ax, 15	ax ← 15 (hex), upisuje se konkretna brojna vrijednost
mov [200], ax	c(200) ← ax, iz ax se šalje u memoriju
mov ax, 16	ax ← 16
mov [202], ax	c(202) ← ax, c znači contents = sadržaj
mov ax, 17	ax ← 17
mov [204], ax	c(204) ← ax, dva bajta (na 204 i 205 se šalje)
mov ax, 18	ax ← 18
mov [206], ax	c(206) ← ax, dva bajta, isto se kaže i word
mov ax, [200]	ax ← c(200), donose se dva bajta iz memorije u ax
mov bx, [202]	bx ← c(202), donose se dva bajta (sa adresa 202 i 203) u bx
mov cx, [204]	cx ← c(204), jedna riječ (word) donosi se u cx
mov dx, [206]	dx ← c(206), slično (znamo da je registar dx veličine dva bajta)
add ax, bx	ax ← ax + bx, sabiranje
add ax, cx	ax ← ax + cx, takođe veličine word = dva bajta
add ax, dx	sada je ax = 15 + 16 + 17 + 18 = 5a (hex)

(prazan red)	izlazi se iz "a"
t =100	korak po korak
t	jedan korak `Pratimo tekuću situaciju u programu`
t itd.	idući korak
(stalno t)	sve dok se ne izvrši i instrukcija add ax, dx
q	quit, izlazi se iz programa

adresa	sadržaj
200–201	15
202–203	16
204–205	17
206–207	18

3. Indirektno adresiranje

debug	poziva se program
a	unošenje
mov ax, 281	ax $\leftarrow$ 281
mov [200], ax	c(200) $\leftarrow$ ax, dva bajta
mov ax, 289	ax $\leftarrow$ 289
mov [202], ax	c(202) $\leftarrow$ ax, dva bajta
xor ax, ax	ax $\leftarrow$ 0
mov bx, 200	bx $\leftarrow$ 200
add ax, [bx]	ax $\leftarrow$ ax + c(bx)
inc bx	bx $\leftarrow$ bx + 1
inc bx	bx $\leftarrow$ bx + 1
add ax, [bx]	ax $\leftarrow$ ax + c(bx)
(prazan red)	izlazi se iz "a"
t =100	korak po korak
t	jedan korak `Pratimo kako se mijenjaju vrijednosti u AX i BX`
t itd.	idući korak
q	quit

Dobili smo da je $281 + 289 = 50a$ (hex). Sabirke smo dohvatali indirektno (zato smo njihove adrese bili stavili u registar bx).

adresa	sadržaj
200–201	281
202–203	289

4. Rad sa stekom

sp – stack pointer – adresa gornjeg člana steka
push – stavi na stek (i smanji sp), pop – skini sa steka (i povećaj sp)
znamo da je Intelov stek usmjeren nadolje

debug	poziva se program
a	unošenje
mov sp, 210	sp $\leftarrow$ 210
mov ax, 111	ax $\leftarrow$ 111
push ax	ax ide na 20e i 20f i sp se podešava
mov ax, 222	ax $\leftarrow$ 222
push ax	ax ide (zavisno od toga čemu je jednako sp sada) i sp se prilagođava
mov ax, 333	ax $\leftarrow$ 333

push ax	ax se stavlja na stek (i sp se prilagođava)
mov bx, [20e]	bx ← c(20e)
mov cx, [20c]	cx ← c(20c)
mov dx, [20a]	dx ← c(20a)
(prazan red)	izlazi se iz "a"
t	jedan korak `Vidimo sadržaje registara BX, CX i DX`
t itd.	idući korak
q	quit

Prvo smo iz ax na stek, a onda iz steka u bx, cx i dx.

adresa	sadržaj
20a–20b	333
20c–20d	222
20e–20f	111

5. Rad sa sistemskom rutinom

Primjer upotrebe potprograma ili bolje reći rutine (koja postoji u operativnom sistemu) koju su napisali (sastavili) i stavili na raspolaganje (za upotrebu) korisnicima stručnjaci firme Microsoft.

Znamo da rutina INT 21 ima razne funkcije. Funkcija (radnja) selektuje se upisivanjem odgovarajuće brojne vrijednosti u AH prije pozivanja rutine. Funkcija AH = 2: prikazivanje na ekranu jednog slova (po ASCII). Prikazaće slovo iz DL.

Šablon našeg slučaja:

INT 21	ulazno–izlazne radnje
AH = 2	radnja/funkcija: prikazivanje karaktera
DL = ...	karakter o kome se radi

debug	poziva se program
a 100	unošenje od 100
mov ah, 2	ah ← 2, definiše se funkcija rutine
mov dl, 50	dl ← 50, karakter je pripremljen
int 21	(ponekad se piše int 21h) poziva se rutina
dec dl	dl ← dl − 1 (sada je dl = 4f)
int 21	poziva se rutina, ostala je ista funkcija a drugi je karakter
inc dl	dl ← dl + 1
inc dl	dl ← dl + 1
inc dl	dl ← dl + 1 (sada je dl = 52)
int 21	poziva se rutina, ostala je ista funkcija a drugi je karakter
inc dl	dl ← dl + 1
inc dl	dl ← dl + 1
inc dl	dl ← dl + 1 (sada je dl = 55)
int 21	poziva se rutina, ostala je ista funkcija
sub dl, a	dl ← dl − a (sada je dl = 4b)
int 21	poziva se rutina
mov dl, 41	dl ← 41
int 21	poziva se rutina
int 20	stop
(prazan red)	(sve je uneseno) izlazi se iz "a"
g =100	"go from 100", da počne izvršavanje (od naznačene adrese)
q	quit

Nakon g =100 na ekranu će pisati:

```
PORUKA
Program terminated normally
```

ASCII (hex):

A	B	C	···	K	L	M	N	O	P	Q	R	S	T	U	···	Z
41	42	43	···	4b	4c	4d	4e	4f	50	51	52	53	54	55	···	5a

6. Uslovni skok, gdje se upotrebljava CMP

Naredba CMP *dest*, *src* postavi flagove kao da se oduzima *dest* – *src*. U programima, za njom dolazi uslovni skok JZ ili JNZ ili slično, po pravilu. Na primjer CMP AX, BX postaviće flagove kao da je računata razlika AX ← AX – BX.

Primjer programa sa uslovnim skokovima: učitano slovo prikazuje se na ekranu dva ili četiri puta i tako se ponavlja.

Dakle: pošalji beep i beep, učitaj slovo, dvaput pošalji slovo na ekran, ako je naše slovo A ili B ili C onda ga još dvaput pošalji, ako je slovo = Q onda stop a ako je ≠ Q onda idi na početak.

```
        A 100          počinje loadovanje programa, ovo je direktiva
→ 100   mov ah, 2      ah ← 2, funkcija prikazivanja karaktera (u slučaju int 21)
  102   mov dl, 7      dl ← 7, karakter o kome se radi, ascii 7 = beep
  104   int 21         poziva se sistemska rutina (sistemski potprogram)
  106   int 21         poziva se sistemska rutina
  108   mov ah, 1      ah ← 1, funkcija učitavanja karaktera (u slučaju int 21)
  10a   int 21         poziva se rutina (karakter o kome se radi dolazi u al)
  10c   mov dl, al     dl ← al
  10e   mov ah, 2      ah ← 2, vrsta radnje (funkcija)
  110   int 21         poziva se rutina
  112   int 21         poziva se rutina
  114   cmp dl, 43     compare, upoređivanje po veličini, podesi flagove, ascii 43 hex = C
  117   jg 11d         ako je dl > 43 onda skoči na 11d, jump if greater
  119   int 21         poziva se rutina
  11b   int 21         poziva se rutina
→ 11d   cmp dl, 51     upoređivanje po veličini dl i 51, ascii 51 hex = Q
  120   jne 100        ako je dl ≠ 51 onda skoči na 100, jump if not equal
  122   int 20         stop
  124   (prazan red)   da se izađe iz "A"
        g =100         da počne izvršavanje, ovo je (asemblerska) direktiva
        otkucaj slovo (A ili B ili sl.), opet otkucaj slovo, itd., na kraju otkucaj slovo Q
```

Ako u programu `debug.exe` uradimo `-?` (enter) onda ćemo dobiti obavještenje:

```
assemble    A [address]    (A 100 unošenje naredbi, A 200 unošenje podataka)
dump        D [range]    (D 200 207 prikazuje dio memorije)
go          G [=address] [addresses]    (G =100 izvršavanje programa)
name        N [pathname] [arglist]
quit        Q
register    R [register]
trace       T [=address] [value]    (T =100, T, T, T, itd. izvršavanje korak po korak)
unassemble  U [range]
write       W [address] [drive] [firstsector] [number]
```

`Windows XP` 'Start' > 'Programs' > 'Accessories' > 'Command Prompt' `debug` (enter).