Principi programiranja
LABORATORIJA – ASEMBLER (INTEL 8086/PENTIUM)

Jezik asemblera za Intelove procesore. Predviđeno je da se program propušta pomoću "debug.exe", tako da se koristi command prompt računara.

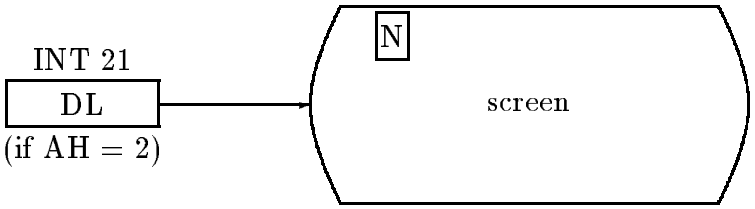
(1) Arutmetika. Napisati program u kome se računaju vrijednosti $y_1 = x_1 + x_2 + x_3 - x_4 - 4$ i $y_2 = x_1 - x_2 - 2x_3 - 2x_4 - 4$, gdje su $x_1, \ldots, x_4$ date veličine. Npr. $x_1 = 100$, $x_2 = 10$, $x_3 = 3$, $x_4 = 1$. Prikazati na ekranu slova korespondirana (po ASCII) vrijednostima y_1, y_2.

Prikazati na ekranu sadržaj onog dijela memorije u kome se nalaze $x_1, \ldots, x_4, y_1, y_2$.

`C:\windows>`debug	debug.exe
-A 100	assemble
mov ax,[200]	$AX \leftarrow x_1$
add ax,[202]	$AX \leftarrow AX + x_2$
add ax,[204]	$AX \leftarrow AX + x_3$
sub ax,[206]	$AX \leftarrow AX - x_4$
sub ax,4	$AX \leftarrow AX - 4$
mov [208],ax	$y_1 \leftarrow AX$
mov ax,[200]	$AX \leftarrow x_1$
sub ax,[202]	$AX \leftarrow AX - x_2$
sub ax,[204]	$AX \leftarrow AX - x_3$
sub ax,[204]	$AX \leftarrow AX - x_3$
sub ax,[206]	$AX \leftarrow AX - x_4$
sub ax,[206]	$AX \leftarrow AX - x_4$
sub ax,4	$AX \leftarrow AX - 4$
mov [20a],ax	$y_2 \leftarrow AX$
mov ah,2	$AH \leftarrow 2$ (function)
mov dx,[208]	$DX \leftarrow y_1$ (character)
int 21	interrupt #21
mov dx,[20a]	$DX \leftarrow y_2$ (character)
int 21	interrupt #21
int 20	interrupt #20, stop
prazna linija	end of assemble
-A 200	assemble
200 dw 64	$x_1 = 64$ hex
202 dw A	$x_2 = A$ hex
204 dw 3	$x_3 = 3$
206 dw 1	$x_4 = 1$
prazna linija	end of assemble
-G =100	go
1N	
Program terminated normally	
-D 200 20b	dump
64 00 0A 00 03 00 01 00-6C 00 4E 00	

`-Q`	quit
`C:\windows>`	command prompt

Prema tome, y_1 = l ASCII = 6C hex = 108, y_2 = N ASCII = 4E hex = 78.
Adrese: y_1 na 208, y_2 na 20a. Bolje rečeno, y_1 208–209, y_2 20a–20b.



(2) Aritmetika: $y_1 = 2x_1 + x_2 + 1$, $y_2 = 4x_3 + x_4 + 1$, npr. $x_1 = 18$, $x_2 = 19$, $x_3 = 20$, $x_4 = 21$.

`C:\windows>`debug	debug.exe
-A 100	assemble
mov ax,[200]	$AX \leftarrow x_1$
shl ax,1	$AX \leftarrow 2 \cdot AX$
add ax,[202]	$AX \leftarrow AX + x_2$
inc ax	$AX \leftarrow AX + 1$
mov [208],ax	$y_1 \leftarrow AX$
mov bx,[204]	$BX \leftarrow x_3$
shl bx,1	$BX \leftarrow 2 \cdot BX$
shl bx,1	$BX \leftarrow 2 \cdot BX$
add bx,[208]	$BX \leftarrow BX + x_4$
inc bx	$BX \leftarrow BX + 1$
mov [20a],bx	$y_2 \leftarrow BX$
mov ah,2	$AH \leftarrow 2$ (funkcija)
mov dx,[208]	$DX \leftarrow y_1$ (karakter)
int 21	prekid broj 21 (da se prikaže DL)
mov dx,[20a]	$DX \leftarrow y_2$ (karakter)
int 21	prekid broj 21 (da se prikaže DL)
int 20	prekid broj 20 (stop the computer)
prazna linija	end of assemble
-A 200	assemble
200 dw 12	$x_1 = 12$ hex (define word)
202 dw 13	$x_2 = 13$ hex (define word)
204 dw 14	$x_3 = 14$ hex (define word)
206 dw 15	$x_4 = 15$ hex (define word)
prazna linija	end of assemble
-G =100	go

```
8f
Program terminated normally
```

-D 200 20b	dump

```
12 00 13 00 14 00 15 00-38 00 66 00
```

-Q	quit
C:\windows>	command.prompt

Znači, $y_1 = 8$ ASCII $= 38$ hex $= 56$, $y_2 =$ f ASCII $= 66$ hex $= 102$.
Adrese: y_1 na 208, y_2 na 20a. Bolje rečeno, y_1 208–209, y_2 20a–20b.
Znamo da je $AX = (AH, AL)$, $\boxed{AX} = \boxed{AH}\boxed{AL}$ i slično BX, CX, DX.

(3) Množenje: $y_1 = (x_1 + 1)(x_2 + 1)$, $y_2 = (x_1 + x_2 + 1)(x_1 + x_2 - 1)$, npr. $x_1 = 5$, $x_2 = 6$.

C:\windows>debug	debug.exe
-A 100	assemble
mov ax,[150]	$AX \leftarrow x_1$
inc ax	$AX \leftarrow AX + 1$
mov bx,[152]	$BX \leftarrow x_2$
inc bx	$BX \leftarrow BX + 1$
mul bx	$(DX, AX) \leftarrow AX \cdot BX$
mov [154],ax	$y_1 \leftarrow AX$
mov ax,[150]	$AX \leftarrow x_1$
add ax,[152]	$AX \leftarrow AX + x_2$
inc ax	$AX \leftarrow AX + 1$
mov bx,[150]	$BX \leftarrow x_1$
add bx,[152]	$BX \leftarrow BX + x_2$
dec bx	$BX \leftarrow BX - 1$
mul bx	$(DX, AX) \leftarrow AX \cdot BX$
mov [156],ax	$y_2 \leftarrow AX$
mov ah,2	$AH \leftarrow 2$
mov dl,[154]	$DL \leftarrow y_1$
int 21	prikazati jedno slovo
mov dl,[156]	$DL \leftarrow y_2$
int 21	prikazati jedno slovo
int 20	stop the computer
prazna linija	end of assemble
-A 150	assemble
150 dw 5	$x_1 = 5$ (two bytes)
152 dw 6	$x_2 = 6$ (two bytes)
prazna linija	end of assemble
-G =100	go
*x	
Program terminated normally	
-D 150 157	dump
05 00 06 00 2A 00 78 00	
-Q	quit
C:\windows>	command prompt

Vidimo da je $y_1 = *$ ASCII $= 2$A hex $= 42$, $y_2 =$ x ASCII $= 78$ hex $= 120$.
Adrese: y_1 na 154, y_2 na 156. Bolje rečeno, y_1 154–155, y_2 156–157.

Množenje $y = x_1 x_2$ ($x_1, x_2 \geq 0$): MUL reg8 $AX \leftarrow AL \cdot reg8$, ili MUL reg16 $(DX, AX) \leftarrow AX \cdot reg16$. Na primjer, ako piše MUL CL onda to znači $AX \leftarrow AL \cdot CL$. Slično, IMUL ($x_1, x_2 \in Z$).

(4) Stek. Napisati program u kome se prvo u stek upisuju vrijednosti $x_1 = 7$, $x_2 = 8$, $x_3 = 9$ i $x_4 = 10$, a zatim se (čitanjem iz steka) računa veličina $y = -x_1 - 2x_2 - 3x_3 - 4x_4$. Ostaviti rezultat y u memoriji, na adresi 200 (na adresi 200–201, dva bajta).

Prikazati na ekranu sadržaj onog dijela memorije u kome se nalazi y.

`C:\windows>`debug	debug.exe
-A 100	assemble
mov ax,7	$AX \leftarrow 7$
push ax	iz AX na stek
mov ax,8	$AX \leftarrow 8$
push ax	iz AX na stek
mov ax,9	$AX \leftarrow 9$
push ax	iz AX na stek
mov ax,A	$AX \leftarrow A$
push ax	iz AX na stek
xor ax,ax	$AX \leftarrow 0$
pop bx	iz steka u BX
add ax,bx	$AX \leftarrow AX + BX$
add ax,bx	$AX \leftarrow AX + BX$
add ax,bx	$AX \leftarrow AX + BX$
add ax,bx	$AX \leftarrow AX + BX$
pop bx	iz steka u BX
add ax,bx	$AX \leftarrow AX + BX$
add ax,bx	$AX \leftarrow AX + BX$
add ax,bx	$AX \leftarrow AX + BX$
pop bx	iz steka u BX
add ax,bx	$AX \leftarrow AX + BX$
add ax,bx	$AX \leftarrow AX + BX$
pop bx	iz steka u BX
add ax,bx	$AX \leftarrow AX + BX$
neg ax	$AX \leftarrow -AX$
mov [200],ax	$c(200) \leftarrow AX$ (contents)
int 20	stop the computer
prazna linija	end of assemble
-G =100	go
`Program terminated normally`	
-D 200 201	dump
`A6 FF`	
-Q	quit
`C:\windows>`	command prompt

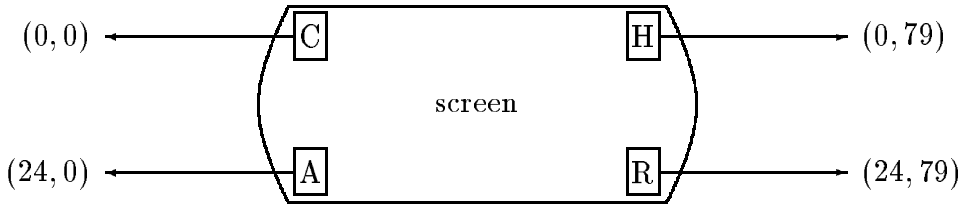
Prema tome, y = FFA6 hex $= -90$. Adrese: y na 200–201. Ako $x \in Z$ onda two's complement.

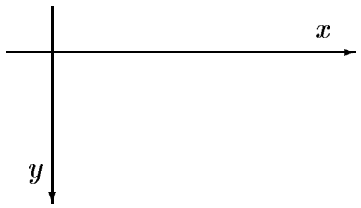
(5) Stek. Napisati program u kome se prvo u stek upisuju redom vrijednosti $x_1 = 11$, $x_2 = 12$, $x_3 = 13$ i $x_4 = 14$, a zatim se (čitanjem iz steka) računa veličina $y = -x_1^2 - x_2^2$. Ostaviti rezultat y u memoriji, na adresi 200 (na adresi 200–201, dva bajta).

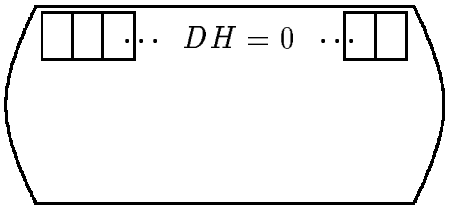
`C:\windows>`debug	debug.exe
-A 100	assemble
mov ax,B	$AX \leftarrow B$
push ax	iz AX na stek
mov ax,C	$AX \leftarrow C$
push ax	iz AX na stek
mov ax,D	$AX \leftarrow D$
push ax	iz AX na stek
mov ax,E	$AX \leftarrow E$
push ax	iz AX na stek
xor bx,bx	$BX \leftarrow 0$
pop ax	x_4 iz steka u AX
pop ax	x_3 iz steka u AX
pop ax	x_2 iz steka u AX
mov cx,ax	$CX \leftarrow AX$
mul cx	$AX \leftarrow AX \cdot CX$ (da tako kažemo)
add bx,ax	$BX \leftarrow BX + AX$
pop ax	x_1 iz steka u AX
mov cx,ax	$CX \leftarrow AX$
mul cx	$AX \leftarrow AX \cdot CX$ (da tako kažemo)
add bx,ax	$BX \leftarrow BX + AX$
neg bx	$BX \leftarrow -BX$
mov [200],bx	$c(200) \leftarrow BX$
int 20	stop the computer
prazna linija	end of assemble
-G =100	go
`Program terminated normally`	
-D 200 201	dump
`F7 FE`	
-Q	quit
`C:\windows>`	command prompt

Prema tome, y = FEF7 hex $= -265$. Adrese: y na 200–201.

(6) Crtanje pravougaonika. Napisati program u kome se učitavaju dva broja p, q ($1 \leq p, q \leq 9$). Na ekranu treba da bude prikazana granica pravougaonika koji se proteže po visini od 0 do m, a po širini od 0 do n ($m = 4p$, $n = 4q$); u gornjem lijevom uglu ekrana.








```
C:\windows>debug    debug.exe
-A 100              assemble
100 MOV AH,1        (function, INT 21)
102 INT 21          (read character, 'p')
104 SUB AL,30       (AL ← AL − 30)
106 SHL AL,1        (AL ← 2 · AL)
108 SHL AL,1        (AL ← 2 · AL)
10A MOV [201],AL    (m = 4p)
10D INT 21          (read character, 'q')
10F SUB AL,30       (AL ← AL − 30)
111 SHL AL,1        (AL ← 2 · AL)
113 SHL AL,1        (AL ← 2 · AL)
115 MOV [202],AL    (n = 4q)
118 MOV AL,58       (AL ← 58)
11A MOV [200],AL    (character X)
11D XOR BH,BH       (BH = 0, video page)
11F MOV AH,2        (function, INT 10, 21)
121 MOV CL,[202]    (CL ← c(202))
125 INC CL          (counter = n + 1)
127 XOR DH,DH       (y = 0)
129 XOR DL,DL       (x = 0)
12B INT 10          (position)
12D MOV DL,[200]    (DL ← c(200))
131 INT 21          (write character)
133 LOOP 131        (loop)
135 MOV CL,[202]    (CL ← c(202))
139 INC CL          (counter = n + 1)
13B MOV DH,[201]    (y = m)
13F XOR DL,DL       (x = 0)
141 INT 10          (position)
143 MOV DL,[200]    (DL ← c(200))
147 INT 21          (write character)
149 LOOP 147        (loop)
14B MOV CL,[201]    (CL ← c(201))
14F INC CL          (counter = m + 1)
151 XOR DH,DH       (y = 0)
153 XOR DL,DL       (x = 0)
155 INT 10          (position)
157 XCHG DL,[200]   (exchange)
15B INT 21          (write character)
```

- Učitavanje visine p i računanje m.
  ASCII brojevi su $(30)_{16}$ za '0', $(31)_{16}$ za '1', ..., $(39)_{16}$ za '9'.
  Zato, AL ← AL − 30.
  Zatim, AL ← 4 · AL.

- Učitavanje širine q i računanje n.
  Slično, karakter 'q' → cifra q → n ($n = 4q$).

- Priprema 'X', AH, BH.
  ASCII broj je $(58)_{16}$ za 'X'.

- Crtanje gornje hor. linije.
  Postavljamo krajnje gore lijevo.
  Zatim redom štampamo 'X'.
  Definicija instrukcije LOOP label:
  CX ← CX − 1, if CX ≠ 0 then goto label. U našem slučaju, label = 131.

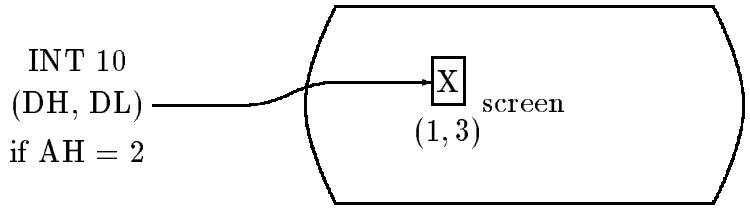
- Crtanje donje hor. linije.
  Slično, prvo na krajnju donju lijevu poziciju.

- Crtanje lijeve vert. linije.
  Kursor na krajnju gornju lijevu.
  Ponavljamo:
  {štampaj slovo 'X';
  INC DH (kursor naniže)},
  sve dok je to potrebno.

```
15D XCHG DL,[200]   (exchange)                |
161 INC DH          (y ← y + 1)               |
163 LOOP 155        (loop)                    |
165 MOV CL,[201]    (CL ← c(201))             |   • Crtanje desne vert. linije.
169 INC CL          (counter = m + 1)         |     Slično, počev od krajnje gornje desne.
16B XOR DH,DH       (y = 0)                   |
16D MOV DL,[202]    (x = n)                   |
171 INT 10          (position)                |
173 XCHG DL,[200]   (exchange)                |
177 INT 21          (write character)         |
179 XCHG DL,[200]   (exchange)                |
17D INC DH          (y ← y + 1)               |
17F LOOP 171        (loop)                    |
181 INT 20          (return to OS)            |   • Nema više.
183                 "enter"
-G =100             go
28                  unosimo p i q, po jedna cifra, bez razmaka, ne treba "enter"
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X                               X
X                               X
X                               X
X                               X
X                               X
X                               X
X                               X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Program terminated normally
-Q                  quit
C:\windows>         command prompt
```

U memoriji: na adresi 200 slovo X, na 201 m, na 202 n. Poznato je: INT 21, if $AH = 1$ then read character to AL, if $AH = 2$ then write character from DL.

Prekid INT 10, u slučaju $AH = 2$, postavlja kursor na ekranu na mjesto (visina, širina) $= (DH, DL)$; treba $BH = 0$ osnovna video stranica; BH znači broj aktivne video stranice (recimo, mogućnosti su od 0 do 7).



(7) Tabela vektora prekida, engl. Interupt Vector Table, IVT

Pojedina rutina za obradu prekida poziva se sa $INT\ n$ $(0 \leq n \leq 255)$. Kod rutine (tekst potprograma) zapisan je u memoriji u određenom području. Označimo sa a_n adresu početka

tog područja. U našem slučaju, $a_n = CS{:}IP$. Uopšte, $a = s{:}d$, $a = 16s + d$. Jedna adresa – 4 bajta (po dva za s i d).

Za pojedinu adresu a_n kaže se da predstavlja jedan vektor prekida ili da predstavlja jedan element tabele vektora prekida. Naime, sve adrese a_n zajedno i čine tabelu vektora prekida. Jasno, pogodno je da se te adrese drže jedna pored druge. One se drže u određenoj oblasti u memoriji. U tabeli ima 256 elemenata, pa $256 \times 4B = 1KB$.

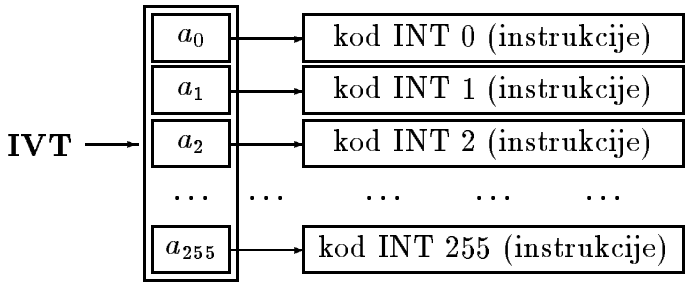
Za oblast IVT namijenjene su najniže adrese u memoriji, od $a = 0$ do $a = 3FF$, i to prva 4 bajta za a_0, itd. Ako je upisano $b_1b_2b_3b_4$ (4 bajta) onda je $a = CS{:}IP$, $CS = (b_4b_3)_{16}$, $IP = (b_2b_1)_{16}$. Npr. $b_1 = 9E$, dvije hex cifre.

U ovoj vježbi, želimo da pročitamo tabelu vektora prekida (sadržaj IVT). Umjesto svih vektora $a_0, \ldots, a_{255}$, izvući ćemo iz memorije samo $a_0, \ldots, a_{31}$.

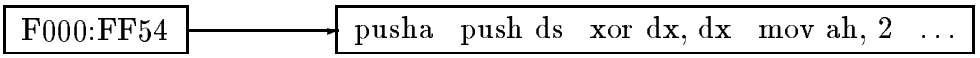
```
C:\windows>debug
-D 0:0
0000:0000  9E 0F C9 00 65 04 70 00-16 00 0C 08 65 04 70 00
0000:0010  65 04 70 00 54 FF 00 F0-CC 93 00 F0 D0 9B 00 F0
0000:0020  00 00 CC 09 28 00 0C 08-D0 9B 00 F0 D0 9B 00 F0
0000:0030  D0 9B 00 F0 D0 9B 00 F0-9A 00 0C 08 65 04 70 00
0000:0040  13 00 F9 09 4D F8 00 F0-41 F8 00 F0 A7 24 67 FD
0000:0050  39 E7 00 F0 3A 05 99 02-2D 04 70 00 B7 20 A6 FD
0000:0060  60 28 00 F0 2F 00 CD 08-6E FE 00 F0 04 06 EF 05
0000:0070  1D 00 CC 09 A4 F0 00 F0-22 05 00 00 C0 3B 00 C0
-Q
C:\windows>
```

Treba `D 0:80` za iduća 32 vektora, itd.

Vidimo da je $a_0 = 00C9{:}0F9E$, $a_1 = 0070{:}0465$, itd.



(8) Kako glasi tekst / kod jedne rutine za obradu prekida?



U ovoj vježbi, mi ćemo izvući iz memorije kod jedne ISR, Interrupt Service Routine. Opredijelili smo se za INT 5. Iz prethodne vježbe vidimo da je $a_5 = F000{:}FF54$.

Pomoću `U F000:FF54`, zatim `U`, itd. `U` (iz "debug"); U znači unassemble.

```
adresa    instrukcija

FF54      PUSHA
FF55      PUSH DS
FF56      XOR DX,DX
FF58      MOV AH,02
FF5A      INT 17
FF5C      MOV BX,0040
FF5F      MOV DS,BX
FF61      TEST AH,80
FF64      JNZ FF6D
FF66      MOV BYTE PTR [0100],FF
FF6B      JMP FFBB
FF6D      MOV AL,01
FF6F      XCHG AL,[0100]
FF73      CMP AL,01
FF75      JZ FFBB
FF77      STI
FF78      CALL F799
FF7B      MOV AH,0F
FF7D      INT 10
FF7F      MOV CH,AH
FF81      PUSH CX
FF82      MOV AH,03
FF84      INT 10
FF86      POP CX
FF87      PUSH DX
FF88      XOR DX,DX
FF8A      MOV AH,02
FF8C      INT 10
FF8E      MOV AH,08
FF90      INT 10
FF92      CALL F7A2
FF95      JB FFAF
FF97      INC DL
FF99      CMP DL,CH
FF9B      JB FF8A
FF9D      CALL F799
FFA0      JB FFAF
FFA2      XOR DL,DL
FFA4      INC DH
FFA6      CMP DH,[0084]
FFAA      JBE FF8A
FFAC      CALL F799
FFAF      SBB CL,CL
FFB1      POP DX
FFB2      MOV AH,02
FFB4      INT 10
FFB6      CLI
FFB7      MOV [0100],CL
FFBB      POP DS
FFBC      POPA
FFBD      IRET
FFBE
```

Kada govorimo o prekidu $n = 5$, koji servis pruža rutina pod tim brojem? Drugim riječima, šta će se desiti prilikom izvršavanja instrukcije INT 5? Kaže se "Print Screen". Tekući sadržaj ekrana biće upućen štampaču i (znači) biće odštampan na papir. Pretpostavlja se da se ekran nalazi u tekstualnom režimu rada.