PRILOG: JOŠ MALO O VREMENU

Zahtev za prekid broj 8 stiže sa sistemskog časovnika na svakih 55 milisekundi. Odgovarajuća sistemska prekidna funkcija ažurira sistemsko vreme. Direktna zamena te prekidne funkcije nekom funkcijom koju bismo sami kreirali, nije preporučljiva. Međutim, u samoj prekidnoj funkciji za prekid broj 8 postoji naredba *INT 1CH* koja obezbeđuje izvršavanje funkcije pridružene prekidu broj *1CH*. Inicijalno, ta prekidna funkcija se sastoji samo iz jedne naredbe *IRET* (povratak iz prekidne funkcije).

Dotična funkcija (1CH) se može zameniti nekom korisničkom funkcijom. Pokažimo kako se to izvodi. Uobičajen niz radnji, koje treba obaviti ukoliko se vrši promena prekidne funkcije pridružene nekom prekidu, je sledeći:

- zapamtiti staru prekidnu funkciju (prekidni vektor), odnosno mesto (adresu) njenog početka, pridruženu tom prekidu;
- pridružiti novu prekidnu funkciju i
- na kraju izvršavanja programa vratiti staru prekidnu funkciju.

Za pamćenje stare prekidne funkcije koristimo DOS-ovu sistemsku funkciju (*21H*, *S(AH)=35H*). Ona u paru registara *ES* i *BX* postavlja segmentni i ofsetni deo adrese dosadašnje prekidne funkcije pridružene prekidu čiji je broj u registru *AL*. Dobijene vrednosti prenosimo u predviđene promenljive.

Postavljanje nove prekidne funkcije postižemo korišćenejm DOS-ove funkcije (*21H*, *S(AH)=25H*), tako što prethodno:

- u registrima *DS* i *DX* pripremimo segmentni i ofsetni deo adrese nove prekidne funkcije;
- u registar *AL* upišemo broj prekida kome pridružujemo novu prekidnu funkciju.

Prekidne funkcije obično pišemo u asembleru (simbolički jezik). Razlikuju se od klasičnih funkcija pisanih u asembleru po tome što svi registri moraju imati sadržaje u trenutku završetka izvršavanja, kao i u trenutku početka. Zato sadržaje svih registara, koje koristimo i menjamo u toku izvršavanja prekidne funkcije,

zapisujemo na stek. Pre povratka iz prekidne funkcije čitamo sa steka podatke koje smo zapisali na početku i pridružujemo odgovarajućim registrima. Prekidna funkcija se obavezno završava naredbom IRET (povratak iz prekidne funkcije).

Povezivanje funkcija pisanih u programskom jeziku C i funkcija pisanih u asembleru relativno se lako ostvaruje. Treba pomenuti neke elemente:

- imenu funkcije pisane u programskom jeziku C u toku prevođenja biće dodat prefiks koji se sastoji od jedne crtice (_, engl. *underscore*). Zato se imenu funkcije pisane u asembleru i pozivanoj iz jezika C dodaje prefiks koji se sastoji od jedne crtice;
- argumenti funkcija pisanih na C-u upisuju se na stek u trenutku poziva, tako što se prvo upisuje poslednji, zatim pretposlednji argument itd.

U našim primerima ne prenosimo argumente tako da se način prenošenja argumenata neće videti.

Napišimo prekidnu funkciju koja će meriti sistemsko vreme i pripremati odgovarajući ispis na monitor u gornjem levom uglu. Koristićemo četiri promenljive: *prom*, *sec*, *min* i *sat* za broj milisekundi, sekundi, minuta i sati.

Kako se prekidna funkcija izvršava na svakih 55 milisekundi, uvećavaćemo *prom* za 55. U trenutku kada vrednost promenljive *prom* postane veća od 999, umanjujemo je za 1000 i za 1 uvećavamo vrednost promenljive *sec*. Kada vrednost promenljive *sec* postane jednaka broju 60, izjednačavamo je sa 0 i istovremeno uvećavamo vrednost promenljive *min* za 1. Kada vrednost promenljive *min* postane jednaka broju 60, izjednačavamo je sa 0 i uvećavamo vrednost promenljive *sat* za 1. Konačno, kada promenljiva *sat* dobije vrednost 24, izjednačavamo je sa nulom.

U isto vreme u promenljivoj *visp* pripremamo odgovarajući ispis na monitor. To je niz koji se sastoji od:

- podataka za ispis i
- atributa koji određuju način ispisa.

I podaci i atributi su bajtovi. Za podatke to je *ASCII-kod* znaka koji treba da se ispiše. Atribut određuje način ispisa tog znaka:

Ako bajt za atribut ima sadržaj:

$$a_7a_6a_5a_4a_3a_2a_1a_0$$

u binarnom brojnom sistemu, onda:

$a_2a_1a_0$	određuju boju ispisa znaka (0 – crna, 1 – plava, 2 – zelena itd.),
a_3	određuje da li će ispisani znak biti tamnije (0) ili svetlije boje (1),
$a_6a_5a_4$	određuje boju pozadine slično kao i za boju samog znaka,
a_7	određuje da li će ispisani znak svetlucati (trepereti) (1) ili ne (0).

Mi smo svim znacima stavili atribut 1, što znači da će biti tamnoplave boje na crnoj pozadini.

Ispis postižemo direktnim upisom u *video memoriju*. Adresa početka za tekstualni režim je *0xB000:0x8000*. Mi ispisujemo na kraju prvog reda ukupno 11 znakova (*cc:mm:ss.pp*). Svaka pozicija na monitoru zauzima 2 bajta u video memoriji (1 bajt za znak koji će biti ispisan i 1 bajt za atribute). Znači, upisujemo u video-memoriju 22 bajta (znaka) od adrese *0xB000:0x808A*.

U delu koji je napisan na programskom jeziku C poziva se funkcija koja pamti staru prekidnu funkciju za prekid broj 1CH. Nakon toga poziva se funkcija koja postavlja novu prekidnu funkciju za prekid 1CH, ali pre toga očita trenutno sistemsko vreme. Posle pritiska bilo koje dirke, biće vraćena stara prekidna funkcija prekidu broj 1CH i okončaće se izvršavanje programa:

```
_datasegment public 'data'
  prom dw 0 ; broj milisekundi
  sec dw 0; broj sekundi
  min dw 0; broj minuta
  sat dw 0; broj sati
; niz za ispis u formatu cc:mm:ss.ss
  visp db '0',1,'0',1,':',1,'0',1,'0',1,':',1
       db '0',1,'0',1,'.',1,'0',1,'0',1

; registri predviđeni za zapisivanje dosadašnjeg vektora koji
; odgovara prekidu 1C
  _1c_sint dw 2 dup(?)
_data ends

_code segment word public 'code'
  assume cs:_code, ds:_data

_h1c_zap_stari proc far public
; funkcija za pamćenje stare prekidne funkcije
  push es
  push bx
  push ax
  mov ah, 35h
  mov al, 1ch
  int 21h
; očitana je adresa početka stare prekidne funkcije i nalazi se
; u paru registara es:bx
  push ds
  mov ax, seg _1c_sint
  mov ds, ax
  mov _1c_sint, bx
  mov _1c_sint+2, es
  pop ds
  pop ax
  pop bx
  pop es
  ret
_h1c_zap_stari endp

_h1c_pos_novi proc far public
  push ds
  push dx
  push cx
  push ax

; očitavamo sistemsko vreme
  mov ah, 2ch
  int 21h

  mov ax, seg prom
  mov ds, ax
```

```
; broj sati upisujemo u promenljivu sat
  mov al, ch
  cbw
  mov sat, ax
; broj minuta prebacujemo u min
  mov al, cl
  cbw
  mov min, ax


; broj sekundi prebacujemo u sec
  mov al, dh
  cbw
  mov sec, ax
; broj milisekundi upisujemo u promenljivu prom
  mov al, dl
  mov ah, 0ah
  mul ah
  mov prom, ax

; pripremamo prve dve cifre za ispis
; one odgovaraju broju sati
  mov ax, sat
  mov bl, 10
  div bl
  add ah, '0'
  add al, '0'
  mov visp[2], ah
  mov visp[0], al

; pripremamo četvrti i peti znak za ispis
; oni odgovaraju broju minuta
  mov ax, min
  mov bl, 10
  div bl
  add ah, '0'
  add al, '0'
  mov visp[8], ah
  mov visp[6], al

  mov ax, sec
  mov bl, 10
  div bl
  add ah, '0'
  add al, '0'
  mov visp[0eh], ah
  mov visp[0ch], al

  mov ax, prom
  mov bl, 10
  div bl
  xor ah, ah
  div bl
  add ah, '0'
  add al, '0'
  mov visp[14h], ah
  mov visp[12h], al

  mov dx, offset _h1c_nint
  mov ax, seg _h1c_nint
  mov ds, ax
  mov ah, 25h
  mov al, 1ch

; u paru registara ds:dx smo pripremili adresu početka funkcije
; _1c_nint i to postavljamo za novu prekidnu funkciju koja
; odgovara prekidu 1C
```

```
  int 21h
  pop ax
  pop dx
  pop cx
  pop ds
  ret
_h1c_pos_novi endp


; funkcija za vraćanje starog prekidnog vektora prekida 1C
_h1c_vr_stari proc far public
  push ds
  push dx
  push ax

; u paru registara ds:dx pripremamo stari vektor koji se
; trenutno nalazi u promenljivoj _1c_sint
  mov ax, seg _1c_sint
  mov ds, ax
  mov dx, _1c_sint
  mov ax, _1c_sint+2
  mov ds, ax
  mov ah, 25h
  mov al, 1ch
  int 21h
  pop ax
  pop dx
  pop ds
  ret
_h1c_vr_stari endp


_h1c_nint proc far public
; nova prekidna funkcija za prekid broj 1CH
  push es
  push ds
  push di
  push si
  push dx
  push cx
  push bx
  push ax

; priprema za ispis na monitor
  mov ax, 0b000h
  mov es, ax
  mov di, 0808ah
  mov cx, 16h
  mov ax, seg visp
  mov ds, ax
  mov si, offset visp

; ažuriranje vremena
; za početak se broj milisekundi uvećava za 55
  add prom, 55
  cmp prom, 1000
  jl  promm1000
; broj milisekundi je veći od 999
  sub prom, 1000
  inc sec
  cmp sec, 60
  jl  secm60
; broj sekundi je veći od 59
  sub sec, 60
  inc min
  cmp min, 60
  jl  minm60
```

```
; broj minuta je veći od 59
  sub min, 60
  inc sat
  cmp sat, 24
  jl satm24
; broj sati je veći od 23
  mov sat, 0
satm24:
  mov ax, sat
  mov bl, 10
  div bl
  add ah, '0'
  add al, '0'
  mov visp[2], ah
  mov visp[0], al
minm60:
  mov ax, min
  mov bl, 10
  div bl
  add ah, '0'
  add al, '0'
  mov visp[8], ah
  mov visp[6], al
secm60:
  mov ax, sec
  mov bl, 10
  div bl
  add ah, '0'
  add al, '0'
  mov visp[0eh], ah
  mov visp[0ch], al
promm1000:
  mov ax, prom
  mov bl, 10
  div bl
  xor ah, ah
  div bl
  add ah, '0'
  add al, '0'
  mov visp[14h], ah
  mov visp[12h], al

  rep movsb
  pop ax
  pop bx
  pop cx
  pop dx
  pop si
  pop di
  pop ds
  pop es
  iret
_h1c_nint endp
_code ends
end


# include <stdio.h>
# include <string.h>
# include <dos.h>

main () {
  h1c_zap_stari();
  h1c_pos_novi();
  getch();
  h1c_vr_stari();
}
```