

Univerzitet u Novom Sadu  
Fakultet tehničkih nauka  
Departman za energetiku, elektroniku i telekomunikacije  
Katedra za elektroniku

## **MIKROPROCESORSKA ELEKTRONIKA**

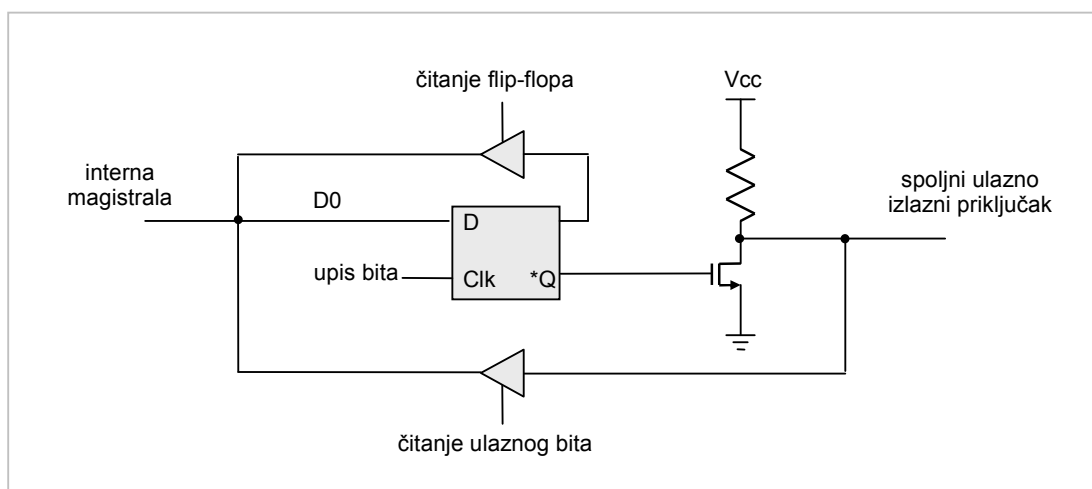
**Beleške sa predavanja  
#07**

Pripremio: prof. dr Veljko Malbaša  
Novi Sad, decembar 2007. godine

## Paralelni ulaz - izlaz

### Konfiguracija ulazno-izlaznih stepeni

Principijelna šema uzlazno-stepena jednog bita paralelnog porta mikrokontrolera 8051 prikazana je na sledećoj slici.

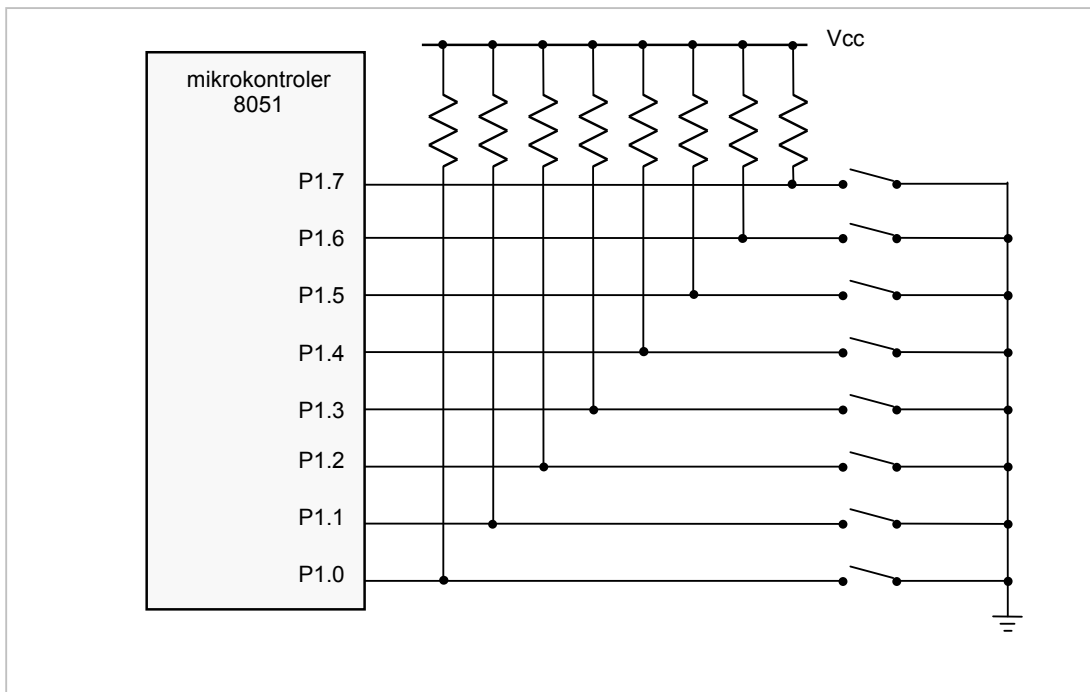


Kod operacije izlaza, sa interne magistrale izlazni bit upisuje se u D flip-flop. Kod izlaza logičke 1, izlazni tranzistor je zakočen i preko *pull-up* otpornika na izlaznom priključku dobija se logička 1. Kod izlaza logičke 0, izlazni tranzistor je provodan i na izlaznom priključku daje aktivnu logičku 0.

Kada se izlazni tranzistor zakoči, isti stepen može da se koristi za čitanje ulaznog signala sa spoljnog priključka. Aktiviranjem upravljačkog signala '*čitanje ulaznog bita*' ulazni signal sa spoljnog priključka prenosi se na internu magistralu i dalje u registar *A* mikrokontrolera. Sa druge strane, aktiviranjem signala '*čitanje flip-flopa*' stanje flip-flopa prenosi se na internu magistralu, pa tako korisnik može da pročita stanje izlaznog porta. Naravno, samo jedan od ova dva upravljačka signala može biti aktivan u bilo kom trenutku.

### Jednostavan paralelni ulaz

Paralelni port može da se jednostavno koristi za ulaz digitalnih podataka. U sledećem primeru prikazan je niz od 8 prekidača koji su priključeni na port *PI* mikrokontrolera. Prekidači mogu biti u stanju 'otvoreno' (kada galvanski odvajaju priključke) ili u stanju 'zatvoreno' (kratko spajaju priključke).



S obzirom da mikrokontroler na ulaznim priključcima na portu *P1* ima *pull-up* otpornike, spoljni otpornici mogu da se izostave.

Čitanje stanja prekidača vrši se tako što se prvo u sve bitove porta *P1* upišu logičke 1 (kako bi se zakočio izlazni transistor) i zatim pročita vrednost porta *P1*:

```
MOV    P1, #FFH      ; izlaz logičke 1 na sve bite porta P1
MOV    A, P1          ; čitanja stanja ulaznih bitova porta P1 i smeštanje u registar A
```

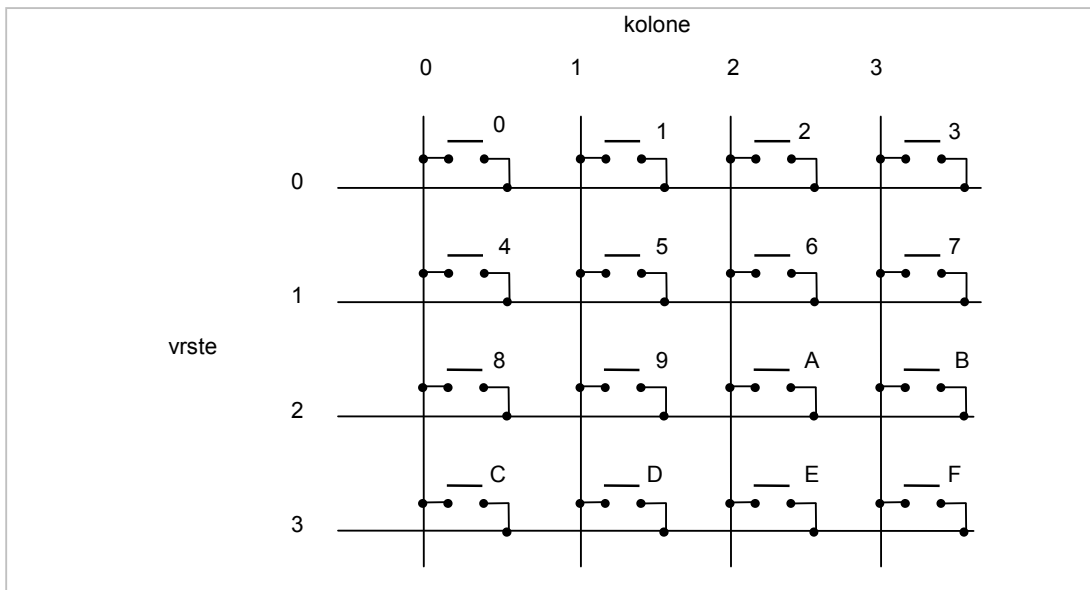
### Ulaz sa jednostavne tastature

Kod realizacije namenskih (ugrađenih) mikroračunarskih sistema često je potrebno obezbediti interfejs sa korisnikom. Ovaj interfejs obično obuhvata jednostavnu tastaturu za unos podataka i jednostavan displej za prikazivanje alfanumeričkih znakova. U ovom odeljku pokazaćemo hardverski i softverski deo rešenja za priključenje jednostavne tastature.

Neka je zadatak da se na mikrokontroler priključi tastatura sa sledećim zahtevima:

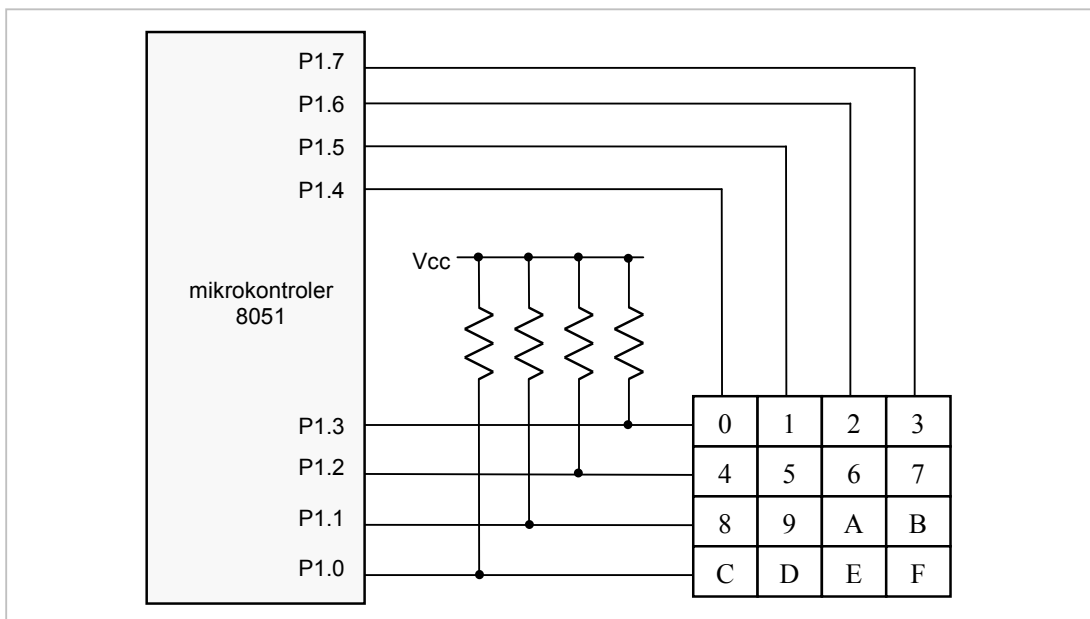
- priključiti tastaturu sa 16 tastera tako da se koristi minimalan broj ulazno/izlaznih linija mikrokontrolera,
- detektovati pritisnuti taster,
- softverski rešiti problem treperenja kontakata tastera,
- generisati ASCII kod pritisnutog tastera,
- pozivanjem potprograma na displeju prikazati znak koji odgovara pritisnutom tasteru.

Da bi smo minimizirali broj potrebnih ulazno izlaznih linija mikrokontrolera, tastere ćemo rasporediti na preseke kolona i vrsta provodne matrice sa 4 provodnika koji predstavljaju kolone i 4 provodnika koji predstavljaju vrste. Pritiskom na taster kratko se spajaju vrsta i kolona na čijem preseku je taster, kao što to pokazuje sledeća slika.



Na primer, ako se detektuje da su kratko spojene kolona 2 i vrsta 3, onda može da se zaključi da je pritisnut taster označen sa D. Pritisak na taster 3 kratko spaja kolonu 3 i vrstu 0.

Kolone matrice sa prethodne slike možemo vezati na 4 izlazne linije porta *PI*, a vrste na 4 ulazne linije porta *PI*, kao što to pokazuje sledeća slika. Na ovoj slici nisu prikazani tasteri, nego je svaki presek vrste i kolone zajedno sa tasterom prikazan kao jedan kvadrat sa upisanim znakom koji je pridružen tasteru.



Linije *PI.4* do *PI.7* porta *PI* su izlazi, a linije *PI.0* do *PI.3* porta *PI* su ulazi. Ako ni jedan taster nije pritisnut, onda je, zbog *pull-up* otpornika na vrstama visoki napon, pa se čitanjem donjih 4 bita porta *PI* dobijaju sve logičke 1.

Detekcija pritisnutog tastera vrši se na sledeći način. Prvo na izlaz *P1.4* dovedemo logičku 0, na *P1.5*, *P1.6* i *P1.7* logičke 1, a zatim pročitamo ulazne bite *P1.0* do *P1.3*. Ako je pritisnut neki od tastera u koloni koja odgovara izlazu *P1.4*, onda ćemo na odgovarajućem ulazu (*P1.0*, *P1.1*, *P1.2* ili *P1.3*) pročitati logička 0. Na primer, ako na ulazu *P1.2* pročitamo logičku 0, a na ostalim ulazima logičke 1, to znači da je pritisnut taster broj 4.

Na sličan način, ako na izlaz *P1.5* dovedemo logičku 0, a na *P1.4*, *P1.6* i *P1.7* logičke 1, čitanjem ulaza *P1.0* do *P1.3* možemo proveriti da li je pritisnut neki od tastera sa brojevima 1, 5, 9 ili D. Prema tome, aktiviranjem kolona proveravamo da li su pritisnuti sledeći tasteri:

- Kolona *P1.4* koristi se za proveru tastera 0, 4, 8 i C,
- Kolona *P1.5* koristi se za proveru tastera 1, 5, 9 i D,
- Kolona *P1.6* koristi se za proveru tastera 2, 6, A i E,
- Kolona *P1.7* koristi se za proveru tastera 3, 7, B i F.

Problem treperenja kontakata kod pritiskanja tastera može se otkloniti hardverski i softverski. Softverski pristup zasniva se na sledećem algoritmu: kada se otkrije pritisnut taster, sačekati oko 10 do 20 ms, što je dovoljan period vremena da treperenje prestane, i ponovo pročitati stanje tastera. Smatramo da je detekcija pritisnutog tastera bila ispravna ako se ponovo pročita isti taster, u suprotnom znači ta je bilo u pitanju treperenje kontakata.

Izvorni kod koji na osnovu pokazane hardverske šeme proverava koji je taster pritisnut i otklanja treperenje kontakata ima sledeći oblik.

```
tast::    MOV  P1,#EFH      ; P1.0 do P1.3 su ulazi, proveravamo kolonu P1.4

          JNB  P1.0, t_C    ; taster C pritisnut
tast_8:   JNB  P1.1, t_8    ; taster 8 pritisnut
tast_4:   JNB  P1.2, t_4    ; taster 4 pritisnut
tast_0:   JNB  P1.3, t_0    ; taster 0 pritisnut

tast_D:   MOV  P1, #DFH    ; provera kolone P1.5

          JNB  P1.0, t_D    ; taster D pritisnut
tast_9:   JNB  P1.1, t_9    ; taster 9 pritisnut
tast_5:   JNB  P1.2, t_5    ; taster 5 pritisnut
tast_1:   JNB  P1.3, t_1    ; taster 1 pritisnut

tast_E:   MOV  P1, #BFH    ; provera kolone P1.6

          JNB  P1.0, t_E    ; taster E pritisnut
tast_A:   JNB  P1.1, t_A    ; taster A pritisnut
tast_6:   JNB  P1.2, t_6    ; taster 6 pritisnut
tast_2:   JNB  P1.3, t_2    ; taster 2 pritisnut

tast_F:   MOV  P1, #7FH    ; provera kolone P1.7

          JNB  P1.0, t_F    ; taster F pritisnut
tast_B:   JNB  P1.1, t_B    ; taster B pritisnut
tast_7:   JNB  P1.2, t_7    ; taster 7 pritisnut
tast_3:   JNB  P1.3, t_3    ; taster 3 pritisnut

          LJMP tast        ; proveriti stanje tastera od početka

t_0:      LJMP trep_0       ; prelazak na proveru treperenja tastera 0
t_1:      LJMP trep_1       ; prelazak na proveru treperenja tastera 1
```

```

t_2:    LJMP trep_2
t_3:    LJMP trep_3
t_4:    LJMP trep_4
t_5:    LJMP trep_5
t_6:    LJMP trep_6
t_7:    LJMP trep_7
t_8:    LJMP trep_8
t_9:    LJMP trep_9
t_A:    LJMP trep_A
t_B:    LJMP trep_B
t_C:    LJMP trep_C
t_D:    LJMP trep_D
t_E:    LJMP trep_E
t_F:    LJMP trep_F

trep_0  LCALL kasnjenje      ; kasnjenje od 10 ms
        JB    P1.3, na_D     ; provera da li je 0 ponovo pritisnuta
        MOV   A, #0          ; ako jeste, u A stavimo 0
        LJMP  ASCII         ; odredjivanje ASCII koda
na_D:   LJMP  tast_D        ; bilo je treperenje, proveravamo sledeći taster

trep_1  LCALL kasnjenje      ; kasnjenje od 10 ms
        JB    P1.3, na_E     ; provera da li je 1 ponovo pritisnuto
        MOV   A, #1          ; ako jeste, u A stavimo 1
        LJMP  ASCII         ; odredjivanje ASCII koda
na_E:   LJMP  tast_E        ; bilo je treperenje, proveravamo sledeći taster

trep_2  LCALL kasnjenje      ; kasnjenje od 10 ms
        JB    P1.3, na_F     ; provera da li je 2 ponovo pritisnuto
        MOV   A, #2          ; ako jeste, u A stavimo 2
        LJMP  ASCII         ; odredjivanje ASCII koda
na_F:   LJMP  tast_F        ; bilo je treperenje, proveravamo sledeći taster

trep_3  LCALL kasnjenje      ; kasnjenje od 10 ms
        JNB   P1.3, ASCII3   ; provera da li je 3 ponovo pritisnuto
        LJMP  tast         ; ako nije, ponavljamo proveru tastera od početka
ASCII3: MOV   A, #3          ; ako je 3 pritisnuto, stavimo 3 u A
        LJMP  ASCII

trep_4  LCALL kasnjenje      ; kasnjenje od 10 ms
        JB    P1.2, na_0     ; provera da li je 4 ponovo pritisnuto
        MOV   A, #4          ; ako jeste, u A stavimo 4
        LJMP  ASCII         ; odredjivanje ASCII koda
na_0:   LJMP  tast_0        ; bilo je treperenje, proveravamo sledeći taster

trep_5  LCALL kasnjenje      ; kasnjenje od 10 ms
        JB    P1.2, na_1     ; provera da li je ponovo pritisnuto 5
        MOV   A, #9          ; ako jeste, u A stavimo 5
        LJMP  ASCII         ; odredjivanje ASCII koda
na_1:   LJMP  tast_1        ; bilo je treperenje, proveravamo sledeći taster

trep_6  LCALL kasnjenje      ; kasnjenje od 10 ms
        JB    P1.2, na_2     ; provera da li je 6 ponovo pritisnuto
        MOV   A, #6          ; ako jeste, u A stavimo 6
        LJMP  ASCII         ; odredjivanje ASCII koda
na_2:   LJMP  tast_2        ; bilo je treperenje, proveravamo sledeći taster

trep_7  LCALL kasnjenje      ; kasnjenje od 10 ms
        JB    P1.2, na_3     ; provera da li je ponovo pritisnuto 7

```

```

MOV A, #7 ; ako jeste, u A stavimo 7
LJMP ASCII ; odredjivanje ASCII koda
na_3: LJMP tast_3 ; bilo je treperenje, proveravamo sledeći taster

trep_8 LCALL kasnjenje ; kasnjenje od 10 ms
JB P1.1, na_4 ; provera da li je ponovo pritisnuto 8
MOV A, #8 ; ako jeste, u A stavimo 8
LJMP ASCII ; odredjivanje ASCII koda
na_4: LJMP tast_4 ; bilo je treperenje, proveravamo sledeći taster

trep_9 LCALL kasnjenje ; kasnjenje od 10 ms
JB P1.1, na_5 ; provera da li je ponovo pritisnuto 9
MOV A, #9 ; ako jeste, u A stavimo 9
LJMP ASCII ; odredjivanje ASCII koda
na_5: LJMP tast_5 ; bilo je treperenje, proveravamo sledeći taster

trep_A LCALL kasnjenje ; kasnjenje od 10 ms
JB P1.1, na_6 ; provera da li je ponovo pritisnuto A
MOV A, #E ; ako jeste, u A stavimo A
LJMP ASCII ; odredjivanje ASCII koda
na_6: LJMP tast_6 ; bilo je treperenje, proveravamo sledeći taster

trep_B LCALL kasnjenje ; kasnjenje od 10 ms
JB P1.1, na_7 ; provera da li je ponovo pritisnuto B
MOV A, #B ; ako jeste, u A stavimo B
LJMP ASCII ; odredjivanje ASCII koda
na_7: LJMP tast_7 ; bilo je treperenje, proveravamo sledeći taster

trep_C LCALL kasnjenje ; kasnjenje od 10 ms
JB P1.0, na_8 ; provera da li je ponovo pritisnuto C
MOV A, #C ; ako jeste, u A stavimo C
LJMP ASCII ; odredjivanje ASCII koda
na_8: LJMP tast_8 ; bilo je treperenje, proveravamo sledeći taster

trep_D LCALL kasnjenje ; kasnjenje od 10 ms
JB P1.0, na_9 ; provera da li je ponovo pritisnuto D
MOV A, #D ; ako jeste, u A stavimo D
LJMP ASCII ; odredjivanje ASCII koda
na_9: LJMP tast_9 ; bilo je treperenje, proveravamo sledeći taster

trep_E LCALL kasnjenje ; kasnjenje od 10 ms
JB P1.0, na_a ; provera da li je ponovo pritisnuto E
MOV A, #E ; ako jeste, u A stavimo E
LJMP ASCII ; odredjivanje ASCII koda
na_A: LJMP tast_A ; bilo je treperenje, proveravamo sledeći taster

trep_F LCALL kasnjenje ; kasnjenje od 10 ms
JB P1.0, na_B ; provera da li je ponovo pritisnuto F
MOV A, #F ; ako jeste, u A stavimo F
LJMP ASCII ; odredjivanje ASCII koda
na_B: LJMP tast_B ; bilo je treperenje, proveravamo sledeći taster

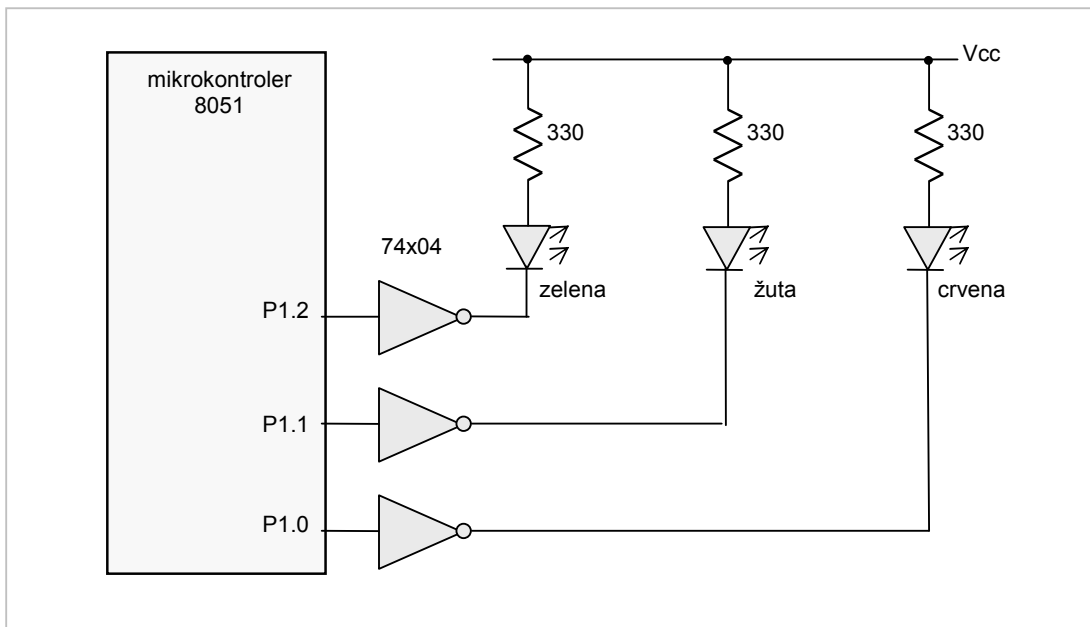
ASCII: MOV DPTR, #tabela ;
MOVC A, @A+DPTR ; uzimamo ASCII kod iz tabele
LCALL displej ; prikazivanje ASCII koda na displeju
LJMP tast ; trazenje pritisnutog tastera od pocetka

tabela: DB '0123456789ABCDEF' ; tabela sa ASCII kodovima

```

## Izlaz na svetleće diode

Paralelni port može da aktivira svetleće diode koje mogu da se koristi kao jednostavan izlaz. Sledeća šema pokazuje tri svetleće diode (zelena, žuta i crvena) koje su vezane na bitove 2, 1 i 0 porta *PI* mikrokontrolera 8051. Na izlaze paralelnog porta *PI* stavljeni su invertori 74x04 koji obezbeđuju potrebnu struju za svetleće diode.



Sledeći program napisan u simboličkom mašinskom jeziku naizmenično uključuje diode po 1 sekundu.

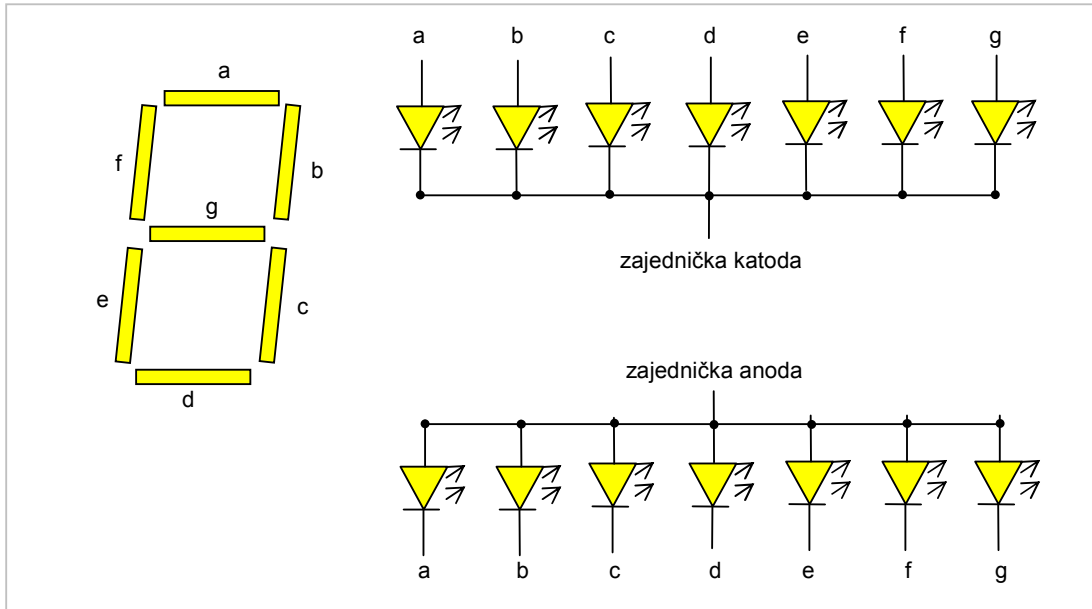
```
LED::  MOV P1,#04H      ; aktivirati zelenu diodu
       LCALL cekanje    ; držati zelenu diodu da svetli 1 s
       MOV P1,#02H      ; aktivirati žutu diodu
       LCALL cekanje    ; držati žutu diodu da svetli 1 s
       MOV P1,#01H      ; aktivirati crvenu diodu
       LCALL cekanje    ; držati crvenu diodu da svetli 1 s
       AJMP LED         ; stalno ponavljati aktiviranje dioda

cekanje: MOV R1, #200    ; spoljnu petlju ponaviti 200 puta
lab1:   MOV R0, #250     ; unutrašnju petlju ponoviti 250 puta
lab2:   PUSH A           ; petlja u kojoj se troši vreme
       POP A
       PUSH A
       POP A
       PUSH A
       POP A
       PUSH A
       POP A
       NOP
       NOP
       DJNZ R0, lab2
       DJNZ R1, lab1
       RET
```

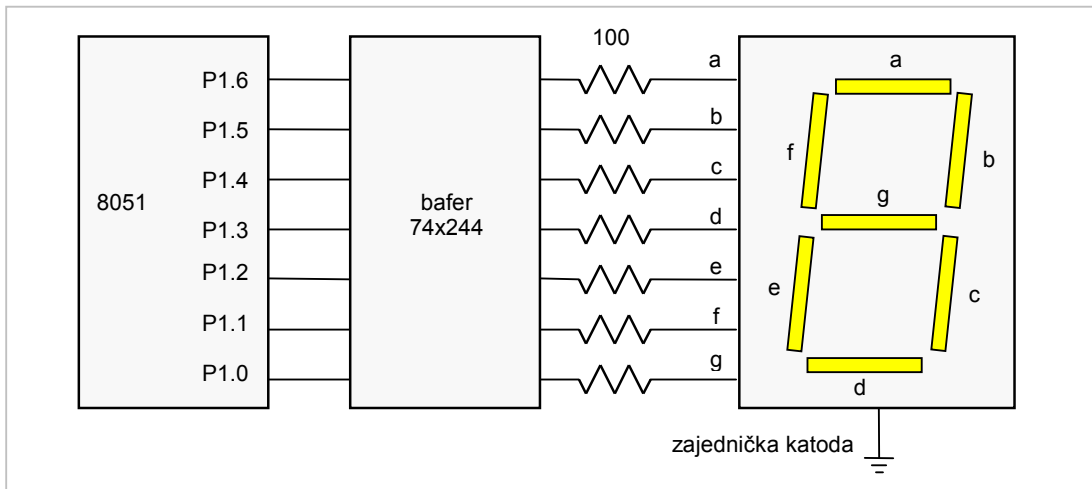


## Izlaz na 7-segmentni displej bez multipleksiranja

Displej sa sedam svetlećih dioda raspoređenih kao što je prikazano na sledećoj slici koristi se za prikazivanje cifara od 0 do 9.



Jedan 7-segmentni displej može da se priključi na paralelni port mikrokontrolera 8051 prema blok šemi prikazanoj na sledećoj slici. Jednosmerni bafer bez memorisanja signala 74x244 obezbeđuje potrebnu struju za aktiviranje segmenata od *a* do *g*.



Za aktiviranje segmenta potrebno je da odgovarajući izlazni bit porta *P1* upisati logičku 1. Sledeća tabela pokazuje kako se mogu na 7-segmentnom displeju sa zajedničkom katodom ispisati decimalne cifre od 0 do 9. Segmenti se aktiviraju logičkom jedinicom na izlaznim bitima porta *P1*. Na primer, sledeća instrukcija ispisuje cifru 3 na displeju:

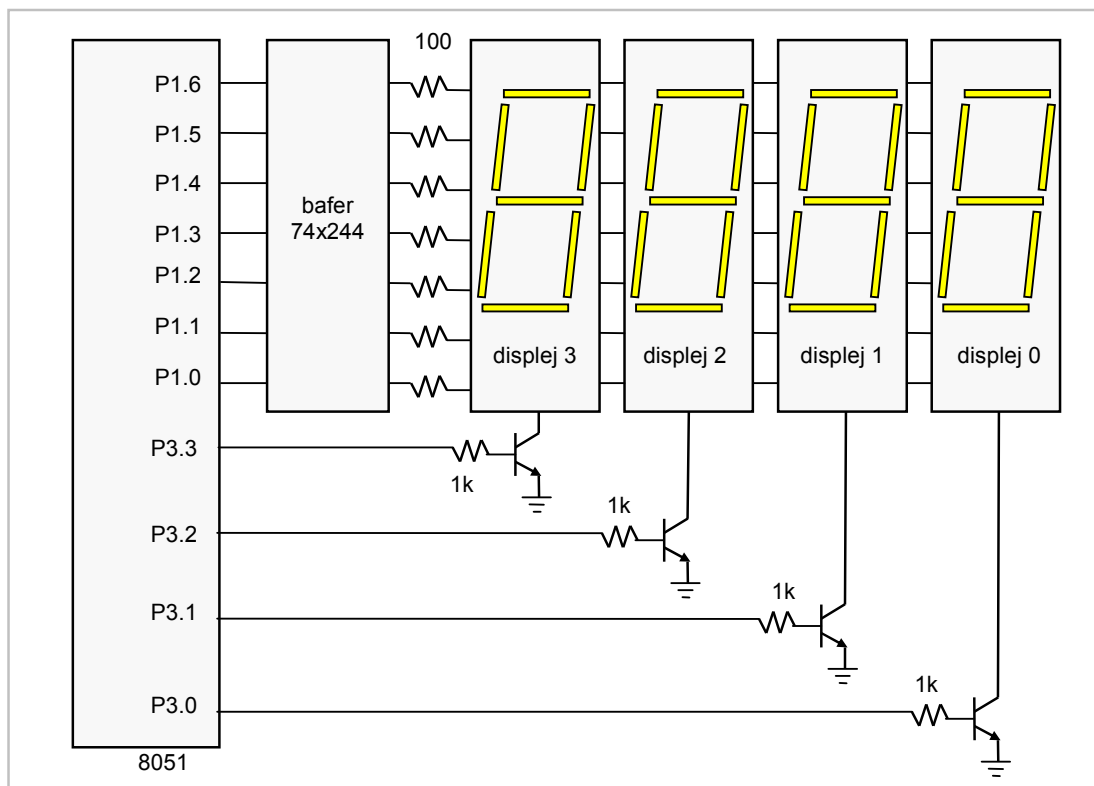
cifra	segmenti koje treba aktivirati							heksa-decimalno
	a	b	c	d	e	f	g	
0	1	1	1	1	1	1	0	7E
1	0	1	1	0	0	0	0	30
2	1	1	0	1	1	0	1	6D
3	1	1	1	1	0	0	1	79
4	0	1	1	0	0	1	1	33
5	1	0	1	1	0	1	1	5B
6	1	0	1	1	1	1	1	5F
7	1	1	1	0	0	0	0	70
8	1	1	1	1	1	1	1	7F
9	1	1	1	1	0	1	1	7B

Segmenti se aktiviraju logičkom jedinicom na izlaznim bitima porta *P1*. Na primer, sledeća instrukcija ispisuje cifru 3 na displeju:

`MOV P1,#79H` ; aktiviranje segmenata a, b, c, d i g

### Izlaz na 7-segmentne displeje sa multipleksiranjem

Multipleksiranje je pogodan način za priključenje više 7-segmentnih displeja, a da se pri tome ne koristi po jedan port za svaki displej. Sledeća blok šema pokazuje priključenje četiri 7-segmentna displeja na mikrokontroler 8051.



Displej se aktivira dovođenjem NPN tranzistora u provodno stanje, čime se zajednička katoda displeja dovodi na masu što omogućava struju kroz svetleće diode displeja. Logička 1 na odgovarajućem izlazu porta 3 prevodi tranzistor u provodno stanje. Na primer, sledeće instrukcije ispisuju cifru 4 na displeju 2, pri čemu su ostali displeji isključeni.

```
MOV P1, #33H      ; aktiviranje segmenata b, c, f i g
ANL P3, #F0H      ; isključenje displeja, gornja 4 bita P3 ostaju nepromenjeni
ORL P3, #04H      ; aktiviranje displeja broj 2
```

Naizmeničnim uključivanjem mogu se aktivirati sva četiri displeja i ispisati proizvoljna 4 decimalna broja. Na primer, za ispisivanje cifara 2, 4, 6 i 8 na displeje 3, 2, 1 i 0, respektivno, potrebno je na portove P1 i P3 dovesti vrednosti prikazane u sledećoj tabeli.

redni broj displeja	broj koji treba ispisati	port P1 (heksadecimalno)	port P2 (heksadecimalno)
3	2	6D	08
2	4	33	04
1	6	5F	02
0	8	7F	01

Sledeći program ispisuje cifre 2, 4, 6 i 8 na displeje 3, 2, 1 i 0, respektivno, tako što svaku cifru ispiše na odgovarajući displej po 2.5 ms.

```
ORG 4000H      ; početna adresa programa
MOV DPTR, #cifre ; u DPTR početna adresa tabele sa ciframa
petlja: MOV B, #0      ; brojač za petlju
sledeci: MOV A, B
MOV A, @A+DPTR ; u A je broj koji treba ispisati na displeju
MOV P1, A      ; na portu P1 je broj koji treba ispisati
INC B
MOV A, B
MOV A, @A+DPTR ; u A je informacija o displeju na koji treba ispisati broj
MOV P3, A      ; izlaz 1 na bit porta P3 koji aktivira displeju

MOV R1, #125   ; inicijalizacija petlje za kasnjenje od 2.5 ms
kas25: PUSH A
POP A
PUSH A
POP A
PUSH A
POP A
PUSH A
POP A
NOP
NOP            ; ukupno 20 masinskih ciklusa, svaki po 1 µs
DJNZ R1, kas25 ; petlja se ponavlja 125 puta
CJNE B, #8, sledeci ; ispisati sledeci broj
LJMP petlja    ; svi brojevi su ispisani, ponoviti sve od pocetka

; tabela sa heksadecimalnim brojevima koje treba ispisati na portove P1 i P3
cifre: DB 6DH, 08H, 33H, 04H, 5FH, 02H, 7F, 01H
```

## Literatura

H.W. Huang, *Using the MCS-51 Microcontroller*, Oxford University Press, 2000.