

# **Projekat iz predmeta Racunarska elektronika**

Studenti:

Milic Damir 2013/0269

Karic Jasmin 2012/0505

## Projekat 12 - Pronadji otpornik

Potrebo je realizovati program koji će studentima u lab. 39 pomoći da na laboratorijskim vezbama brze nađu vrednost otpornika. Student unosi celobrojnu vrednost koja predstavlja željenu vrednost otpornika a program treba u konzoli da iscrta četiri pravougaonika čije boje predstavljaju tu vrednost (pravougaonik koji odgovara toleranciji se izostavlja). Program treba pretragom tabele otpornosti, da nađe boje koje odgovaraju unetoj vrednosti. Ukoliko korisnik unese otpornost koja ne postoji u tabeli neophodno je obavestiti korisnika o tome a zatim naći vrednost u tabeli koja je najbliža unetoj vrednosti i za nju iscrtaati pravougaonike u odgovarajućim bojama.

### Opis rada programa

Rad programa sastoji se iz par koraka:

1. Upis otpornosti od strane korisnika
2. Transformacija unetog stringa u integer format
3. Provera unete otpornosti i prilagodjenje
4. Upis odgovarajućih vrednosti u prstenove
5. Kodiranje vrednosti prstenova
6. Iscrtavanje prstenova

### 1. Upis otpornosti od strane korisnika

Program ispisuje poruku i čeka da korisnik unese željenu otpornost. Uneta vrednost se čuva kao string od najviše 11 simbola. Ovde 9 simbola predstavlja cifre unete otpornosti, dok su poslednja dva simbola 0dh i 0ah. Ukoliko je broj unetih cifara veći od 2, ide na 2. korak, inace ukoliko uneta otpornost ima 1 ili 2 cifre, ide direktno na 4. korak. Ukoliko je uneta otpornost 0, program se gasi.

### 2. Transformacija unetog stringa u integer format

Ova obrada je neophodna radi pronalaska najblize standardne vrednosti ukoliko uneta vrednost otpornosti nije standardna. Transformacija pocinje upisom nule u promenljivu “otpornost”. Nakon toga se ide kroz petlju u kojoj se izvlaci jedan po jedan simbol iz stringa, od učitano g simbola se oduzima 48 (kako bi se od ASCII simbola dobila cifra od 0 do 9). Na kraju se dobivena cifra mnozi sa odgovarajućim stepenom 10 i ovako dobiven broj sabira sa vrednoscu promenljive “otpornost”.

$$\text{otpornost} = \sum_{i=0}^{\text{procitanoBy}-1} \text{cifra} * 10^i$$

### 3. Provera unete otpornosti i prilagodjenje

Provera unete otpornosti obavlja se tako sto program cita cifru po cifru, pocevsi od 4. cifre s leva i ukoliko naleti na cifru razlicitu od nule, to znaci da uneta otpornost nije standardna vrednost i mora se izmeniti. Izmena otpornosti se moze objasniti kroz sledeci primer:

Neka je uneta otpornost 123 050 oma

Kada program naleti na cifru 5, detektuje da je obrada neophodna. Uneti broj deli sa 1000 i mnozi sa 1000. Na taj nacin dobijamo 123 000. Na ovu vrednost se dodaje  $1000/2$ , dime dobijamo 123 500. Sada se uneta otpornost upoređuje sa ovom vrednoscu. Ukoliko je manja od ove vrednosti, najbliza standardna vrednost otpornosti je 123 000. Ukoliko je veca ili jednaka 123 500, najbliza standardna vrednost otpornosti je 124 000. Ovim smo dobili otpornost koja nam treba i data vrednost se upisuje u promenljivu "otpornost".

### 4. Upis odgovarajucih vrednosti u prstenove

Sada kad imamo standarnu vrednost otpornosti, potrebno je da upisemo cifru po cifru u promenljive prsten1, prsten2 i prsten3. U slucaju otpornosti 123 000, izvlacenje cifara i upis izgleda ovako:

$123000 - 0 * 1000000 \rightarrow 123000$

$123000 / 100000 \rightarrow 1 = \text{cifra1}$

$123000 - 1 * 100000 \rightarrow 23000$

$23000 / 10000 \rightarrow 2 = \text{cifra2}$

$23000 - 2 * 10000 \rightarrow 3000$


$3000 / 1000 \rightarrow 3 = \text{cifra3}$

"cifra4" se odredjuje ako se prosto od "ucitanoBY", sto predstavlja i broj cifara, oduzme 3, ili se upisuje nula, ukoliko otpornost ima 3 ili manje cifara.

### 5. Kodiranje vrednosti prstenova

Kodiranje vrednosti prstenova neophodno je zbog assemblera i naredbe SetTextColor. Ova naredba boji tekst i pozadinu po obrascu **text\_background \* 16 + text\_color**, pri cemu vrednosti za text\_background i text\_color uzimamo iz naredne tabele:

Color	Value
Black	0
Blue	1
Green	2
Cyan	3
Red	4
Magenta	5
Brown	6
LightGray	7
DarkGray	8
LightBlue	9
LightGreen	10
LightCyan	11
LightRed	12
LightMagenta	13
Yellow	14
White	15

Tako na primer za  $\text{text\_background} * 16 + \text{text\_color} = 4 * 16 + 1 = 65$  dobijamo 

Ovo kodiranje izvodimo za svaku od promenljivih prsten1, prsten2, prsten3 i prsten4, pomocu procesa “**kodirajBoju**”

## 6. Iscrtavanje prstenova

Prikaz prstenova ostvarujemo pomocu procesa “**prikaziPrstenove**”. On preuzima vrednost prstena i u petlji iscrtava red po red stringove str1, str2, str3 menjajuci im boje i na taj nacin vrsi samo iscrtavanje.

```
str1 byte "..CRNA..", 0
str2 byte "   ", 0
str3 byte 0dh, 0ah, 0
```

str1 je postavljen na vrednost ..CRNA.. za slucaj iscrtavanja crnog prstena. On se iscrtava kao ovaj string, crnim slovima, na tamno sivoj pozadini jer u slucaju crne pozadine sam prsten se ne bi video. str2 sluzi za dobijanje razmaka izmedju prstenova. str3 za prelazak u novi red.

Za izradu programa koriscena je Irvine32 biblioteka.

## Kod programa:

```
INCLUDE Irvine32.inc
```

```
BufSize = 11 ;//velicina buffera. Izabrano je 11 polja jer je najveca dozvoljena  
otpornost 999 Moma,  
                ;//dakle 7 cifara plus 0dh i 0ah za novi red
```

```
.data
```

```
;//stringovi za iscrtavanje boja  
str1 byte "..CRNA..", 0  
str2 byte " ", 0  
str3 byte 0dh, 0ah, 0
```

```
;//stringovi za ispisivanje odgovarajucih poruka
```

```
porUnosa LABEL BYTE  
BYTE "Unesi zeljenu vrednost otpornika (vrednost 0oma gasi program)...", 0dh, 0ah  
porUnDuz DWORD($ - porUnosa)  
ispisanoBY DWORD ?
```

```
porLosavR LABEL BYTE  
BYTE "Uneta vrednost nije standardna...", 0dh, 0ah  
porLVDuz DWORD($ - porLosavR)  
ispisBY DWORD ?
```

```
buffer BYTE BufSize DUP(? )  
procitanoBY DWORD ?
```

```
consoleHandle HANDLE 0
```

```
otpornost DWORD 0  
cifra DWORD ?  
stepen10 DWORD ?  
pomocna DWORD ?  
prsten1 WORD 0  
prsten2 WORD 0  
prsten3 WORD 0  
prsten4 WORD 0  
pom WORD 0
```

```
.code
```

```
kodirajBoju proc c uses eax ulazni : word
```

```
    mov ebx, 0  
    cmp bx, ulazni  
    jne necrna  
    mov cifra, 128  
    jmp obojeno  
necrna :
```

```
    mov ebx, 1  
    cmp bx, ulazni  
    jne nebraon  
    mov cifra, 4
```

```

        jmp obojeno
nebraon :

        mov ebx, 2
        cmp bx, ulazni
        jne necrvena
        mov cifra, 12
        jmp obojeno
necrvena :

        mov ebx, 3
        cmp bx, ulazni
        jne nenarandzasta
        mov cifra, 6
        jmp obojeno
nenarandzasta :

        mov ebx, 4
        cmp bx, ulazni
        jne nezuta
        mov cifra, 14
        jmp obojeno
nezuta :

        mov ebx, 5
        cmp bx, ulazni
        jne nezelena
        mov cifra, 2
        jmp obojeno
nezelena :

        mov ebx, 6
        cmp bx, ulazni
        jne neplava
        mov cifra, 9
        jmp obojeno
neplava :

        mov ebx, 7
        cmp bx, ulazni
        jne neljubicasta
        mov cifra, 5
        jmp obojeno
neljubicasta :

        mov ebx, 8
        cmp bx, ulazni
        jne nesiva
        mov cifra, 7
        jmp obojeno
nesiva :

        mov ebx, 9
        cmp bx, ulazni
        jne nebela
        mov cifra, 15
        jmp obojeno
nebela :

```

obojeno :

```
    ret  
kodirajBoju endp
```

```
prikaziPrstenove proc c uses eax,  
                    arg1 : word, arg2 : word, arg3 : word, arg4 : word
```

```
    xor eax, eax  
    xor ebx, ebx  
  
    mov edx, offset str3  
    call writestring  
  
    mov ecx, 5
```

oboji :

```
    mov eax, 0  
    call SetTextColor  
    mov edx, offset str2  
    call writestring  
  
    mov bx, arg1  
    mov eax, 16  
    mul ebx  
    add eax, ebx  
    call SetTextColor  
    mov edx, offset str1  
    call writestring  
  
    mov eax, 0  
    call SetTextColor  
    mov edx, offset str2  
    call writestring  
  
    mov bx, arg2  
    mov eax, 16  
    mul ebx  
    add eax, ebx  
    call SetTextColor  
    mov edx, offset str1  
    call writestring  
  
    mov eax, 0  
    call SetTextColor  
    mov edx, offset str2  
    call writestring  
  
    mov bx, arg3  
    mov eax, 16  
    mul ebx  
    add eax, ebx  
    call SetTextColor  
    mov edx, offset str1  
    call writestring
```

```

    mov eax, 0
    call SetTextColor
    mov edx, offset str2
    call writestring

    mov bx, arg4
    mov eax, 16
    mul ebx
    add eax, ebx
    call SetTextColor
    mov edx, offset str1
    call writestring

    mov edx, offset str3
    call writestring

    dec ecx
    jnz oboji

    ret
prikaziPrstenove endp

main proc

    unesi_novu_vrednost: ;// stvorena petlja kako bi korisnik mogao da unosi nove
vrednosti, bez stalnog gasenja i paljenja programa

    mov eax, 15 ;// kako bi tekst bio beo na crnoj pozadini
    call SetTextColor

    ;// pocetne vrednosti prstenova na otporniku
    mov prsten1, 0
    mov prsten2, 0
    mov prsten3, 0
    mov prsten4, 0
    ;// vrednost otpornosti i pomocne promenljive
    mov otpornost, 0
    mov pom, 0

    ;// ciscenje registara
    xor eax, eax
    xor ebx, ebx
    xor edx, edx
    xor ecx, ecx

    ;// ispisivanje poruke za unos i preuzimanje unete vrednosti
    INVOKE GetStdHandle, STD_OUTPUT_HANDLE
    mov consoleHandle, eax

    INVOKE WriteConsole, consoleHandle, ADDR porUnosa, porUnDuz, ADDR ispisanoBY, 0

    INVOKE GetStdHandle, STD_INPUT_HANDLE
    mov consoleHandle, eax

    INVOKE ReadConsole, consoleHandle, ADDR buffer,
BufSize, ADDR procitanoBY, 0

```



```

    ;// duzina procitanog stringa umanjena za 2 simbola (0dh, 0ah), daje broj cifara
    mov eax, procitanoBY
    sub eax, 2
    mov procitanoBY, eax

    ;//ako otpornost ima vise od 2 cifre, skace na obradu, ako nema, upisuje direktno
    vrednosti prstenova
    mov ebx, 2
    cmp eax, ebx
    jg vise_od_dve
    mov ebx, 1
    cmp eax, ebx
    je ima_jednu
    xor ebx, ebx
    mov bl, buffer[0]
    sub ebx, 48
    mov prsten2, bx
    mov bl, buffer[1]
    sub ebx, 48
    mov prsten3, bx
    jmp nemanula

ima_jednu:
    mov bl, buffer[0]
    sub ebx, 48
    mov eax, ebx
    jz kraj_rada ;//ukoliko je uneta vrednost otpornosti 0 oma, program se gasi
    mov prsten3, bx
    jmp nemanula
vise_od_dve:

    ;//obrada ukoliko uneta otpornost ima vise od 2 cifre
    mov ecx, procitanoBY ;//zadaje se counter
    ;// ideja ove petlje je citati cifru po cifru iz bafera i
    ;// mnoziti svaku sa odgovarajucim stepenom 10, zatim sabirati
    ;// da bi se na kraju dobila otpornost kao integer umesto stringa
petlja1:
    xor eax, eax
    xor ebx, ebx
    mov stepen10, 10
    mov ebx, procitanoBY
    sub ebx, ecx
    mov al, buffer[ebx]
    sub al, 48 ;// iz bafera je procitana prva cifra kao znak iz ASCII tabele, te je
    ;// neophodno oduzeti od procitane vrednosti 48 kako bi se dobila cifra
    mov cifra, eax ;// procitana cifra se upisuje u pomocnu promenljivu cifra
    push ecx ;// vrednost countera privremeno upisana na stack zbog promene istog
    dec ecx
    mov eax, ecx
    jz poslCif
    mov eax, 1
    ;// untrasnja petlja za racunanje stepena 10 za svaku cifru
unutrasnjaP :
    mul stepen10
    loop unutrasnjaP
    mov stepen10, eax
    jmp sracunato
poslCif:

```

```

        mov stepen10, 1
sracunato :
        pop ecx
        mov ebx, stepen10
        mov eax, cifra
        mul ebx
        add otpornost, eax
        loop petlja1

        ;//ako otpornost ima 3 cifre, skace na kraj
        mov eax, procitanoBY
        sub eax, 3
        jz kraj

        ;// ako otpornost ima vise od 3 cifre, proverava je i trazi najblizu
        ;// zamenu ukoliko uneta vrednost nije standardna

        mov ecx, procitanoBY
petlja2:
        xor eax, eax
        mov ebx, procitanoBY
        sub ebx, ecx
        add ebx, 3
        mov al, buffer[ebx];// cita cifru po cifru iz bafera(4. cifru, 5. cifru...) i
svaku uporedjuje sa nulom
        sub al, 48
        jz ispravnaCif
        ;// ukoliko neka od unetih cifara nakon prve 3 nije 0, ispisuje da uneta vrednost
nije
        ;// standardna i trazi najblizu zamenu
        INVOKE GetStdHandle, STD_OUTPUT_HANDLE
        mov consoleHandle, eax

        INVOKE WriteConsole, consoleHandle, ADDR porLosavR, porLVDuz, ADDR ispisBY, 0

        mov ecx, procitanoBY
        sub ecx, 3
        mov eax, 1
        mov ebx, 10
        ;// odredjivanje stepena 10 npr za otpornost 12300 je 10^2, isto i za 12345
stepenpetlja:
        mul ebx
        loop stepenpetlja
        mov stepen10, eax
        mov eax, otpornost
        div stepen10
        mul stepen10
        mov ebx, eax
        xor eax, eax
        xor edx, edx
        mov eax, stepen10
        shr eax, 1 ;//zamena za eax/2
        add eax, ebx ;// ovim smo nasli otpornost 12350 za slucaj unete otpornosti 12300
        ;// sada je neophodno uporediti da li je uneta otpornost veca ili manja od 12350
        ;// i time dobijamo standardnu vrednost koja je najbliza unetoj

        sub eax, otpornost
        jle manjaje

```

```

        mov otpornost, ebx
        jmp losa_vrednost
manjaje:
        add ebx, stepen10
        mov otpornost, ebx
        jmp losa_vrednost

ispravnaCif:
        mov eax, ecx
        sub eax, 4
        jz kraj
        dec ecx
        mov eax, ecx
        jnz petlja2 ;//koriscen jump umesto loop jer je broj operacija u ovoj petlji
prevelik za loop

losa_vrednost:
kraj:

        ;// u ovom trenutku imamo standardnu vrednost otpornosti bilo da je takvu uneo
        ;// korisnik ili je program morao da je prilagodi

        ;// sada upisuje vrednost prve 3 cifre promenljive otpornik u promenljive
        ;// prsten1, prsten2, prsten3
        mov ecx, 3
        mov eax, otpornost
        mov pomocna, eax
loopPrstena:
        ;// prvo izvlaci cifru
        mov eax, 3
        sub eax, ecx
        jnz NPCifra
        mov cifra, 0
NPCifra:
        mov eax, procitanoBY
        mov ebx, 3
        sub ebx, ecx
        sub eax, ebx
        push ecx
        mov ecx, eax
        mov eax, 1
        mov ebx, 10
loopUnutr:
        mul ebx
        loop loopUnutr
        pop ecx
        mov stepen10, eax
        mul cifra
        mov ebx, pomocna
        sub ebx, eax
        mov pomocna, ebx
        mov eax, stepen10
        mov esi, 10
        div esi
        mov esi, eax
        mov eax, ebx
        mov ebx, esi
        div ebx

```

```

    mov cifra, eax

    ;//sad upisuje cifru u prsten1, prsten2 ili prsten3
    mov ebx, 3
    sub ebx, ecx
    jnz nije_prva
    mov prsten1, ax
nije_prva:
    mov ebx, 2
    sub ebx, ecx
    jnz nije_druga
    mov prsten2, ax
nije_druga:
    mov prsten3, ax

    dec ecx
    jnz loopPrstena

    ;//prsten4 se odredjuje zasebno
    mov eax, procitanoBY
    sub eax, 3
    jng nemanula
    mov prsten4, ax
nemanula:

    ;// kodiranje boja zbog asemblera po tabeli datoj u izvestaju
    push prsten1
    call kodirajBoju ;// poziv procesa za kodiranje
    add esp, 2 ;// ciscenje stacka
    mov eax, cifra
    mov prsten1, ax

    push prsten2
    call kodirajBoju
    add esp, 2
    mov eax, cifra
    mov prsten2, ax

    push prsten3
    call kodirajBoju
    add esp, 2
    mov eax, cifra
    mov prsten3, ax

    push prsten4
    call kodirajBoju
    add esp, 2
    mov eax, cifra
    mov prsten4, ax

    ;// vrednosti prstenova upisujemo na stack radi poziva procesa za iscrtavanje
prstenova
    push pom
    push prsten4
    push pom
    push prsten3
    push pom
    push prsten2

```

```
    push pom
    push prsten1

    call prikaziPrstenove ;// poziv procesa za iscrtavanje
    add esp, 4 * 4 ;// ciscenje stacka

    jmp unesi_novu_vrednost

kraj_rada:

    invoke ExitProcess,0
main endp
end main
```