

UNIVERZITET U BEOGRADU ELEKTROTEHNIČKI FAKULTET

Katedra za elektroniku

Predmet: Računarska elektronika



Projekat: Vigenèr-ovo šifrovanje, zadatak 6

Asistent:
Aleksandra Lekić

Projekat radile:

Ime	Prezime	broj indeksa
Jelena	Šehovac	337/2013
Milena	Radomirović	217/2013

Zadatak:

Napisati program koji kodira tekst koji se kuca na tastaturi Vigenèr-ovom šifrom. Prilikom kucanja poruke na standardnom izlazu se prikazuje šifrovana poruka. Prilikom pokretanja teksta se pita prvo za ključnu reč. Ključna reč treba da bude pisana velikim slovima, kao i sam tekst.

Nakon toga se po kucanju teksta kodira reč po reč i ispisuje na standardni izlaz.

Primer kodiranja je:

Otvoreni tekst: **ATTACKATDAWN**

Ključ: **LEMONLEMONLE**

Šifra: **LXFOPVEFRNHR**

Naime, otvoreni tekst se sabira po modulu 26 sa ključem. Ukoliko je ključ duži od teksta, ključ se ponavlja koliko je puta potrebno, kao u primeru iznad. Formula za računanje vrednosti kodirane reči je:

$$S_i \equiv O_i + K_i \pmod{26},$$

gde je O_i i-to slovo reči, K_i ključa, a S_i slovo kodirane reči.

Izrada:

Pri izradi projekta koristile smo procedure i sve dostupne registre.

U varijanti kada je uneti tekst duži od ključa radi uštede memorije, nakon poslednjeg slova u ključu pokazivač na ključnu reč smo vraćale na početak reči, umesto da vršimo konkatenciju ključne reči, što smatramo prednošću naše realizacije.

Prema tekstu zadatka, program ispisuje stringove koji sugerišu korisniku da unese ključnu reč i tekst koji želi da kodira, a zatim ispisuje rezultat.

U samom kodu pored instrukcija pisale smo komentare kojima objašnjavamo logiku koja stoji iza samog koda zadatka.

U nastavku se nalaze slike koda sa objašnjenjima:

```

1
2  INCLUDE Irvine32.inc
3
4  BufSize = 255
5
6  .data
7  buffer1 BYTE BufSize DUP(? )
8  buffer2 BYTE BufSize DUP(? )
9  ; buffer3 BYTE BufSize DUP(? )
10 tekst1 BYTE "Unesite tekst koji zelite da kodirate:", 0Ah, 0
11 tekst2 BYTE "Unesite kljucnu rec:", 0Ah, 0
12 tekst3 BYTE "Sifovani tekst:", 0Ah, 0
13 stdInHandle HANDLE ?
14
15 bytesRead1  DWORD ?
16 bytesRead2  DWORD ?
17
18 adr1 DWORD ?
19 adr2 DWORD ?
20 i DWORD 0
21 j DWORD 0
22 a BYTE ?
23
24
25 .code
26 ucitaj PROC c
27
28     INVOKE GetStdHandle, STD_INPUT_HANDLE
29     mov stdInHandle, eax
30
31     INVOKE ReadConsole, stdInHandle, ADDR buffer1,
32     BufSize, ADDR bytesRead1, 0
33
34     mov esi, OFFSET buffer1
35     mov adr1, esi; adresa pocetka teksta koji se kodira
36
37     ret
38 ucitaj ENDP
39
40 ucitaj_kljucnu_rec PROC c
41
42
43     INVOKE GetStdHandle, STD_INPUT_HANDLE
44     mov stdInHandle, eax
45
46     INVOKE ReadConsole, stdInHandle, ADDR buffer2,
47     BufSize, ADDR bytesRead2, 0
48
49     mov esi, OFFSET buffer2; esi pokazuje na pocetak kljucne reci
50     mov adr2, esi
51
52     ret
53 ucitaj_kljucnu_rec ENDP
54
55 moduo PROC c
56     xor eax, eax
57     mov al, dh      ; Broj koji delimo( zbir dva slova)
58     mov dl, a       ; a je moduo, nekad 26, nekad duzina kljuka
59
60     div dl
61
62     mov dh, ah      ; dovoljno je jednom podeliti jer je najveći broj ciji se moduo može računati 180 ('Z' = 90)
63     ; u dh se nalazi rezultat, ostatak pri deljenju sa a
64     ret

```

```

65     moduo ENDP
66
67     saberi PROC c           ; u bl se nalazi slovo na koje trenutno pokazuje pokazivac na tekst
68     and ebx, 000000FFh     ; radimo AND jer zelimo da izdvojimo samo najnizi bajt jer njega sabiramo (kod slova)
69     add ebx, [esi]         ; dodajemo slovo kljuca na koje pokazuje esi
70     and ebx, 000000FFh     ; opet AND
71     mov dh, bl             ; smestamo rezultat sabiranja u dh, jer cemo dalje menjati ebx
72
73     inc esi                ; pomeramo pokazivace na sledeci karakter
74     inc edi
75     inc i                  ; i broji do kog smo karaktera u tekstu stigli, ne racunajuci razmak
76
77     ret
78     saberi ENDP
79
80     mod_duzina_kljuca PROC c
81     mov bh, BYTE PTR bytesRead2 ; u ovoj proceduri u dh smestamo indeks slova u ključnoj reci sa kojom sabiramo
82     sub bh, 2              ; trenutno slovo u tekstu na koje pokazuje edi
83     mov a, bh
84     call moduo; rezultat je u dh
85     cmp dh, 0
86     ret
87     mod_duzina_kljuca ENDP
88
89     razmak1 PROC c
90     inc edi                ; ispisuje razmak, ali pokazivac na ključnu rec ostaje da pokazuje na isto slovo
91     inc j                  ; kao i pre razmaka(ne povecavamo esi)jer ce se to slovo kljuca sabrati po mod26
92                             ; sa prvim sledecim slovom u tekstu nakon razmaka
93     ret
94     razmak1 ENDP
95
96     main PROC
97
98     mov edx, offset tekst2
99
100    call WriteString
101    call ucitaj_kljucnu_rec
102
103    mov edx, offset tekst1
104    call WriteString
105    call ucitaj             ; ucitavamo tekst koji se kodira
106    xor edx, edx
107    mov edi, adr1           ; edi ce pokazivati na pocetak teksta koji kodiramo
108
109    mov esi, offset buffer2
110
111
112    xor ecx, ecx
113    mov ecx, bytesRead1
114    sub ecx, 2              ; bytesRead1 predstavlja duzinu teksta uvecanu za 2, zato oduzimamo i tu vrednost smestamo u brojac
115                             ; da bi se petlja ponovila onoliko puta kolika je duzina teksta
116
117    JECXZ done2             ; ako je string prazan, ide na kraj
118    xor eax, eax
119    mov eax, bytesRead2
120    sub al, 2
121    mov a, al              ; smestili smo duzinu ključne reci u a, po tom modulu cemo racunati
122
123    labela:
124    xor ebx, ebx
125    mov ebx, [edi]         ; u bl se nalazi slovo na koje trenutno pokazuje pokazivac na tekst
126    cmp bl, ' '            ; proveravamo da li je razmak, ako jeste, skacemo na labelu "razmak"
127    jnz lab3
128
129    jmp razmak
130
131    lab3:

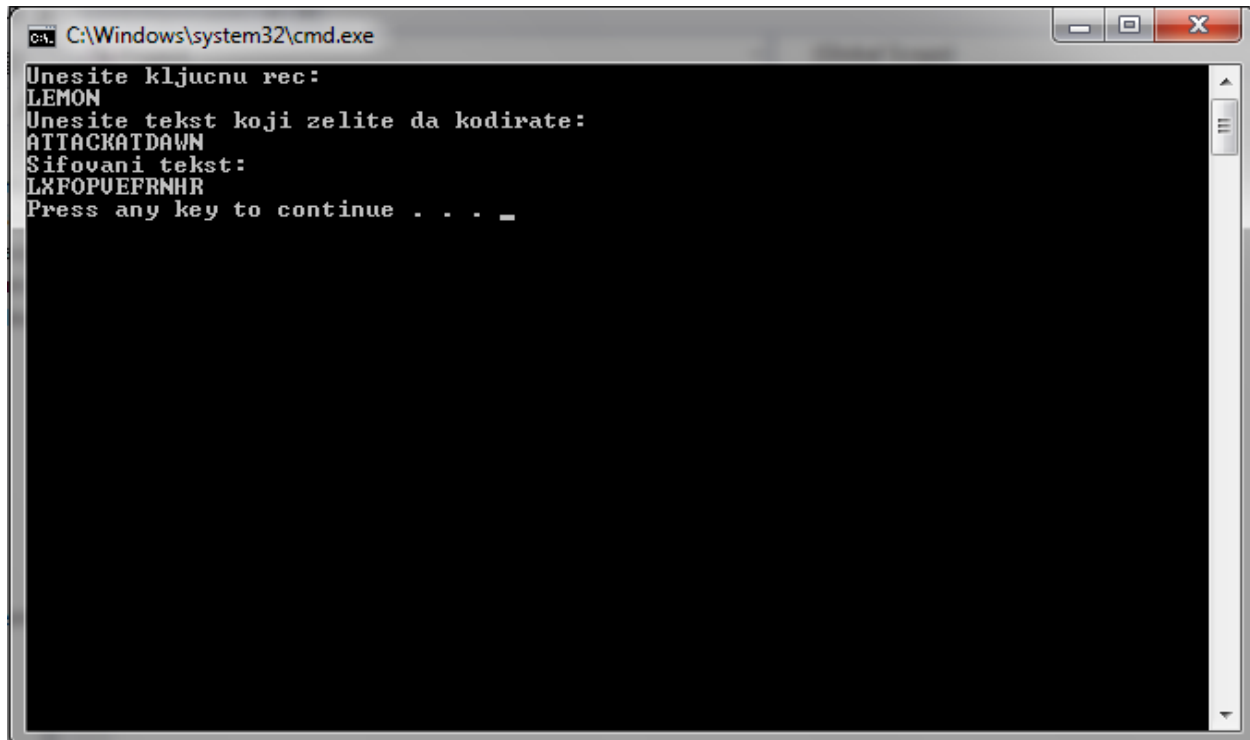
```

```

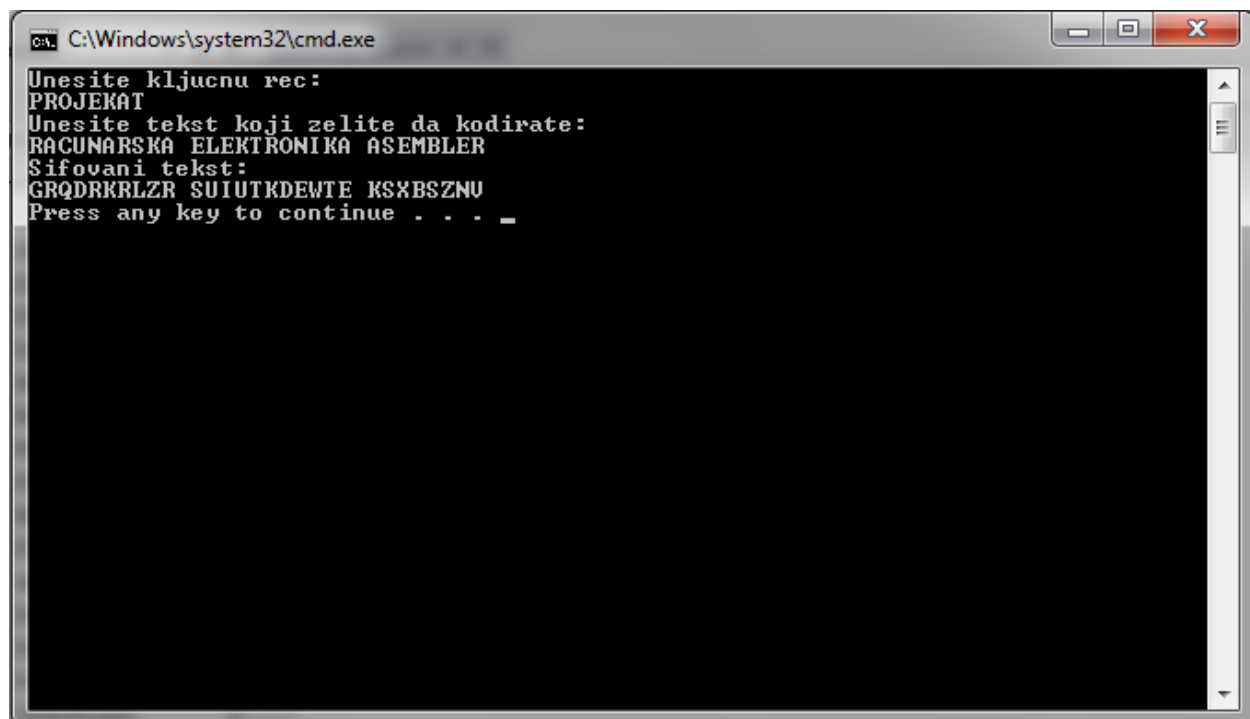
132     call saberi
133
134     mov dh, BYTE PTR i           ; da bismo izracunali i % nesto, a bl gde je smesten rezultat sabiranja za sada ne menjamo
135
136     call mod_duzina_kljuca
137     jnz mod26                   ; ako je rezultat deljenja po mod_duzina_kljuca 0, znaci da sledece slovo u tekstu
138     sub esi, (bytesRead2)       ; treba da se sabere sa prvim slovom kljuca, pa zato pokazivac na kljuc umanjujemo za duzinu kljuca
139     add esi, 2
140
141     mod26:
142     mov dh, bl                  ; vraćamo u dh rezultat sabiranja 2 slova
143
144     mov bl, 1Ah                 ; u bl smestamo 26=1Ah
145     mov a, bl                   ; u a ce se nalaziti broj po cijem modulu racunamo, dakle u ovom slucaju 26
146     call moduo
147     mov ebx, adr1               ; pri sifrovanju menjamo originalan tekst sifrovanim
148     add ebx, j                  ; j ce nam sluziti za kretanje po stringu, na pocetku je 0, a svakim prolazom kroz petlju se inkrementira
149     inc j;
150     mov al, 41h                 ; posto bi trebalo da slova A-Z imaju kod u opsegu 0-25, a ASCII kod slova A je 65, taj broj dodajemo
151     add dh, al                  ; svakom rezultatu sabiranja kako bismo dobili odgovarajuće slovo u Vignerovom kodu
152
153     mov[ebx[0]], dh             ; zamenjujemo svako slovo u tekstu odgovarajucim kodiranim
154
155     jmp lab
156
157     razmak: call razmak1
158
159     lab:
160     loop labela                 ; petlja se ponavlja onoliko puta kolika je vrednost broja smestenog u ecx
161     done2:
162
163     mov edx, offset tekst3
164     call WriteString
165     mov edx, adr1               ; MORA EDX!
166     call WriteString
167
168
169     exit
170
171     main ENDP
172     END main

```

Primeri ispravnog rada programa:



```
C:\Windows\system32\cmd.exe
Unesite kljucnu rec:
LEMON
Unesite tekst koji zelite da kodirate:
ATTACKATDAWN
Sifovani tekst:
LXFOPUFEFRNHR
Press any key to continue . . . _
```



```
C:\Windows\system32\cmd.exe
Unesite kljucnu rec:
PROJEKAT
Unesite tekst koji zelite da kodirate:
RACUNARSKA ELEKTRONIKA ASEMBLER
Sifovani tekst:
GRQDRKRLZR SUIUTKDEWTE KSXBSZNU
Press any key to continue . . . _
```