UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

Katedra za elektroniku

Predmet: Računarska elektronika

Projekat 30: Pronalaženje subliminal poruka u pesmama (Invertovanje pesme ili nekog njenog dela)

Predmetni profesor: Dr Milan Prokin
Predmetni asistent: Aleksandra Lekic

Projekat radili:

Ime	Prezime	broj indeksa
Luka	Veljović	77/2014
Ivan	Pružljanin	196/2014

1. Zadatak

Projekat 30 - otkrivanje Sublimal poruka

Napraviti program za čitanje skrivenih poruka (Subliminal messages) u pesmama. Program treba da čita stereo pesmu u WAV formatu i ispisuje je u obrnutom redosledu. Izlazni fajl je takođe u WAV formatu. Na početku programa potrebno je omogućiti da se izabere ulazni fajl, a zatim segment u pesmi koji je potrebno izvrnuti. Jedna od ponuđenih opcija je potrebno da bude i cela pesma.

**Program je rađen u assembly programskom jeziku, korišćenje alatke VisualStudio i biblioteke Irvine32..*

2. Opis koda

Program na početku traži od korisnika da unese ime fajla, koji želi da okrene. Program zatim proverava da li je zadata veličina bafera dovoljno velika da se u njega učita fajl i ako nije javlja odgovarajuću poruku korisniku. Ista veličina bafera se dodeljuje i izlaznom fajlu. Ovde je u pitanju statička alokacija memorije.

Ukoliko je program učitao fajl kako treba, sada se iz wav fajla izvlače potrebni podaci. Kako wav fajl ima dobro definisano zaglavlje ovi podaci se lako izvlače iz fajla. Najčešće je dužina zaglavja 43 bajta, a sa odgovarajućih mesta se izvlače bitni podaci (bits per sample, sample rate, data_size).

Nakon što program ima ove podatke, pita korisnika da li želi da obrne celi pesmu ili samo neki njen deo. Bez obzira šta je korsnik izabrao, program funkcioniše tako što se u odgovarajuće promenjive (ukupno_odb i ukupno_dob) upišu brojevi koji pokazuju od kog do kog bajta program treba da okrene pesmu.

Slučaj 1) Ukoliko je korisnik izabrao da obrće deo pesme, od njega se traži da unese minut i sekund od kog želi da obrene i minut i sekund do kog želi da obrne. Nakon provera da li su ove vrednosti ispravno unete, ove vrednosti se pretvaraju u odgovarajuce bajtove, pomoću podataka koje smo prvobitno izvukli.

Slučaj 2) Ukoliko je u pitanju obrtanje cele pesme program u prvu pormenljivu upisuje nulu, a u drugu(ukupno_dob) veličiniu data dela wav fajla.

Obrtanje se vrši na isti način, pomoću već navedenih promenljivih, i pomoću funkcije movsb. Destination registar pokazuje na kraj bafera, soruce registar pokazuje na početak originalnog bafera od kog treba kopirati, a u counter registru se nalazi broj bajtova koji treba kopirati. Važno je napomenuti da su razdvojeni kodovi u zavisnosti od toga da li odbirci u wav fajlu zauzimaju jedan ili dva bajta. Kopiranje se vrši unazad, tako da je lokacija na koju se kopira sledeći odbirak, pre lokacije na koju se kopirao prethodni. Na ovaj način se vrši invertovanje.

Deo koda u kome se vrši kopiranje:

```
mov esi,OFFSET buffer
add esi,header_size
add esi,ukupno_odb
inc esi
mov edi,OFFSET buffer2
add edi,header_size
inc edi
add edi,ukupno_dob
sub edi,ukupno_odb

add ecx,ukupno_dob
```

```
        sub ecx,ukupno_odb

        xor eax,eax
        mov    ax,8
        cmp    ax,bps
        jne    dvabajta
        next10:
        dec edi
        movsb
        dec edi
        loop next10
        jmp createnewfile

dvabajta:

        shr ecx,1

next1:
        dec edi
        dec edi
        movsb
        movsb
        dec edi
        dec edi
        loop next1
```

U nastavnku je kod projekta.

```
INCLUDE Irvine32.inc
INCLUDE macros.inc

BUFFER_SIZE = 2000

.data
data_size DWORD 0
header_size DWORD 0
pom1 DWORD 0; pomocna promenljiva za dohvatanje reci data
bps WORD 0
buffer BYTE BUFFER_SIZE DUP(0)
buffer2 BYTE BUFFER_SIZE DUP(0)
filename    BYTE 80 DUP(0)
pom BYTE "invertovan_"
pom2 DWORD 0
pom3 DWORD 0
samplerate DWORD 0
ukupno_odb DWORD 0;bajtovski od
ukupno_dob DWORD 0;bajtovski do
dstFilename BYTE 80 DUP(0)
fileHandle  HANDLE ?


minut DWORD 0
sekunda DWORD 0
ukupno_od DWORD 0
ukupno_do DWORD 0
pocetak DWORD 0
kraj DWORD 0
```

```
bool DWORD 0
bool1 DWORD 0
bool2 DWORD 0
bool3 DWORD 0
bool4 DWORD 0
charIn BYTE ?
celapesma BYTE 0
deopesme BYTE 0
prviput DWORD 0



.code
main PROC


	mWrite "Unesite ime fajla za obradu: "
	mov	edx,OFFSET filename
	mov	ecx,SIZEOF filename
	call	ReadString


	mov	edx,OFFSET filename
	call	OpenInputFile
	mov	fileHandle,eax


	cmp	eax,INVALID_HANDLE_VALUE; greska pri citanju fajla?
	jne	file_ok; no: preskoci
	mWrite <"fajl ne moze biti otvoren.",0dh,0ah>
	jmp	quit; zavrsi
file_ok:

  ;Upisivanje fajla

	mov	edx,OFFSET buffer
	mov	ecx,BUFFER_SIZE
	call	ReadFromFile
	jnc	check_buffer_size; greska pri citanju?
	mWrite "Greska pri citanju fajla. "; yes: prikazi poruku greske
	call	WriteWindowsMsg
	jmp	close_file

	check_buffer_size:
	cmp	eax,BUFFER_SIZE; da li je alocirani bafer dovoljan
	jb	buf_size_ok; yes
	mWrite <"Bafer je suvise mali.",0dh,0ah>
	jmp	quit; izadji

	buf_size_ok:
	mov	buffer[eax],0; upisujemo null
	mWrite "Velicina ucitanog fajla: "
	call	WriteDec; prikazi velicinu fajla
	call	Crlf


close_file:
	mov	eax,fileHandle
```

```
        call   CloseFile

;pravimo ime izlaznog fajla

        mov ecx,lengthof pom;prvo smo ispisali "invertovan_" u dstFilename
        mov esi,offset pom
        mov edi,offset dstFilename
        rep movsb

;zatim ispisujemo i ime ulaznog fajla

        mov ecx,lengthof filename
        mov esi,offset filename
        mov edi,offset dstFilename
        add edi,lengthof pom
        rep movsb

;trazimo velicinu HEDERA

        xor eax,eax
        xor ecx,ecx
        xor ebx,ebx
        mov    eax,1635017060; eax = data(ascii)
        mov pom3,eax

        mov esi,offset buffer
sledeci:
        mov ecx,4
        mov edi,offset pom2
        rep movsb
        inc ebx
        mov eax,pom2
        cmp eax,pom3
        je gotovo
        dec esi
        dec esi
        dec esi
        loop sledeci

gotovo:
        add ebx,6
        mov header_size,ebx

;trazimo velicinu data,ona pise na kraju hedera

        xor ecx,ecx
        add ecx,OFFSET buffer
        add ecx,header_size
        mov ah,[ecx]
        dec ecx
        mov al,[ecx]
        shl eax,16
        dec ecx
        mov ah,[ecx]
        dec ecx
        mov al,[ecx]
        mov data_size,eax
```

```
;trazimo velicinu sempla(1 ili 2 bajta) ona se uvek nalazi na 34. i 35. bajtu
;hedera i upisujemo je u pomocnu promenljivu bps

	xor eax,eax
	xor ecx,ecx
	add ecx,OFFSET buffer
	add ecx,35
	mov ah,[ecx]
	dec ecx
	mov al,[ecx]
	mov bps,ax

;trazimo samplerate
	xor eax,eax
	xor ecx,ecx
	add ecx,OFFSET buffer
	add ecx,25
	mov ah,[ecx]
	dec ecx
	mov al,[ecx]
	mov samplerate,eax

;pocetak interaktivnog menija

	xor edx,edx
ispocetka:
	call Clrscr
	xor eax,eax
	xor ebx,ebx


	mov edx,prviput
	cmp edx,0
	je next
	mWrite <"Izaberite jednu od ponudjenih opcija!",0dh,0ah>
	next:
	mWrite <"Zelim da obrnem: ",0dh,0ah,"1.celu pesmu",0dh,0ah,"2.deo pesme",0dh,0ah>
	mov edx,1
	mov prviput,edx
	call ReadChar
	mov bl,al
	cmp bl,"1"
	jne dalje

	mov eax,1
	mov celapesma,al
	jmp kraj1

dalje:
	cmp bl,"2"
	jne ispocetka
	mov eax,1
	mov deopesme,al
	jmp deo_pesme


deo_pesme:
	call Clrscr
```

```
        mWrite <"Unesite broj minuta i sekundi od kojeg zelite da invertujete.",0dh,0ah>


        ;Unos minuta i ispitivanje ispravnosti datog unosa

ispis_minuta:
        xor eax,eax
        mov eax,bool4
        cmp eax,1
        je ispis_do_manje_nego_od
        jmp dalje_4
        ispis_do_manje_nego_od:
        mWrite <"Krajnji trenutak mora biti veci nego pocetni! ",0dh,0ah>
        jmp ispis_minuta1

dalje_4:
        xor eax,eax
        mov eax,bool1
        cmp eax,1
        jne ispis_minuta1
        call Clrscr
        mWrite <"Broj minuta mora biti pozitivan broj i u toku trajanja pesme! Unesite
        ponovo!",0dh,0ah>
        ispis_minuta1:
        mWrite <"Minut(pocetni): ",0dh,0ah>
        xor eax,eax
        mov eax,1
        mov bool1,eax
        xor eax,eax
        call ReadInt
        mov minut,eax
        cmp eax,0
        jl ispis_minuta
        call Clrscr
        jmp ispis_sekundi1


        ;Unos sekundi i ispitivanje ispravnosti datog unosa

ispis_sekundi:
        xor eax,eax
        mov eax,bool
        cmp eax,1
        jne ispis_sekundi1
        call Clrscr
        mWrite <"Broj sekundi mora biti u intervalu [0,59]! Unesite ponovo!",0dh,0ah>
        ispis_sekundi1:
        mWrite <"Sekunda(pocetna): ",0dh,0ah>
        xor eax,eax
        mov eax,1
        mov bool,eax
        xor eax,eax
        call ReadInt
        mov sekunda,eax
        cmp eax,59
        jg ispis_sekundi
        cmp eax,0
        jl ispis_sekundi
```

```
        jmp sabiranje


sabiranje:
        xor ecx,ecx
        xor ebx,ebx
        mov ebx,sekunda
        xor eax,eax
        mov eax,minut
        mov ecx,60
        mul ecx
        add ebx,eax
        mov ukupno_od,ebx
        jmp deo_pesme1



        ;Unos krajnje tacke intervala

deo_pesme1:
        call Clrscr
        mWrite <"Unesite broj minuta i sekundi do kojeg zelite da invertujete.",0dh,0ah>


        ;Unos minuta i ispitivanje ispravnosti datog unosa

ispis_minuta3:
        xor eax,eax
        mov eax,bool2
        cmp eax,1
        jne ispis_minuta2
        call Clrscr
        mWrite <"Broj minuta mora biti pozitivan broj i u toku trajanja pesme! Unesite
        ponovo!",0dh,0ah>
        ispis_minuta2:
        mWrite <"Minut(krajnji): ",0dh,0ah>
        xor eax,eax
        mov eax,1
        mov bool2,eax
        xor eax,eax
        call ReadInt
        mov minut,eax
        cmp eax,0
        jl ispis_minuta3
        call Clrscr
        jmp ispis_sekundi2


;Unos sekundi i ispitivanje ispravnosti datog unosa

ispis_sekundi3:
        xor eax,eax
        mov eax,bool3
        cmp eax,1
        jne ispis_sekundi2
        call Clrscr
        mWrite <"Broj sekundi mora biti u intervalu [0,59]! Unesite ponovo!",0dh,0ah>
        ispis_sekundi2:
```

```
        mWrite <"Sekunda(krajnja): ",0dh,0ah>
        xor eax,eax
        mov eax,1
        mov bool3,eax
        xor eax,eax
        call ReadInt
        mov sekunda,eax
        cmp eax,59
        jg ispis_sekundi3
        cmp eax,0
        jl ispis_sekundi3
        jmp sabiranje1


sabiranje1:
        xor ecx,ecx
        xor ebx,ebx
        mov ebx,sekunda
        xor eax,eax
        mov eax,minut
        mov ecx,60
        mul ecx
        add ebx,eax
        mov ukupno_do,ebx;
        jmp kraj2
kraj2:
        call Clrscr
        mWrite <"Validan unos!",0dh,0ah>
        mWrite <"Izvrsena je konverzija dela pesme!",0dh,0ah>


        ;trazenje ukupno_odb do b

        xor eax,eax
        xor edx,edx
        mov ebx,8
        add eax,ukupno_od
        imul bps
        imul samplerate
        idiv ebx

        mov ukupno_odb,eax

        xor eax,eax
        xor edx,edx
        add eax,ukupno_do
        imul bps
        imul samplerate
        idiv ebx

        mov ukupno_dob,eax

        xor ebx,ebx
        xor eax,eax
        mov eax,ukupno_odb
        mov ebx,data_size
        cmp eax,ebx
        jg deo_pesme
```

```
	xor ebx,ebx
	xor eax,eax
	mov eax,ukupno_dob
	mov ebx,data_size
	cmp eax,ebx
	jg deo_pesme

	xor ebx,ebx
	xor eax,eax
	mov bool4,1;		promenljiva bool4 sluzi za proveru da li je ukupno_odb <
ukupno_dob
	mov eax,ukupno_dob
	mov ebx,ukupno_odb
	cmp eax,ebx
	jl deo_pesme

	jmp unos_pesme


kraj1:
	call Clrscr
	mWrite <"Izvrsena je konverzija cele pesme!",0dh,0ah>
	xor eax,eax
	mov ebx,8
	mov eax,data_size
	mov ukupno_dob,eax

unos_pesme:

	;kopiramo header fajla(prvih header_size bajta iz bafera) u buffer2

	cld
	mov ecx,header_size
	mov esi,OFFSET buffer
	mov edi,OFFSET buffer2
	rep movsb


	xor edi,edi
	xor esi,esi
	xor ecx,ecx

	cld

	;vrsimo obrnut upis semplova u deo bafera nakon hedera

	mov esi,OFFSET buffer
	add esi,header_size
	add esi,ukupno_odb
	inc esi
	mov edi,OFFSET buffer2
	add edi,header_size
	inc edi
	add edi,ukupno_dob
	sub edi,ukupno_odb

	add ecx,ukupno_dob
```

```
        sub ecx,ukupno_odb

        xor eax,eax
        mov    ax,8
        cmp    ax,bps
        jne    dvabajta
        next10:
        dec edi
        movsb
        dec edi
        loop next10
        jmp createnewfile

dvabajta:

        shr ecx,1

next1:
        dec edi
        dec edi
        movsb
        movsb
        dec edi
        dec edi
        loop next1


        createnewfile:


        ;Kreiranje izlaznog fajla

        xor edx,edx
        mov    edx,OFFSET dstFilename
        call   CreateOutputFile
        mov    fileHandle,eax


        ;Ispisivanje bafera2 u izlazni fajl

        mov    eax,fileHandle
        mov    edx,OFFSET buffer2
        mov    ecx,LENGTHOF buffer2
        call   WriteToFile


quit:
exit

main ENDP

END main
```