UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

Katedra za elektroniku

Predmet: Računarska elektronika

Projekat 29: Pravljenje stereo zvuka

Projekat radili:

Ime	Prezime	Broj indeksa
Igor	Beracka	0549/2013
Petar	Marin	0555/2014

Predmetni profesor: prof. Dr Milan Prokin
Predmetni asistent: asis. Aleksandra Lekić

Sadržaj

Tekst projektnog zadatka

Napraviti program koji od dva ulazna PCM fajla koji predstavljaju stereo zvuk jedne iste melodije pravi izlazni fajl u WAV formatu. Odbirci stereo zvuka se zapisuju tako što se upisuju redom odbirci za levi i desni zvučnik.Naime, upisuje se jedan odbirak signala za levi zvučnik, a zatim jedan odbirak za desni zvučnik i tako dok se ne upišu svi odbirci.Izlazni WAV fajl ima specijalno zaglavlje koje iznosi 44B, nakon čega slede odbirci signala.
Zaglavlje WAV falja je kao na linku:
http://www.topherlee.com/software/pcm-tut-wavformat.html
Obratiti pažnju da između 41-44 bajta u zaglavlju treba ispisati veličinu segmenta koji čine odbirci signala u bajtovima.Odbirci se predstavljaju kao 16-bitni i prepisuju direktno kao u PCM fajlu

Opis projektnog koda

- Na početku smo uključili biblioteku **irvine32.inc** i iz nje smo koristili sledeće funkcije:
  - *SetConsoleTitle* – zadaje ime konzoli
  - *GetStdHandle* – učitava "handle" za upis i ispis na konzolu
  - *WriteConsole* – ispisuje tekst na konzolu tamo gde je trenutna lokacija kursora
  - *ReadConsole* – čita tekst iz ulaznog bafera konzole i briše ga iz bafera
  - *OpenInputFile* – otvara ulazni fajl tj. stavlja njegov "handle" u akumulator
  - *CreateOutputFile* – pravi izlazni fajl
  - *WriteToFile* – ispisuje bafer u izlazni fajl
  - *ReadFromFile* – učitava bafer iz ulaznog fajla
  - *CloseFile* – zatvara fajl
  - *ExitProcess* – izlazni iz programa
- Win32API funkcije:
  - *GetFileSize* – u akumulator upisuje veličinu fajla u bajtovima (nema je u Irvine32.lib pa smo morali da je dodamo)
- U **.const** delu smo definisali konstante koje će nam trebati za zaglavlje izlaznog .wav fajla.Stringove smo pisali sa desna na levo zato što operacija mov tim redom prebacuje bajtove.U ovom delu se nalaze veličina ulaznog(256KB) i izlaznog(512KB) bafera i maksimalna dužina putanje u kojoj se nalaze ulazni/izlazni fajlovi koja iznosi 260 ASCII karaktera.
- **.data** deo sadrži inicijalizovane podatke programa kao što su veličina uzorka, broj kanala, učestanost uzimanja uzorka, zatim naslov konzole, intro tekst i razne poruke koje se ispisuju na konzolu.

- U **.data?** delu su svi ostali podaci koje nismo mogli unapred da inicijalizujemo: veličina fajla, pomoćne promenljive, "handleri", baferi i putanje ulaznih i izlaznog fajla.
- U **.code** sekciji prvo imamo proceduru populateOutBuff , a zatim glavni program:
  - **populateOutBuff** služi da ispiše uzorke ulaznih .pcm fajlova u izlazni .wav fajl.Registar ecx koristili smo kao brojač, a edx za broj bajtova koje treba ispisati u izlazni fajl, esi i ebx smo koristili kao lokacije na kojima se nalaze adrese uzoraka prvog i drugog ulaznog fajla respektivno i edi za adresu izlaznog uzorka.Najpre smo ecx i eax sačuvali na steku, inicijalizovali ecx sa nulom, zatim se upisuju bajtovi iz [esi] u akumulator, pa iz akumulatora u [edi](zato što ne može direktno iz [esi] u [edi]]), inkrementiraju se ovi registri i isto se radi i za drugi ulazni fajl.Nakon toga se inkrementira ecx i upoređuje sa vrednošću registra edx.Kada brojač dostigne vrednost registra edx procedura se prekida i sa steka se vraćaju vrednosti registara eax i ecx.
  - Prilikom startovanja ovaj program pita korisnika za putanju ulaznih fajlova i izlaznog fajla. Ukoliko je detektovana nepostojeća putanja ulaznih fajlova ili izlazni fajl ne može da se napravi tražiće od korisnika da se ponovo unese putanja. Zatim računa parametre zaglavlja i formira ga u baferu izlaznog fajla. Zatim učitava ulazne fajlove i smešta ih u ulazni bafer.Ukoliko je ulazni bafer manji od veličine fajla učitaće ga iz nekoliko puta.Svaki put kada se izlazni bafer napuni njegov sadržaj se ispiše u izlazni fajl i nakon toga se učitavaju ulazni baferi a pokazivač izlaznog bafera se vraća na početak.

Projektni kod

```
include Irvine32.inc

 GetFileSize proto, hFile: handle, lpFileSizeHigh: ptr dword;Win32 API function

.const
	riffConst EQU "FFIR" ;Marking the riff type of file
	waveConst EQU "EVAW" ;File type header mark
	formatChunkConst EQU " tmf"; Format chunk marker
	dataConst EQU "atad" ;Marking the begining of data section
	inBuffSize EQU 262144 ; 256KB
	outBuffSize EQU 524288; Min size 44 bytes, 512KB
	maxPathLen EQU 260 ;Max path length for MS-DOS, 260 ASCII characters

.data
	sampleSize dword 16 ;Sample bit size, default value 16 bit
	channels word 2; Number of channels, default value 2
	sampleRate dword 44100; Sample rate, default value 44100 (CD) 48000 (DVD)
	byteRate dword 176400; (sampleRate *sampleSize*channels)/8

	consoleTitle byte "WAVMUX",0h

	introText label byte
	byte "=================================================",0ah
	byte "|                    W A V M U X                  |",0ah
	byte "|Program that muxes two .wav          Authors: Petar Marin    |",0ah
	byte "|mono files into single .wav stereo file              Igor Beracka  |",0ah
	byte "=================================================",0ah,0ah
	introTextSize dword ($ - introText)

	message1 label byte
	byte "Enter left channel filepath: "
	message1Size dword ($ - message1)

	message2 label byte
	byte "Enter right channel filepath: "
	message2Size dword ($ - message2)

	message3 label byte
	byte "ERROR: Filepath doesn't exist!",0ah
	message3Size dword ($ - message3)
```

```
        message4 label byte
        byte "Enter output filepath: "
        message4Size dword ($ - message4)

        message5 label byte
        byte "ERROR: File can't be created",0ah
        message5Size dword ($ - message5)

        message6 label byte
        byte "Finished!",0ah,0ah
        message6Size dword ($ - message6)

.data?
        fileSize dword ?
        dataSize dword ?
        consoleInputHandle handle ?
        consoleOutputHandle handle ?
        lchFileHnd handle ?
        rchFileHnd handle ?
        outFileHnd handle ?
        lchFilepathSize dword ?
        rchFilepathSize dword ?
        outFilepathSize dword ?

        outBuff byte outBuffSize DUP(?)
        in1Buff byte inBuffSize DUP(?)
        in2Buff byte inBuffSize DUP(?)

        lchFilepath byte maxPathLen DUP(?) ;Left channel filepath
        rchFilepath byte maxPathLen DUP(?);Right channel filepath
        outFilepath byte maxPathLen DUP(?);Output filepath

.code
populateOutBuff proc
        ;ecx = local counter
        ;edx = how many bytes to move in output buffer
        push ecx
        push eax
        mov ecx, 0; initialising local counter
        lp: mov ax, [esi]
        mov [edi], ax
        add esi, 2
        add edi, 2
        mov ax, [ebx]
        mov [edi], ax
        add ebx, 2
```

```
        add edi, 2
        add ecx, 4
        cmp ecx, edx
        jne lp

        pop eax
        pop ecx
        ret
populateOutBuff endp

main proc
        invoke SetConsoleTitle, ADDR consoleTitle
        invoke GetStdHandle, STD_OUTPUT_HANDLE
        mov consoleOutputHandle, eax
        invoke GetStdHandle, STD_INPUT_HANDLE
        mov consoleInputHandle, eax
        invoke WriteConsole, consoleOutputHandle, ADDR introtext, introtextSize, 0, 0

        ;Opening left channel file
j1:     invoke WriteConsole, consoleOutputHandle, ADDR message1, message1Size, 0, 0
        invoke ReadConsole, consoleInputHandle, ADDR lchFilepath, maxPathLen, ADDR
lchFilepathSize, 0
        mov edx, offset lchFilepath
        mov ecx, lchFilepathSize
        mov esi, edx
        mov al, 0h
        ;Inserting 0h string termination char
        mov [ecx+esi-2], al
        ;Getting input file 1 handle
        call OpenInputFile
        mov lchFileHnd, eax
        cmp eax, INVALID_HANDLE_VALUE
        jne j2
        ;Error msg
        invoke WriteConsole, consoleOutputHandle, ADDR message3, message3Size, 0, 0
        jmp j1

        ;Opening right channel file
j2:     invoke WriteConsole, consoleOutputHandle, ADDR message2, message2Size, 0, 0
        invoke ReadConsole, consoleInputHandle, ADDR rchFilepath, maxPathLen, ADDR
rchFilepathSize, 0
        mov edx, offset rchFilepath
        mov ecx, rchFilepathSize
        mov esi, edx
        mov al, 0h
        ;Inserting 0h string termination char
```

```
        mov [ecx+esi-2], al
        ;Getting input file 1 handle
        call OpenInputFile
        mov rchFileHnd, eax
        cmp eax, INVALID_HANDLE_VALUE
        jne j3
        ;Error msg
        invoke WriteConsole, consoleOutputHandle, ADDR message3, message3Size, 0, 0
        jmp j2

        ;Creating output file
j3:     invoke WriteConsole, consoleOutputHandle, ADDR message4, message4Size, 0, 0
        invoke ReadConsole, consoleInputHandle, ADDR outFilepath, maxPathLen, ADDR
outFilepathSize, 0
        mov edx, offset outFilepath
        mov ecx, outFilepathSize
        mov esi, edx
        mov al, 0h
        ;Inserting 0h string termination char
        mov [ecx+esi-2], al
        mov [ecx+esi-1], al
        call CreateOutputFile
        mov outFileHnd, eax
        cmp eax, INVALID_HANDLE_VALUE
        jne j4
        ;Error msg
        invoke WriteConsole, consoleOutputHandle, ADDR message5, message5Size, 0, 0
        jmp j3

        ;Writing file header to output file buffer
j4:     mov edi, offset outBuff
        mov eax, riffConst
        mov [edi], eax
        add edi, 4;shifting buffer pointer
        mov eax, offset dataSize
        ;Calculating file size
        invoke GetFileSize, lchFileHnd, 0
        mov dataSize, eax
        invoke GetFileSize, rchFileHnd, 0
        add eax, dataSize
        mov dataSize, eax
        add eax, 36;filesize = headersize+datasize-8, headersize= 44-8 bytes, 8 bytes read already
        mov fileSize, eax
        mov [edi], eax ;writing fileSize to buffer
        add edi, 4
        mov eax, waveConst
```

```
        mov [edi], eax
        add edi, 4
        mov eax, formatChunkConst
        mov [edi], eax
        add edi, 4
        mov eax, 16;Length of format data
        mov [edi], eax; to output buffer byte 17-20
        add edi, 4
        mov ax, 1;value=1 for PCM format
        mov [edi], ax; to output buffer byte 21-22
        add edi, 2
        mov ax, channels; default value=2 channels
        mov [edi], ax
        add edi, 2
        mov eax, sampleRate; default value 44100
        mov [edi], eax
        add edi, 4
        mov eax, byteRate
        mov [edi], eax
        add edi, 4
        mov ax, 4; bitrate*channels/8
        mov [edi], ax
        add edi, 2
        mov eax, sampleSize
        mov [edi], eax
        add edi, 2
        mov eax, dataConst
        mov [edi], eax
        add edi, 4
        mov eax, dataSize
        mov [edi], eax
        add edi, 4
        ;Header is loaded in buffer
        ;Writing header to file
        mov eax, outFileHnd
        mov edx, offset outBuff
        mov ecx, 44
        call WriteToFile
        ;skipping header from input files
        invoke SetFilePointer, lchFileHnd, 44, 0, 0;moving file pointer to position 44
        invoke SetFilePointer, rchFileHnd, 44, 0, 0
        ;Reading from input buffers
ld:     mov eax, lchFileHnd
        mov edx, offset in1Buff
        mov ecx, inBuffSize
        call ReadFromFile
```

```
        mov eax, rchFileHnd
        mov edx, offset in2Buff
        mov ecx, inBuffSize
        call ReadFromFile
        mov esi, offset in1Buff;reseting input buffer pointer
        mov ebx, offset in2Buff;
        mov edi, offset outBuff;reseting output buffer pointer
        cmp eax, inBuffSize
        jne eof
        mov edx, outBuffSize; input value for procedure populateOutBuff
        call populateOutBuff
        ;writing output buffer to file
        mov eax, outFileHnd
        mov edx, offset outBuff
        mov ecx, outBuffSize
        call WriteToFile
        jmp ld
eof:    mov edx,eax; eof-End of file label,  procedure populateOutBuff counter end value
        shl edx, 1 ;Output buffer is 2 times larger than input buffer
        call populateOutBuff

        ;Writing output buffer to file
        mov eax, outFileHnd
        mov ecx, edx
        mov edx, offset outBuff
        call WriteToFile

        ;Closing files
        mov eax, lchFileHnd
        call CloseFile
        mov eax, rchFileHnd
        call CloseFile
        mov eax, outFileHnd
        call CloseFile
        invoke WriteConsole, consoleOutputHandle, ADDR message6, message6Size, 0, 0

        invoke ExitProcess,0
main endp
end main
```

Zaključak

Prilikom izrade projekta susreli smo se sa nekoliko problema. Jedan od problema je bio kako da nađemo veličinu ulaznih fajlova. Pošto u Irvine32.inc nije bio deklarisan prototip funkcije koja vraća veličinu fajla za "handler" fajla morali smo da je deklarišemo sami u okviru wavmux.asm fajla. Funkcija GetFileSize je deo Win32API. Mi smo je implementirali tako da može da pročita veličinu fajla do 4GB jer se u praksi neće koristiti fajl veće veličine od navedene jer FAT32 fajl sistem, koji se koristi danas u većini računara, ne podržava fajlove veće od 4GB. Baferi su statički alocirani u operativnoj memoriji prilikom pokretanja programa. Morali smo da napravimo kompromis između brzine rada programa i zauzeća operativne memorije. Hard disk ima brzinu oko ~60MBps prilikom sekvencijalnog čitanja, ali ona može da opadne na vrlo malu vrednost prilikom nasumičnog čitanja blokova veličine manje od 4KB (~0,5MBps). Vodeći računa o ovome odlučili smo da ulazni baferi budu veličine po 256KB a izlazni veličine 512KB. Pošto današnji računari poseduju više od 1GB operativne memorije smatrali smo da ovi baferi ne zauzimaju mnogo operativne memorije, a performanse programa su zadovoljavajuće s obzirom da spajanje dva audio fajla veličine 17MB traje manje od sekunde. Radi jednostavnosti punjenja bafera ulazni baferi su dimenzinisani tako da budu duplo manji od izlaznog bafera da bi sa jednom petljom mogli da vršimo upis novog sadržaja u ulazne bafere i ispis sadržaja iz izlaznog bafera u fajl. Ovim smo 3 puta smanjili broj provera popunjenosti bafera i povećali brzinu rada programa.