Univerzitet u Beogradu
Elektrotehnički fakultet
Katedra za elektroniku

Projekat iz predmeta Računarska elektronika

Autori:
Petar Kajganić 045/12
Katarina Rakić 156/12

Београд, мај 2016.

Contents

1. Zadatak

Naposati program koji analizira ulazni tekstualni fajl i na osnovu njegovog sadržaja formira odgovarajući ispis na ekranu. Ulazni fajl je oblika:

[r, g, b] tekst pauza

Prvi karakter označava boju teksta r–red, g-green i b-blue, tekst je tekst koji treba da se ispiše u zadatoj boji, a pauza je vreme u milisekundama koje treba da se sačeka dok se ne ispiše sledeća poruka u novom redu.

Primer ulaznog i izlaznog teksta

Ulazni tekst:

```
g neki_tekst 123456
```

Izlazni tekst

```
neki_tekst
```

2. Opis korišćenih funkcija i procedura

U ovom delu su navedene i opisane sve funkcije i procedure redosledom kojim su korišćene u programu.

ReadString

ReadString - Čita string koji se unosi sa tastature, prekida se pritiskom Enter-a. Potrebno je uneti offset u registar EDX i maksimalni broj karaktera za unos (plus jedan) u registar ECX.

U programu se koristi za unošenje imena tekstualnog fajla koji se obrađuje.

OpenInputFile

OpenInputFile – Otvara postojeći fajl. Potrebno je proslediti offset imena fajla u registar EDX. Ako je uspešno otvoren fajl, EAX će sadržati filehandle.

U programu se koristi za otvaranje fajla za obradu i za proveru uspešnosti otvaranja fajla.

ReadFromFile

ReadFromFile – Ubacuje sadržaj fajla u bafer. Potrebno je proslediti ofset bafera u registar EDX i maksimalnu veličinu bafera u registar ECX. Po završetku se proverava Carry flag, ako je nula, EAX sadrži broj procitanih karaktera, a ako je jedan, EAX sadrži gresku koja se ispisuje upotrebom funkcije WriteWindowsMsg

U programu se koristi za čitanje fajla za obradu i za proveru broja karaktera u fajlu (da li je veci od bafera tj od 255).

GetStdHandle

GetStdHandle – Funkcija vraća handle za pisanje ili citanje konzole. Po izvršavanju funkcije, handle se nalazi u EAX pa je dobro kopirati u neku promenljivu da se ne bi izgubilo.

Prototip funkcije:

```
GetStdHandle PROTO,
    nStdHandle:HANDLE
```

SetTextColor

SetTextColor – Postavlja boju teksta i pozadine za sva naredna pisanja u konzolu.

U programu se koristi svaki put pre ispisa teksta u konzolu jer je potrebno imati crvena, zelena i plava slova.

WriteConsole

WriteConsole – Funkcija za ispis karaktera u konzolu na mesto gde se nalazi pokazivac u boji koja je podešena prethodnom procedurom.

Prototip funkcije:

```
WriteConsole PROTO,
        hConsoleOutput:HANDLE,
         lpBuffer:PTR BYTE,
        nNumberOfCharsToWrite:DWORD,
        lpNumberOfCharsWritten:PTR DWORD,
        lpReserved:DWORD
```

gde je lpBudder pokazivac na niz karaktera za ispis, nNumberOfChararsToWrite broj karaktera za ispis, nNumberOfChararsToWritten broj karaktera koji su ispisani a posledji parametar se postavlja na 0 jer se ne koristi.

Sleep

Sleep – Funkcija suspenduje sva trenutna izvršavanja zadat broj milisekundi.

Prototip funkcije:

```
Sleep PROTO,
        dwMilliseconds:DWORD
```

gde je dwMiliseconds broj milisekundi koji program treba da pauzira.

U programu se koristi na kraju svakog reda kako bi se napravila pauza između ispisivanja.

ExitProcess

ExitProcess – Funkcija za izlazak iz programa.

3. Opis rada programa

Prvo je potrebno učitati ime fajla, otvoriti fajl i ubaciti sadržaj fajla u bafer koji služi za obradu. Ovo se izvršava upotrebom procedura koje su gore opisane.

Kroz bafer se prolazi upotrebom brojača koji pokazuje na karakter koji se obrađuje. U registar EBX se upisuje 0 tako da pokazuje na početak bafera tj na prvi karater za obradu i u nastavku programa se koristi kao brojač.

Na početku svakog reda učita se karakter u registar AL i poveća se EBX za 2 jer je na sledećem mestu razmak koji ne treba da se obrađuje. U AL se nalazi jedno od slova r,g ili b. Proberava se koje je od ova tri. Npr. ako je g, skače se na labelu zeleno. Deo programa koji sleduje posle svake od labela crveno, zeleno i plavo radi po istom principu.Učita se karakter iz bafera na koji pokazuje EBX u registar AL. Proverava se da li je taj karakter razmak, ako jeste, ide se na labelu novi_red, a ako nije, karakter se ispisuje određenom bojom funkcijama opisanim u prethodnom poglavlju. Posle ispisa, EBX se uveća za jedan i vraća se na labelu boje koja se koristi. Iz ove petlje se izlazi kada se u AL učita razmak i skače se na labelu novi_red.

Sada je program učitao i ispisao tekst iz jednog reda. Pointer za ispis u konzolu se pomera u sledeći red kako bi bio spreman za obradu sledećeg reda teksta.

Potrebno je učitati broj sa kraja reda koji predstavlja vreme čekanja u milisekundama. U regostar EDI se upisuje 0 i u njega će se smeštati broj koji se čita. Kako se broj čita sa desna na levo, prvo se učitava cifra najveće težine. Vrednost registra EBX se poveća za jedan tako da pokazuje na cifru koja se učita u registar AL i proverava se da li označava kraj reda (EOL) ili kraj fajla (EOF). Ako nije ni jedno od ta dva, registar EDI se pomnoži sa deset. Broj u AL se smanji za 48 jer je u registru ASCII vrednost broja, a nama treba dekadna. Rezultat operacije sub koja se koristi za oduzimanje se nalazi u registru EAX tako da se sadržaj tog registra doda na sadržaj registra EDI i vraća se na početak obrade broja tj na labelu broj. Iz ove petlje se izlazi kada se procita EOL ili EOF.

Kada se pročita EOL, registar EBX se uveća za dva kako bi pokazivao na prvi karakter u sledećer redu i poziva se funkcija Sleep kao što je objašnjeno. Ako se naleti na EOF, poziva se funkcija Sleep, zatvara se fajl sa CloseFile i ExitProcess.

4. Kod

Ovde je priložen kod. Komentari su pisani kao ;// da bi ih program prepoznao kao komentare. Par redova je pod ; jer su oni korišćeni za testiranje tj lakše pregledanje sadržaja bafera.

```
INCLUDE Irvine32.inc
INCLUDE macros.inc

BUFFER_SIZE = 255

.data
buffer BYTE BUFFER_SIZE DUP(? )
filename    BYTE 80 DUP(0)
fileHandle  HANDLE ?
outHandle   HANDLE 0
bytesWritten DWORD ?
endl EQU <0dh, 0ah> ;//end of line
novired LABEL BYTE
BYTE endl
msize DWORD($ - novired)
var DWORD 0 ;  //pomoc za sabiranje vremena

.code
main PROC

;//UCITAVANJE FILE-a
;// Let user input a filename.
mWrite "Unesite ime file-a: "
mov   edx, OFFSET filename
mov   ecx, SIZEOF filename
call  ReadString

;// Open the file for input.
mov   edx, OFFSET filename
call  OpenInputFile
mov   fileHandle, eax

;// Check for errors.
cmp   eax, INVALID_HANDLE_VALUE; error opening file ?
jne   file_ok; no: skip
mWrite <"Cannot open file", 0dh, 0ah>
jmp   quit; and quit
file_ok :

;// Read the file into a buffer.
mov   edx, OFFSET buffer
mov   ecx, BUFFER_SIZE
call  ReadFromFile
jnc   check_buffer_size; error reading ?
mWrite "Error reading file. "; yes: show error message
call  WriteWindowsMsg
jmp   close_file
```

```
check_buffer_size :
cmp   eax, BUFFER_SIZE; buffer large enough ?
jb    buf_size_ok; yes
mWrite <"Error: Buffer too small for the file", 0dh, 0ah>
jmp   quit; and quit

;//Isipisuje broj karaktera
 buf_size_ok:
; mov buffer[eax], 0; insert null terminator
; mWrite "File size: "
; call     WriteDec; display file size
; call     Crlf

;//Ispis buffer-a
; Display the buffer.
 mWrite <"  ", 0dh, 0ah>
; mov edx, OFFSET buffer; display the buffer
; call     WriteString
; call     Crlf


mov ebx,0   ;//brojac za buffer
ponovo:
;//provera boje
mov al, [buffer + ebx] ;//ucitava prvo slovo
add ebx, 2 ;//preskace razmak

cmp al, 'g'; //zeleno?
jz zeleno

cmp al, 'b'; //plavo?
jz plavo

crveno:
mov al, [buffer + ebx] ;// ucitava char
cmp al, ' ' ;//razmak?
jz novi_red;//ako jeste, izadji
;//ako nije ispisi slovo
INVOKE GetStdHandle, STD_OUTPUT_HANDLE
mov outHandle, eax
mov eax,red
call settextcolor
INVOKE WriteConsole,
outHandle,ADDR [buffer + ebx],1,ADDR bytesWritten,0
add ebx, 1; ;// pokazivac na sledece slovo
jmp crveno

zeleno:
mov al, [buffer + ebx];// ucitava slovo
cmp al, ' ';// razmak?
jz novi_red;// ako jeste, izadji
;//ako nije ispisi slovo
INVOKE GetStdHandle, STD_OUTPUT_HANDLE
```

```
mov outHandle, eax
mov eax, green
call settextcolor
INVOKE WriteConsole,
outHandle, ADDR[buffer + ebx], 1, ADDR bytesWritten, 0
add ebx, 1; ;// pokazivac na sledece slovo
jmp zeleno

plavo:
mov al, [buffer + ebx];// ucitava slovo
cmp al, ' ';//razmak?
jz novi_red;//ako jeste, izadji
;//ako nije ispisi slovo
INVOKE GetStdHandle, STD_OUTPUT_HANDLE
mov outHandle, eax
mov eax, blue
call settextcolor
INVOKE WriteConsole,
outHandle, ADDR[buffer + ebx], 1, ADDR bytesWritten, 0
add ebx, 1; // pokazivac na sledece slovo
jmp plavo

;//Pomera pointer za ispis u novi red
novi_red:
INVOKE GetStdHandle, STD_OUTPUT_HANDLE ;
mov outHandle, eax
INVOKE WriteConsole, outHandle, ADDR novired, msize, ADDR bytesWritten, 0

;//citanje broja
mov edi, 0 ;//BR CEKANJE [ms]
broj:
add ebx, 1 ;//pokazuje na trenutni
mov al, [buffer + ebx] ;//cita ga
cmp al, 0 ; // EOF
jz  close_file
cmp al,0dh ;// EOL
jz kraj_linije

imul edi, 10;
sub  al, 48;
add  edi,eax

jmp broj

kraj_linije:
add ebx, 2
INVOKE Sleep,edi
jmp ponovo ;//obrada sedeceg reda


close_file:
INVOKE Sleep, edi ;// cekanje pre izlaza
mov   eax, fileHandle
call  CloseFile
```

```
quit :
invoke ExitProcess, 0
main ENDP

END main
```

5. Pokretanje programa i test

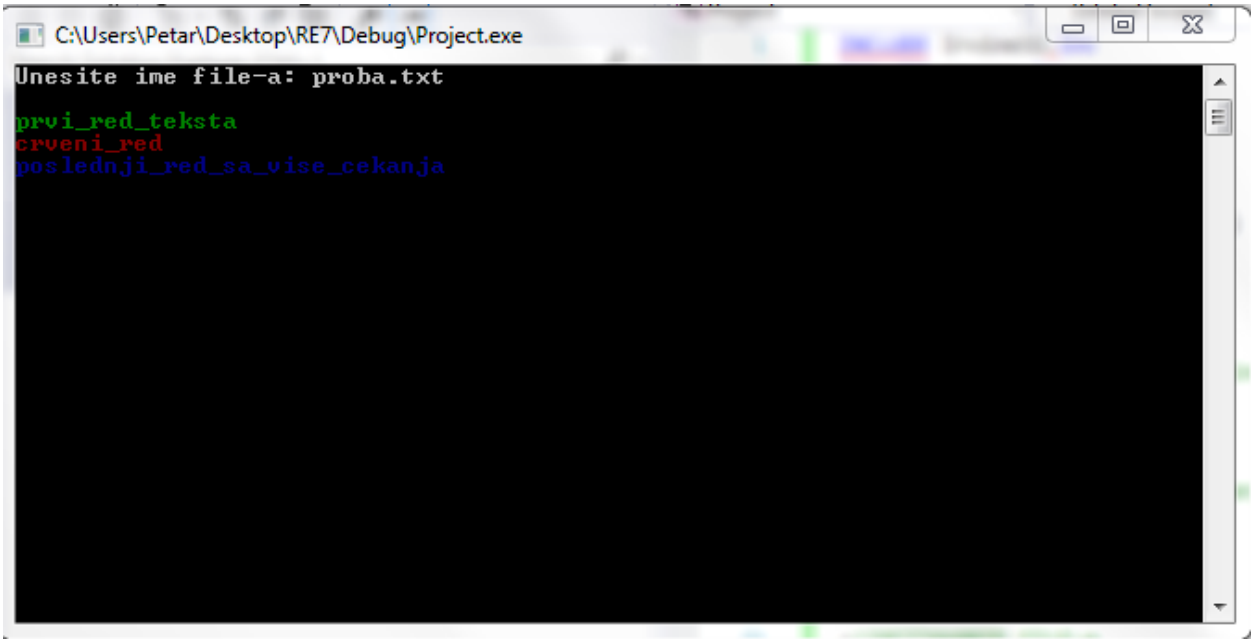
Test fajl se zove proba.txt i nalazi se u folderu gde i sam program. Program će tražiti unošenje imena fajla gde se treba uneti proba.txt i pritisnuti enter. Test fajl treba da bude napisan u odgovarajućem formatu i da sadrži manje od 255 karaktera kako program ne bi izbacio grešku. Sadržaj fajla proba.txt je sledeći:

g prvi_red_teksta 2543

r crveni_red 98

b poslednji_red_sa_vise_cekanja 18992

Na Slici1 je prikazan rezultat:



Slika 1. Konzola nakon završetka rada programa

6. Literatura

1. Vežbe iz predmeta Računarska elektronika
2. Kip R. Irvine Assembly Language for x86 Processors