Izveštaj o projektnom zadatku iz predmeta

RAČUNARSKA ELEKTRONIKA

OE3RE

Student: Aleksandar Ivanković
Broj indeksa: 2012/145

Predmetni profesor:
prof Milan Prokin
Predmetni asistent:
Aleksandra Lekić

Projektni zadatak broj 1

Zadatak: Napraviti program koji pravi PGM sliku duplo manjih dimenzija od pročitane slike decimacijom. Naime, "preskakanjem" svakog drugog piksela dobija se slika duplo manjih dimenzija od početne.

Program je napisan u windows assembly programskom jeziku niskog nivoa. Upotrebljena biblioteka je biblioteka Kip-a Irvine-a *Irvine32.inc*. U okviru ove biblioteke je uobličena i olakšana upotreba windows biblioteke *smallwin.inc*.

Tok programa:

U *.data* sekciji programa se deklarišu promenljive, i to:

bufferIn BYTE BUFFER_SIZE DUP(?) – Ulazni bafer dužine BUFFER_SIZE bajtova. Ovaj bafer služi za dohvatanje dela fajla.

bufferOut BYTE 10 DUP(?) – Izlazni bafer dužine 10 bajtova. Ovaj bafer se najčešće koristi za upisivanje u izlazni fajl.

inputFileName BYTE 80 DUP(0) – String koji sadrži ime ulaznog fajla.

outputFilename BYTE 80 DUP(0) – String koji sadrži ime ulaznog fajla.

inputFileHandle HANDLE ? – Handle za ulazni fajl

outputFileHandle HANDLE ? – Handle za izlazni fajl

pixCount BYTE 0 – Brojač piksela, služi za ispisivanje znaka za novi red nakon određenog broja prepisanih piskela.

pixCount1 DWORD 1 – Brojač koji pokazuje trenutnu kolonu

pixCount2 DWORD 1 – Brojač koji pokazuje trenutnu vrstu

pixWidth DWORD 0 – Sačuvana vrednost širine slike

buffPtr DWORD 0 – Iterativni pokazivač na ulazni bafer.

bytesRead DWORD 0 – Broj bajtova pročitanih ReadFile instrukcijom. Služi za proveru da li je dostignut kraj fajla.

bytesWritten DWORD ? – Opcionalan brojač upisanih bajtova. Nema posebnu funkciju.

numBuffer BYTE 4 DUP(?) – Bafer za prepisivanje podataka o visini, širini slike, kao i maksimalnoj vrednosti piksela.

Tri procedure se koriste za bezbedniji i koncizniji tok programa. To su:

- checkBuffer – Vrši inkrementaciju pokazivača buffPtr, kao i proveru da li je neophodno dovući novih BUFFER_SIZE bajtova u bafer iz fajla. Poziva proceduru read_more_from_file ako jeste.
- read_more_from_file – Vrši dovlačenje sledećih BUFFER_SIZE bajtova u bafer. Poziva je samo checkBuffer rutina.
- intToString – Rutina za konverziju decimalnog broja u string. Koristi registre edx, ecx i edi i smešta novi string u bafer sa početnom adresom u edi. Vraća broj upisanih bajtova u registru eax.

Iz *Irvine.inc* biblioteke se koriste rutine:

- ReadFromFile
- WriteToFile
- ParseDecimal32
- ReadString
- OpenInputFile
- CreateOutputFile
- CloseFile

Struktura glavnog programa:

Pre svega se vrši učitavanje imena ulaznog i izlaznog fajla i smeštaju u odgovarajuće stringove. Zatim se proverava validnost imena fajlova i ako je sve u redu, otvara se ulazni fajl, a kreira izlazni. Sledeći korak je učitavanje prvih BUFFER_SIZE bajtova u ulazni bafer bufferIn. Ukoliko je ovo izvršeno bez problema, počinje se sa upisivanjem/prepisivanjem u izlazni fajl.

Prvi korak obrade je prepisivanje oznake P2 u izlazni fajl. Bilo koji komentar u toku obrade fajla (označen sa karakterom #) se preskače. Sledi čitanje širine i visine slike, konverzija u decimalne brojeve pomoću rutine ParseDecimal32 iz biblioteke *Irvine.inc*. Širina i visina slike se dele sa 2 i upisuju u izlazni fajl. Podatak o maksimalnoj vrednosti piksela se prepisuje.

Nakon ovih pripremnih operacija, na redu je glavna obrada podataka. U ulazni bafer se učitava karakter po karakter i vrši se analiza:

Prepisuju se cifre dok se ne stigne do znaka za razmak (20h). Kad se stigne do znaka za razmak, inkrementira se brojač kolone pixCount1. Proverava se da li je ovaj brojač trenutno paran. Ukoliko jeste, resetuje se flag prepisuj koji označava da li treba prepisati piksel. Ako pixCount1 nije paran, proverava se brojač vrste na isti način i njegova parna vrednost resetuje flag prepisuj. Kad pixCount1 dobroji do pixWidth, resetuje se na 0, a pixCount2 se inkrementira. Ovako se iterira po nizu piksela. Ovime se postiže da se svaka druga kolona i svaka druga vrsta "preskoče". Svaki put kad brojač pixCount dobroji do neke određene vrednosti, upisuje se znak za novi red (0ah). Ovime se zadovoljava preporuka standarda PGMA da u svakom redu ne bude više od 70 karaktera.

Kad se sa analizom stigne do kraja ulaznog fajla, oba fajla se zatvaraju i izlazi se iz programa.

Program je proveren na fajlovima balloons.pgm, mona_lisa.pgm i pepper.pgm koji su priloženi uz source kod i izvršni fajl.

Zaključak:

Rad na ovom projektu nam je doneo korisna saznanja o programiranju u asembleru, kao i važno iskustvo sa windows API-jem. Iako bi se ovaj projekat mogao uraditi znatno brže i lakše u nekom jeziku višeg nivoa, u brzini izvršavanja i kontroli memorije asembler nema konkurenciju.

Priloženi kod celog programa:

```
; Projekat iz Racunarske elektronike
; Studenti: Aleksandar Ivankovic 145/2012 i Jelena Puzovic 002/2012
; Elektrotehnicki Fakultet u Beogradu
; april 2016.
;      Program koji od ulaznog .pgm fajla (PGM - portable greyscale map) pravi .pgm
sliku
; koja je duplo manje velicine. Metod je decimacija niza stringova koji predstavljaju
vrednost
; (nijansu) piksela.
; main.asm - Glavni asemblerski fajl
;
INCLUDE Irvine32.inc
INCLUDE macros.inc

BUFFER_SIZE = 1000

.data

bufferIn BYTE BUFFER_SIZE DUP(?)
bufferOut BYTE 10 DUP(?)
inputFileName BYTE 80 DUP(0)
outputFilename BYTE 80 DUP(0)
inputFileHandle HANDLE ?
outputFileHandle HANDLE ?
pixCount BYTE 0
pixCount1 DWORD 1
pixCount2 DWORD 1
pixWidth DWORD 0
buffPtr DWORD 0
bytesRead DWORD 0
bytesWritten DWORD ?
numBuffer BYTE 4 DUP(?)
; flegovi
prepisuj BYTE 1
EOF_indicator BYTE 0
widthIndicator BYTE 1
pixValueIndicator BYTE 1

.code
; rutina za 'dovlacenje' dodatnih BUFFER_SIZE bajtova iz fajla u bafer
read_more_from_file proc
    push eax
    push edx
    push ecx
    invoke ReadFile, inputFileHandle, OFFSET bufferIn, BUFFER_SIZE, OFFSET bytesRead,
NULL
    mov buffPtr, OFFSET bufferIn
    cmp eax, 1
    je read_is_ok
    mWrite <"Error while getting more data from file. ", 0dh, 0ah>
    exit
read_is_ok:
    cmp bytesRead, 0
    jne return
    mov EOF_indicator, 1
return:
    pop ecx
    pop edx
    pop eax
    ret
```

```
read_more_from_file endp

; rutina za inkrementiranje pokazivaca na ulazni bafer, kao i
; simultanu proveru da li treba 'dovuci' jos BUFFER_SIZE bajtova iz fajla u bafer
checkBuffer proc
    push eax
    inc buffPtr
    mov eax, buffPtr
    sub eax, BUFFER_SIZE
    cmp eax, OFFSET bufferIn
    jne bufferOK
    call read_more_from_file
bufferOK:
    pop eax
    ret
checkBuffer endp

; rutina za konverziju decimalnog broja u string
; koristi registre edx, ecx i edi i smesta novi string
; u bafer sa pocetnom adresom u edi
; vraca broj upisanih bajtova u registru eax
intToString proc
    push  edx
    push  ecx
    push  edi
    push  ebp
    mov   ebp, esp
    mov   ecx, 10
 pushDigits:
    xor   edx, edx        ; zero-extend eax
    div   ecx             ; divide by 10; now edx = next digit
    add   edx, 30h        ; decimal value + 30h => ascii digit
    push  edx             ; push the whole dword, cause that's how x86 rolls
    test  eax, eax        ; leading zeros suck
    jnz   pushDigits
 popDigits:
    pop   eax
    stosb                 ; don't write the whole dword, just the low byte
    cmp   esp, ebp        ; if esp==ebp, we've popped all the digits
    jne   popDigits
    xor   eax, eax        ; add trailing nul
    stosb
    mov   eax, edi
    pop   ebp
    pop   edi
    pop   ecx
    pop   edx
    sub   eax, edi        ; return number of bytes written
    ret
intToString endp

main proc
    ; pripremanje fajlova
    mWrite "Unesite ime fajla slike za obradu: "
    mov edx, OFFSET inputFileName
    mov ecx, SIZEOF inputFileName
    call ReadString
    mWrite "Unesite ime fajla za smestanje obradjene slike: "
    mov edx, OFFSET outputFileName
    mov ecx, SIZEOF outputFileName
    call ReadString
open_input:
```

```
    mov edx, OFFSET inputFileName
    call OpenInputFile
    mov inputFileHandle, eax
    cmp eax, INVALID_HANDLE_VALUE
    jne create_output
    mWrite<"Cannot open file", 0dh, 0ah>
    jmp quit
create_output:
    mov  edx,OFFSET outputFilename
    call CreateOutputFile
    mov  outputFilehandle, eax
    cmp eax, INVALID_HANDLE_VALUE
    jne files_ok
    mWrite<"Cannot create file", 0dh, 0ah>
    jmp close_input_file
files_ok:
    mov eax, inputFileHandle
    mov edx, OFFSET bufferIn
    mov ecx, BUFFER_SIZE
    call ReadFromFile ; ucitavanje prvih BUFFER_SIZE bajtova
    ; kraj pripremanja fajlova
    jnc P2
    mWrite "Error reading file. "
    call WriteWindowsMsg
    jmp close_files
    ; prepisivanje 'magicnog broja' P2
P2:
    cld
    mov ecx, 3
    mov esi, OFFSET bufferIn
    mov edi, OFFSET bufferOut
    rep movsb
    mov bufferOut[3], 0ah
    mov  eax, outputFileHandle
    mov  edx, OFFSET bufferOut
    mov  ecx, 4
    call WriteToFile
    jc   error_writing
    add  bytesWritten,eax
    mov buffPtr, esi
    mov widthIndicator, 1
    ; kraj prepisivanja P2

    ; citanje karaktera iz ulaznog bafera
    ; odavde se skace na delove za obradu odredjenih delova fajla
read_char:
    cmp EOF_indicator, 1
    je close_files
    mov edx, buffPtr
    mov al, [edx]
    cmp al, '#'
    je comment_sign
    cmp widthIndicator, 1
    je width_scaling
    cmp pixValueIndicator, 1
    je pix_value_prepisi
    jmp obrada
    ; deo za prepisivanje i korekciju informacije o visini i sirini slike (dele se sa
2)
width_scaling:
    mov ecx, 0
load_width:
```

```
    mov edx, buffPtr
    mov al, [edx]
    call IsDigit
    jnz width_not_digit
    mov numBuffer[ecx], al
    inc ecx
    call checkBuffer
    jmp load_width
width_not_digit:
    mov edx, OFFSET numBuffer
    call ParseDecimal32
    mov pixWidth, eax
    shr eax, 1
    cmp eax, 100
    jb new_width_under100
    jmp width_cont
new_width_under100:
    mov ecx, 2
width_cont:
    mov edi, OFFSET numBuffer
    call intToString
    mov  numBuffer[ecx], 20h
    inc ecx
    mov  eax, outputFileHandle
    mov  edx, OFFSET numBuffer
    call WriteToFile
    jc   error_writing
    add  bytesWritten,eax
    mov widthIndicator, 0
    call checkBuffer
height_scaling:
    mov ecx, 0
load_height:
    mov edx, buffPtr
    mov al, [edx]
    call IsDigit
    jnz height_not_digit
    mov numBuffer[ecx], al
    inc ecx
    call checkBuffer
    jmp load_height
height_not_digit:
    mov edx, OFFSET numBuffer
    call ParseDecimal32
    shr eax, 1
    cmp eax, 100
    jb  new_height_under100
    jmp cont_height
new_height_under100:
    mov ecx, 2
cont_height:
    mov edi, OFFSET numBuffer
    call intToString
    mov numBuffer[ecx], 0ah
    inc ecx
    mov  eax, outputFileHandle
    mov  edx, OFFSET numBuffer
    call WriteToFile
    jc   error_writing
    add  bytesWritten,eax
    call checkBuffer
    jmp read_char
```

```
    ; kraj dela za prepisivanje i korekciju informacije o visini i sirini slike
    ; deo za obradu komentara, kad god se naidje na liniju koja pocinje znakom '#', ta
linija se preskace
comment_sign:
    inc bytesWritten
    call checkBuffer
    mov edx, buffPtr
    mov al, [edx]
    cmp al, 0ah
    jne comment_sign
    call checkBuffer
    jmp read_char
    ; kraj dela za obradu komentara
    ; deo za prepisivanje informacije o maksimalnoj vrednosti piksela
pix_value_prepisi:
    mov ecx, 3
    mov esi, buffPtr
    mov edi, OFFSET numBuffer
    rep movsb
    mov buffPtr, esi
    call checkBuffer
    mov numBuffer[3], 0ah
    mov  eax, outputFileHandle
    mov  edx, OFFSET numBuffer
    mov  ecx, 4
    call WriteToFile
    jc   error_writing
    add  bytesWritten,eax
    mov pixValueIndicator, 0
    jmp read_char
    ; kraj dela za prepisivanje informacije o maksimalnoj vrednosti piksela


    ; pocetak obrade (decimacije) niza piksela
    ; algoritam je sledeci:
    ; pixCount1 je brojac koji pokazuje koji se piksel u okviru vrste trenutno
analizira (broj kolone)
    ; pixCount2 je brojac koji pokazuje broj vrste
    ; prepisuje se broj dok se ne stigne do znaka za razmak (20h)
    ; kad se stigne do znaka za razmak, inkrementira se pixCount1
    ; proverava se da li je pixCount1 paran broj, ako jeste onda se resetuje flag
prepisuj,
    ; ako nije, proverava se pixCount2 na isti nacin
    ; ako je pixCount1 dosao do kraja vrste, resetuje se, a pixCount2 se inkrementira
    ; ako je fleg prepisuj setovan, onda se piksel ispisuje u izlazni fajl
    ; svaki put kad brojac pixCount dobroji do neke odredjene vrednosti,
    ; upisuje se znak za novi red (0ah). Ovime se zadovoljava preporuka standarda
    ; PGMA da u svakom redu ne bude vise od 70 karaktera

obrada:
    cmp al, 0ah
    jne not_newline
    call checkBuffer
    jmp read_char
not_newline:
    cmp al, 20h
    je toggle_prepisi
    jmp nije_razmak
toggle_prepisi:
    inc pixCount1
    mov eax, pixWidth
    cmp eax, pixCount1
```

```
    jne leave_count2
    inc pixCount2
    mov pixCount1, 1
    jmp reset_prepisuj
leave_count2:
    mov eax, 01h
    test eax, pixCount1
    jz reset_prepisuj
    test eax, pixCount2
    jz reset_prepisuj
    jmp set_prepisuj
reset_prepisuj:
    mov prepisuj, 0
    jmp cont
set_prepisuj:
    mov prepisuj, 1
cont:
    inc pixCount
    cmp pixCount, 36
    je reset_pix_count
    jmp nije_razmak
reset_pix_count:
    mov pixCount, 0
    mov bufferOut[0], 0ah
    mov  eax, outputFileHandle
    mov  edx, OFFSET bufferOut
    mov  ecx, 1
    call WriteToFile
    jc   error_writing
    add  bytesWritten,eax
nije_razmak:
    cmp prepisuj, 1
    je prepisivanje
    call checkBuffer
    jmp read_char
prepisivanje:
    cld
    mov ecx, 1
    mov esi, buffPtr
    mov edi, OFFSET bufferOut
    rep movsb
    mov  eax, outputFileHandle
    mov  edx, OFFSET bufferOut
    mov  ecx, 1
    call WriteToFile
    jc   error_writing
    add  bytesWritten,eax
    call checkBuffer
    jmp read_char
    ; kraj dela za obradu piksela
    ; detekcija greske
error_writing:
    mWrite <"Error while writing to file. ", 0dh, 0ah>

    ; zavrsni deo programa, zatvaranje fajlova i izlazak
close_files:
    mov eax, outputFileHandle
    call CloseFile
close_input_file:
    mov eax, inputFileHandle
    call CloseFile
quit:
```

```
    exit
main ENDP
END main
```