Elektrotehnički fakultet, Univerzitet u Beogradu

Izveštaj

Računarska elektronika

projekat: Skaliranje slike primenom metoda najbližeg suseda

Kristina Dolovac 304/2011

Marko Lazović 357/2014

Predmetni profesor: dr Milan Prokin

Predmetni asistenti: Aleksandra Lekić, Aleksandra Brkić

Sadržaj

1. Postavka zadatka

Napisati program za skaliranje slike primenom metode najbližeg suseda. Faktor s, sa kojim se vrši skaliranje, unosi se sa standardnog ulaza i uvek je veći ili jednak 1. Pored toga, sa standardnog ulaza na određeni način se definiše da li se vrši povećanje ili decimacija slike s puta.

Povećanje dimenzija slike primenom metoda najbližeg suseda, vrši se na sledeći način:

Neka je $g(x', y')$ intezitet piksela u skaliranoj slici na poziciji (x', y'). Koordinate u ulaznoj slici koje odgovaraju ovom pikselu izlazne slike su:

$$x = \frac{x'}{s} \text{ i } y = \frac{y'}{s}$$

Kako x i y ne moraju biti celi brojevi, intezitet piksela $(g(x', y'))$ dobija se posmatranjem 4 najbliža piksela u originalnoj slici: $f(x_0, y_0)$, $f(x_0, y_0 + 1)$, $f(x_0 + 1, y_0)$ i $f(x_0 + 1, y_0 + 1)$, pri čemu je:

$$x_0 = \left[\frac{x'}{s}\right] \quad \text{i} \quad x_r = x' - x_0 * s$$

$$y_0 = \left[\frac{y'}{s}\right] \quad \text{i} \quad y_r = y' - y_0 * s$$

Sada se intezitet piksela u izlaznoj slici računa tako što se:

- Ukoliko je $x_r < s - x_r$, uzima se piksel u ulaznoj slici čija je x koordinata x_0, dok se u suprotnom uzima piksel čija je x koordinata $x_0 + 1$
- Ukoliko je $y_r < s - y_r$, uzima se piksel u ulaznoj slici čija je y koordinata y_0, dok se u suprotnom uzima piksel čija je y koordinata $y_0 + 1$

U slučaju decimacije slike dimenzije slike se odredjuju kao: $\left[\frac{N}{s}\right]$ i $\left[\frac{M}{s}\right]$ gde su M i N dimenzije slike, a inteziteti piksela u izlaznoj slici se odredjuju prema formuli:

$$g(x', y') = f(sx', sy')$$

2. Realizacija projekta

Projekat je realizovan pomoću mašinskog jezika *asembler* u programskom paketu *Visual Studio*.

U .const sekciji programa deklarišu se konstante,a to su:

BUFFER_SIZE = 200000 - dozvoljena veličina ulaznog i izlaznog bafera;

MAX_PICTURE_SIZE = 65536 - maksimalna dozvoljena veličina slike;

U .data sekciji programa deklarišu se promenljive,a to su:

buffer BYTE BUFFER_SIZE DUP(?) - Ulazni bafer dužine BUFFER_SIZE bajtova. Ovaj bafer služi za dohvatanje celog ulaznog fajla;

filename BYTE 80 DUP(0) - String koji sadrži ime ulaznog fajla;

fileHandle HANDLE ? - Handle za ulazni fajl;

outbuffer BYTE BUFFER_SIZE DUP(?) - Izlazni bafer dužine BUFFER_SIZE bajtova. Ovaj bafer se koristi za upisivanje u izlazni fajl;

outfilename BYTE 80 DUP(0) - String koji sadrži ime izlaznog fajla;

outfileHandle HANDLE ? - Handle za izlazni fajl;

outindex DWORD 0 – Pomoćni izlazni indeks koji pokazuje na indeks outbuffer-a od kojeg se upisuju promenjeni pikseli u proceduri IzlazniFajl;

Osnovni parametri slike:

P2 WORD 5032h - Pomoćna konstanta za proveru formata slike

velicinaBafera DWORD ? -Pomoćna promenljiva koja ispituje ispravnost veličine ulaznog fajla;

krajBuffera DWORD ? -Pomoćna promenljiva koja predstavlja kraj bafera

M WORD ? - Visina ulazne/izlazne slike;

N WORD ? - Širina ulazne/izlazne slike;

Lmax WORD ? - Maksimalna vrednost intenziteta piksela u slici;

Pomoćne promenljive:

s WORD ? – Promenljiva koja predstavlja faktor kojim se skalira slika

newM WORD ? - Visina ulazne/izlazne slike nakon skaliranja

newN WORD? - Širina ulazne/izlazne slike nakon skaliranja

x0 WORD? – Pomoćne promenljive koje koristimo u formulama za skaliranje slike

y0 WORD ? – Pomoćne promenljive koje koristimo u formulama za skaliranje slike

xr WORD? – Pomoćne promenljive koje koristimo u formulama za skaliranje slike

yr WORD ? – Pomoćne promenljive koje koristimo u formulama za skaliranje slike

x_prim WORD? – Pomoćne promenljive koje koristimo u formulama za skaliranje slike

y_prim WORD? – Pomoćne promenljive koje koristimo u formulama za skaliranje slike

indexIn WORD ? – Promenljiva koja predstavlja indeks u baferu pri obradi slike

pocetakNiza DWORD 0 - Promenljiva koja se koristi kao indeks;

ulazniPixeli WORD MAX_PICTURE_SIZE DUP(?) - Niz koji se koristi za smeštanje vrednosti iz ulaznog bafera. Kada se naiđe na EOL, u niz se upisuje vrednost -1, zbog toga je niz tipa WORD. U suprotnom, ako bi bilo tipa BYTE, vrednost -1 bila bi ista kao vrednost 255;

izlazniPixeli WORD MAX_PICTURE_SIZE DUP(?) - Niz koji se koristi za smeštanje izlaznih vrednosti piksela

broj BYTE 4 DUP(?) - Niz koji služi za smeštanje trocifrenih,dvocifrenih i jednocifrenih brojeva (piksela) kada se konvertuju iz ulaznog fajla (gde su smešteni u string) u decimalne vrednosti, ali i za smeštanje izlaznih piksela koji se konvertuju na kraju u string i šalju u izlazni fajl;

cifra WORD 0 - Određuje da li je broj jednocifreni, dvocifreni ili trocifreni (WORD zbog ECX registra);

brojac DWORD 0 – Pomoćna promenljiva koja obezbeđuje da se neće obraditi vrednosti koje ukazuju na EOL;

STOTINA WORD 100 – Promenljiva koja sluzi pri odredjivanju cifara za ispis u izlazni fajl, za stotine

DESETICA WORD 10 - Promenljiva koja sluzi pri odredjivanju cifara za ispis u izlazni fajl, za desetice

zaNoviRed WORD 70 – Promenljiva koja pokazuje kad treba preci u novi red pri ispisu slike u izlazni fajl

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

- *CitajBroj* - Pretvara vrednost čiji je početni član buffer[početakNiza] iz CHAR u INT. Po izlasku iz procedure vrednost je sačuvana u EAX, dok buffer[pocetakNiza] pokazuje na 20h (space). Procedura za dobijanje decimalnih vrednosti piksela.

- *Uvod* - Otvara se ulazni fajl i proverava ispravnost njegovog formata. Ukoliko je fajl ispravnog formata preskače se prvi red,i drugi red sa komentarom. Učitavaju se parametri M, N, Lmax i preskače se treći red. Formira se izlazni fajl. Prepisuju se prva četiri reda u outbuffer. Po izlasku iz procedure vrednost buffer[pocetakNiza] pokazuje na prvi pixel, dok outindex pokazuje na index outbuffer-a od kojeg upisujemo promenjene pixele u proceduri IzlazniFajl.

- *ObradaSlike* – Ova procedura služi za skaliranje slike. Definišu se nove dimenzije slike u zavisnosti od toga da li korisnik želi da poveća ili da smanji sliku i u zavisnosti od parametra *s* (faktor za skaliranje slike). Skaliranje slike se radi pomoću formula date u specifikaciji projekta primenom metoda najbližeg suseda. Skalirane vrednosti piksela se upisuju u izlazni niz koji se zove *izlazniPixeli*.

- *IzlazIspis* - Popunjava se outbuffer[] počevši od vrednosti outindex, a zatim se outbuffer[] šalje u izlazni fajl. Petlja *NoviIzlazni* popunjava outbuffer novim vrednostima niza pixeli[]. Da bi se popunio outbuffer moraju se pretvoriti decimalne vrednosti u string (INT u CHAR), tako da se vrši provera broja cifara INT, a zatim na tu vrednost doda 30h i tako se dobija ASCII vrednost broja. Upisuju se vrednosti iz outbuffer-a u izlazni fajl i dobija se skalirana slika.

Iz *Irvine.inc* biblioteke se koriste rutine:

- *ReadFromFile*
- *WriteToFile*
- *ParseDecimal32*
- *ReadString*

- *OpenInputFile*
- *CreateOutputFile*
- *CloseFile*
- *WriteWindowsMsg*

Struktura glavnog programa .main:

Vrši se učitavanje imena ulaznog fajla i koje se smešta u odgovarajući string,ukoliko nema greške pri otvaranju i čitanju ulaznog fajla i ukoliko su dimenzije slike odovarajuće. Sledeći korak je pozivanje odgovarajućih procedura *Uvod*, *ObradaSlike* i *IzlazIspis*. Kada se završi formiranje skalirane slike (uslov projektnog zadatka) tj.izlaznog fajla,oba fajla se zatvaraju i izlazi se iz programa.

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

3. Implementacija

Priloženi kod celog programa:

```
;// Projekat iz Računarske elektronike
;// Studenti: Kristina Dolovac 304/2011 i Marko Lazovic 357/2014
;// Elektrotehnički fakultet u Beogradu
;// jun 2017
;// Projekat broj 2: Skaliranje slike primenom metoda najblizeg suseda

INCLUDE Irvine32.inc
INCLUDE macros.inc

.const
	BUFFER_SIZE = 200000
	MAX_PICTURE_SIZE = 65536

.data
	buffer BYTE BUFFER_SIZE DUP(?)
	filename    BYTE 80 DUP(0)
	fileHandle  HANDLE ?

	outbuffer BYTE BUFFER_SIZE DUP(?)
	outfilename BYTE 80 DUP(0)
	outfileHandle HANDLE ?
	outindex DWORD 0

	; Osnovni parametri slike:
	P2 WORD 5032h
	velicinaBafera DWORD ?
	krajBuffera DWORD ?
	M WORD ?
	N WORD ?
	Lmax WORD ?

	s word ?

	newM word ?
	newN word ?

	x0 word ?
	y0 word ?
	xr word ?
	yr word ?
	x_prim word ?
	y_prim word ?
	indexIn word ?

	pocetakNiza DWORD 0; Promenljivu koristimo kao index.
	ulazniPixeli WORD MAX_PICTURE_SIZE dup (?); Niz koji koristimo za smestanje
vrednosti iz buffera. Kada naidjemo na EOL, u niz upisujemo vrednost -1, zbog toga je
niz tipa WORD.U suprotnom, ako bismo imali BYTE, vrednost -1 bila bi ista kao vrednost
255.
	izlazniPixeli WORD MAX_PICTURE_SIZE DUP(?)

	broj BYTE 4 DUP(?)
	cifra WORD 0; Odredjuje da li je broj jednocif,dvocif ili trocif (WORD zbog cx).
	brojac WORD 0
```

```
	STOTINA WORD 100
	DESETICA WORD 10
	zaNoviRed WORD 70

.code

	close_file PROC
		mov    eax,fileHandle
		call CloseFile
		exit
	close_file ENDP

	;                    CitajBroj
	; Pretvara vrednost ciji je pocetni clan buffer[pocetakNiza] u INT.
	; Po izlasku iz procedure vrednost je sacuvana u EAX, dok buffer[pocetakNiza]
pokazuje na 20h(space).

	CitajBroj PROC STDCALL USES ebx esi ecx

		mov edx,OFFSET buffer;
		xor eax,eax
		xor ecx,ecx
		xor ebx,ebx

		add edx,pocetakNiza
		; Petlja ce se obradjivati sve dok ne naidjemo na 20h(space) ili
0Ah(EOL).
	Ucitavanje:
		mov al,[edx]
		mov broj[ebx],al
		inc edx
		inc pocetakNiza
		inc ebx
		cmp edx, krajBuffera
		je Pretvaranje
		mov al,[edx]
		cmp al,20h
		je Pretvaranje
		cmp al,0ah
		jne Ucitavanje

	Pretvaranje:
		mov broj[ebx],3 ; Kraj stringa u ASCII je 3h.
		; Ovim resavamo problem jednocifrenih i dvocifrenih brojeva.

	; ParseDecimal zahteva da EDX i ECX budu popunjeni na ovaj nacin.
		mov edx,OFFSET broj
		mov ecx,ebx
		call ParseDecimal32

	; Pre povratka iz rutine vracamo offset buffer-a u EDX.
		mov edx,OFFSET buffer
		ret
		CitajBroj ENDP


;                        Uvod
; Otvaramo sliku i formiramo izlaznu sliku uz provere ispravnosti.
; Ucitavamo parametre M,N,Lmax i komentar.
; Prepisujemo prva cetiri reda u outbuffer.
; Po izlasku iz procedure vrednost buffer[pocetakNiza] pokazuje na prvi pixel,
```

```
; dok outindex pokazuje na index outbuffer-a od kojeg upisujemo promenjene pixele u
proceduri IzlazniFajl.

	Uvod PROC
		mov edx, OFFSET buffer
		xor ebx,ebx
		mov ebx,edx
		add ebx,pocetakNiza
		mov ah,[ebx]
		inc ebx
		mov al,[ebx]
		cmp ax,P2
		je DrugiRed
		mWrite <"Format slike je pogresan.">
		call WriteWindowsMsg
		call close_file

	DrugiRed:
		add pocetakNiza,3  ; Sada pocetakNiza pokazuje na #.
		add ebx,2
		; Prelazimo preko komentara.
	Komentar:
		inc pocetakNiza
		inc ebx
		mov dl,[ebx]
		cmp dl,0ah
		jne Komentar

	TreciRed:  ; Labela resava problem za nesting!
		inc pocetakNiza
		call CitajBroj    ; Ucitali smo M.
		mov M,ax
		inc pocetakNiza
		call CitajBroj
		mov N,ax          ; Ucitali smo N.
		inc pocetakNiza
		call CitajBroj
		mov Lmax,ax       ; Ucitali smo Lmax.

		mWrite "Unesite zeljeno ime izlaznog fajla: "
		mov edx,OFFSET outfilename
		mov ecx,SIZEOF outfilename
		call ReadString

		call CreateOutputFile
		mov  outfileHandle,eax

		; Prepisujemo prva cetiri reda u izlazni fajl.
		; Po izlasku iz petlje rucno upisujemo buffer[0], jer za vrednost ECX=0
nismo prosli kroz petlju.
		mov ecx,pocetakNiza
		Prepisivanje:
		mov al,buffer[ecx]
		mov outbuffer[ecx],al
		loop Prepisivanje

		mov al,buffer[0]
		mov outbuffer[ecx],al

		inc pocetakNiza     ; PocetakNiza pokazuje na prvi pixel.
		mov eax,pocetakNiza
		mov outindex,eax
```

```
		mov eax, offset buffer
		add eax, velicinaBafera
		mov krajBuffera, eax

		ret
	Uvod ENDP


	IzdvajanjePixela PROC
		mov eax, offset ulazniPixeli
		xor eax, eax
		xor ecx, ecx
		mov ax, M
		mul N
		mov cx, ax
		xor edi, edi
	upisuj:
		call CitajBroj
		mov ulazniPixeli[edi*2], ax
		inc pocetakNiza
		inc edi
		loop upisuj

		mov    esi, OFFSET ulazniPixeli
		mov ax, M
		mul N
		add ax, ax
		mov    cx, ax
		mov    ebx, SIZEOF ulazniPixeli
		call   DumpMem

		ret

	IzdvajanjePixela ENDP

;                         ObradaSlike
; Ova procedura sluzi za skaliranje slike. Definisu se nove dimenzije slike
; u zavisnosti od toga da li korisnik zeli da poveca ili da smanji sliku i u
zavisnosti
; od parametra s (faktor za skaliranje slike).
; Skaliranje slike se radi pomocu formula date u specifikaciji projekta
	ObradaSlike PROC
		mov ax, s
		imul ax, M
		mov newM, ax

		mov ax, s
		imul ax, N
		mov newN, ax

		imul ax, newM

		xor ecx, ecx
		mov cx, ax
		xor edi, edi
		xor esi, esi
		xor eax, eax
		xor ebx,ebx
	petlja:
		xor edx, edx      ;definisanje x' i y'
		mov ax, si
```

```
        idiv newN
        mov x_prim, ax
        mov y_prim, dx

        xor edx,edx       ;odredjivanje x0 i y0
        mov ax,  x_prim
        idiv s
        mov x0, ax

        xor edx, edx
        mov ax,  y_prim
        idiv s
        mov y0, ax

        mov bx, s         ;odredjivanje xr i yr
        imul bx, x0
        mov ax, x_prim
        sub ax, bx
        mov xr, ax

        mov bx, s
        imul bx, y0
        mov ax, y_prim
        sub ax, bx
        mov yr, ax

        mov ax, s       ; odredjivanje koji se pixel prepisuje
        sub ax, xr
        cmp xr, ax
        jl manjixr
        mov ax, x0
        inc ax
        imul N
        mov indexIn, ax
        jmp dalje
manjixr:
        mov ax, x0
        imul N
        mov indexIn, ax
dalje:

        mov ax, s
        sub ax, yr
        cmp yr, ax
        jl manjiyr
        mov ax, y0
        inc ax
        add indexIn, ax
        jmp dodela
manjiyr:
        mov ax, y0
        add indexIn, ax

dodela:
        mov di, indexIn
        mov ax, ulazniPixeli[edi*2]
        mov izlazniPixeli[esi*2], ax

        inc esi
        dec cx
        jnz petlja
```

```
	mov	esi, OFFSET izlazniPixeli
	mov ax, newM
	mul newN
	add ax, ax
	mov	cx, ax
	mov	ebx, SIZEOF izlazniPixeli
	call	DumpMem

	ret
ObradaSlike ENDP

Decimacija PROC
	xor edx, edx
	mov ax, M
	idiv s
	mov newM, ax

	xor edx, edx
	mov ax, N
	idiv s
	mov newN, ax

	xor esi, esi
	xor edi, edi
	xor ecx, ecx
	xor edx,edx
	mov ax, M
	imul N
	mov duzina, ax

petlja:

	xor edx, edx
	mov ax, si
	idiv N
	mov x0, ax
	mov y0, dx

	xor edx, edx
	mov ax, x0
	idiv s
	cmp dx, 0
	jnz preskociRed

	xor edx, edx
	mov ax, y0
	idiv s
	cmp dx, 0
	jnz dalje

	mov ax, ulazniPixeli[esi*2]
	mov izlazniPixeli[edi*2], ax
	inc edi
dalje:
	inc esi
	jmp provera

preskociRed:
	add si, N

provera:
	cmp si, duzina
```

```
		jnz petlja

		ret
	Decimacija ENDP

;                          IzlazIspis
; Ova procedura sluzi za ispisivanje slike u izlazni fajl.
; Petlja NoviIzlazni popunjava outbuffer vrednostima niza pixeli[].
; Da bismo popunili outbuffer moramo pretvoriti INT u CHAR, tako da vrsimo proveru
broja cifara INT,
; a zatim na tu vrednost dodajemo 30h i tako dobijamo ascii vrednost broja.

	IzlazIspis PROC

		mov edx,OFFSET izlazniPixeli
		xor ebx,ebx
		xor edx,edx

		mov ax, newM
		mul newN
		mov brojac,ax;

		xor eax,eax
		xor ebx,ebx;// EBX je indeks od outbuffer BYTE, dok je EDI indeks od
izlazniPixeli WORD.
		xor edi,edi
		mov ebx,outindex

	NoviIzlazni:

		xor edx, edx
		mov ax, brojac
		idiv zaNoviRed
		cmp dx, 0
		je noviRed
		jmp preskoci

	noviRed:
		mov outbuffer[edi], 0ah

	preskoci:
		mov ax,izlazniPixeli[edi]   ; DA LI DA MNOZIM SA 2
		xor edx,edx
		div STOTINA
		cmp ax,0

		je Dvocifren
	Trocifren:;// Trocifreni broj
		mov cifra,3
		add al,30h
		mov broj[0],al;// Stotine
		mov ax,dx
		xor edx,edx
		div DESETICA
		add al,30h
		add dl,30h
		mov broj[1],al;// Desetice
		mov broj[2],dl;// Jedinice
		jmp Ispis

	Dvocifren:;// Dvocifreni broj
		mov cifra,2
```

```
		mov al,dl
		xor edx,edx
		div DESETICA
		cmp al,0
		je Jednocifren
		add al,30h
		add dl,30h
		mov broj[0],al;// Desetice
		mov broj[1],dl;// Jedinice
		jmp Ispis

	Jednocifren:;// Jednocifreni broj
		mov cifra,1
		add dl,30h
		mov broj[0],dl;// Jedinice

	Ispis:
		xor edx,edx
		xor eax,eax
		mov eax,edi;// EAX sada cuva index od izlazniPixeli[] dok koristimo EDI
za index niza broj.
		xor edi,edi
	Dodavanje:
		mov dl,broj[edi]
		mov outbuffer[ebx],dl
		inc ebx
		inc edi
		dec cifra
		cmp cifra,0
		jnz Dodavanje
		mov outbuffer[ebx],20h;// Ne povecavamo ebx, zato sto se to radi u labeli
Sledeci.

		inc ebx
		mov edi,eax
		inc edi
		inc edi

		dec brojac
		cmp brojac,0
		jne NoviIzlazni

		mov  eax, outfileHandle
		mov  edx, OFFSET outbuffer
		mov  ecx,BUFFER_SIZE
		call WriteToFile

		ret

IzlazIspis ENDP



	main PROC

		mWrite <"Unesite ime slike u .pgm formatu: ">
		mov    edx,OFFSET filename
		mov    ecx,SIZEOF filename
		call ReadString

		mov    edx,OFFSET filename
		call OpenInputFile
```

```
        mov    fileHandle,eax

        cmp    eax,INVALID_HANDLE_VALUE
        jne    file_ok
        mWrite <"Greska pri otvaranju fajla.",0dh,0ah>
        call WriteWindowsMsg
        jmp    quit

file_ok:
        mov    edx,OFFSET buffer
        mov    ecx,BUFFER_SIZE
        call ReadFromFile
        jnc    check_buffer_size

        mWrite <"Greska pri citanju fajla. ",0dh,0ah>
        call WriteWindowsMsg
        mov eax, fileHandle
        call   close_file
        jmp quit

        check_buffer_size:
        cmp    eax,BUFFER_SIZE
        jbe    buf_size_ok

        mWrite <"Greska: Dimenzije slike su prevelike.",0dh,0ah>
        call WriteWindowsMsg
        mov eax, fileHandle
        call   close_file
        jmp quit

buf_size_ok:
        mov velicinaBafera,eax

        call Uvod
        call IzdvajanjePixela

        mWrite "Unesite faktor skaliranja slike: "
        mov edx, OFFSET broj
        mov ecx, SIZEOF broj
        call ReadString

        mov broj[eax], 3

        mov edx,OFFSET broj
        mov ecx,eax
        call ParseDecimal32
        mov s, ax

        mWrite "Za povecanje slike unesite 1, a za decimaciju 0: "
        call ReadChar

        cmp al, 31h
        jz uvecanje
        call Decimacija
        jmp kraj
uvecanje:
        call ObradaSlike
kraj:

        call IzlazIspis

        mov  eax,outfileHandle
```

```
        call CloseFile

        mov    eax,fileHandle
        call CloseFile

quit:
        exit
main ENDP
END main
```

4. Zaključak

Rad na ovom projektu doprineo je našem znanju i iskustvu o programiranju u asembleru. Iako bi ovaj projekat mogao mnogo brže i lakše da se uradi u nekom jeziku višeg nivoa, u brzini izvršavanja i kontroli memorije asembler je mnogo bolji.