Izveštaj o projektnom zadatku iz predmeta

RAČUNARSKA ELEKTRONIKA
OE3RE

Studenti: Nikola Planić 2014/304
Anja Kokerić 2012/369

Predmetni profesor:
prof Milan Prokin

Predmetni asistent:
Aleksandra Lekić
Aleksandra Brkić

Projektni zadatak broj 3

Zadatak: Napisati program kojim se vrši ekvalizacija histograma ulazne slike.
Histogram ekvalizacija predstavlja metod poboljšanja slike transformacijom intenziteta i može se opisati sledećim setom koraka:
Korak 1 : formiranje histograma slike
Histogram slike može se predstaviti nizom koji sadrži onoliko članova koliko različitih intenziteta piksela postoji u slici (pri izradi zadatka smatrati da su to intenziteti od 0- 255), a vrednost svakog pojedinačnog elementa niza predstavlja koliko piksela određenog intenziteta postoji u slici.
Korak 2 : transformacija intenziteta
Transformacija koja se vrši nad ulaznim pikselima sada se može opisati na sledeći način:

$$g_k = \left(Lmax / MN\right)\sum_{i=0}^{k} n_i \quad k = 0,...,Lmax$$

L*max* = maksimalna vrednost intenziteta piksela u slici;
n_i = broj piksela u slici koji imaju intenzitet i ;
M, N = dimenzije slike;

Dakle, svaki piksel inteziteta s_k , k = 0,..., Lmax u ulaznoj slici, zamenjuje se pikselom intenziteta g_k određenog na način koji je prethodno opisan.
Program je napisan u assembly programskom jeziku niskog nivoa.
Upotrebljena biblioteka je biblioteka Kip-a Irvine-a *Irvine32.inc*.

Tok programa:

- 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 (ispravan format .pgm);

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

M WORD ? - Visina ulazne/izlazne slike;

N WORD ? - Širina ulazne/izlazne slike;

Lmax WORD ? - Maksimalna vrednost intenziteta piksela u slici;

- Pomoćne promenljive:
pocetakNiza DWORD 0 - Promenljiva koja se koristi kao indeks;

suma DWORD 0 - Promenljiva koja se koristi u proceduri ObradaSlike za zadatu formulu;

histogram WORD 256 DUP(0) - Vrednost člana niza zavisi od broja ponavljanja u nizu pikseli;

pixeli 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;

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;

- 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.

- *PixHis* - Popunjava se histogram. Koristi se procedura *CitajBroj* da bi se dobila INT vrednost piksela, koje upisujemo u niz pixeli[]. Kada se naiđe na EOL, upisuje se vrednost -1 u niz pixeli. Na ovaj način se obezbeđuje prepoznavanje novog reda u ulaznom fajlu. Po izlasku iz procedure, niz pixeli[] je popunjen vrednostima piksela originalne slike.

- *ObradaSlike* – Koristi se formula za promenu vrednosti piksela. Vrednost histogram[EDI] jednaka je broju ponavljanja piksela EDI. On se prvo doda sumi, a zatim se menja. Po izlasku iz procedure u nizu histogram[] se nalaze nove vrednosti pixel-a.

- *IzlazniFajl* - Menja se niz pixeli[] novim vrednostima iz histogram-a. 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. Ako se naiđe na -1, signalira se kraj reda i upisuje se 0Ah u outbuffer.U tom slucaju promenljiva brojac se ne smanjuje.
  Upisuju se vrednosti iz outbuffer-a u izlazni fajl i dobija se izoštrena 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*,*PixHis*,*ObradaSlike* i *IzlazniFajl*.
  Kada se završi formiranje izoštrene slike (uslov projektnog zadatka) tj.izlaznog fajla,oba fajla se zatvaraju i izlazi se iz programa.

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

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.

- Priloženi kod celog programa:

```
;// Projekat iz Računarske elektronike
;// Studenti: Nikola Planić 304/2014 i Anja Kokerić 369/2012
;// Elektrotehnički Fakultet u Beogradu
;// jun 2017.
;// Program kojim se vrši ekvalizacija histograma ulazne slike.

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 ?
M WORD ?
N WORD ?
Lmax WORD ?

;// Promenljive
pocetakNiza DWORD 0;// Promenljivu koristimo kao index.
suma DWORD 0;// Promenljivu koristimo u proceduri ObradaSlike.
histogram WORD 256 DUP(0);// Vrednost clana niza zavisi od broja ponavljanja u nizu pixeli.
pixeli 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.
broj BYTE 4 DUP(?)
cifra WORD 0;// Odredjuje da li je broj jednocif,dvocif ili trocif (WORD zbog cx).
brojac DWORD 0

;//CODE
.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
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

ret
Uvod ENDP

;// PixHis
;// Koristimo proceduru CitajBroj da bismo dobili INT vrednosti, koje upisujemo u niz pixeli[].
;// Kada naidjemo na EOL, upisujemo vrednost -1.
;// Po izlasku iz procedure niz pixeli[] je popunjen vrednostima pixel-a originalne slike.
PixHis PROC

xor ebx,ebx;// Koristimo ga kao brojac
xor eax,eax
mov ax, M
mul N
mov ecx,eax

Petlja:
call CitajBroj
mov pixeli[ebx],ax;// Upisujemo svaki pixel slike.
imul eax,2;// EAX sada postaje indeks histograma. Mnozimo sa 2,jer je histogram WORD!
mov esi,eax
inc histogram[esi]
inc pocetakNiza
add edx,pocetakNiza
mov edi,[edx]
mov edx,edi
cmp dl,0ah;// Proveravamo da li nam pocetakNiza trenutno pokazuje na EOL.
jnz Nastavi
inc ebx;// Povecavamo ebx za 2 jer je on indeks pixel-a, koji su WORD.
inc ebx
mov pixeli[ebx],-1
inc pocetakNiza;// Pomeramo pocetakNiza na prvu cifru narednog pixela
Nastavi:
inc ebx
inc ebx

loop Petlja

ret
PixHis ENDP

;//ObradaSlike
;// Koristimo formulu za promenu vrednosti pixel-a.
;// Vrednost histogram[edi] jednaka je broju ponavljanja pixel-a EDI. Njega prvo dodajemo sumi,
;// a zatim ga menjamo.
;// Po izlasku iz procedure u nizu histogram[] se nalaze nove vrednosti pixel-a.
ObradaSlike PROC
xor ecx,ecx
xor eax,eax
xor ebx,ebx
mov cx,Lmax
mov edi,0

Obrada:
mov ax,histogram[edi]
add suma,eax
mov eax,suma
cmp eax,0;// Proveravamo da li se pixel EDI pojavio u originalnoj slici.
jz Sledeci
```

```
xor edx,edx
mov eax,suma;// Ubacujemo ponovo u ax, jer u suprotnom prijavljuje gresku za bracket.
mul Lmax
mov bx,M
imul bx,N
div bx
cmp eax,0; // Nakon promene u formuli proveravamo da li je nova vrednost pixel-a nula.
jne NijeNula

Nula:
mov histogram[edi],0
jmp Sledeci
NijeNula:
cmp eax,255
jg Crno

mov histogram[edi],ax
jmp Sledeci
Crno:
mov histogram[edi],255
Sledeci:
add edi,2
loop Obrada

ret
ObradaSlike ENDP

;//IzlazniFajl
;// Menjamo niz pixeli[] novim vrednostima iz histogram-a.
;// Popunjavamo outbuffer[] pocevsi od vrednosti outindex, a zatim outbuffer[] saljemo
;// na izlazni fajl.
IzlazniFajl PROC
mov edx,OFFSET pixeli
xor ebx,ebx
xor edx,edx

mov ax,M
mul N
mov brojac,eax;// Koristimo brojac,jer zelimo da se osiguramo da necemo obraditi vrednosti koje ukazuju na EOL.

;// Menjamo niz pixeli[] novim vrednostima iz histogram-a.
NoviPixeli:
mov ax,pixeli[ebx]
cmp ax,-1
je VrednostNovogReda;// Ovim smo sigurni da smo obradili samo pixele, a ne i vrednosti koje ukazuju na EOL.
imul eax,2
mov dx,histogram[eax]
mov pixeli[ebx],dx
jmp SledeciPix

VrednostNovogReda:
inc brojac
SledeciPix:
inc ebx
inc ebx
dec brojac
cmp brojac, 0
jne NoviPixeli

;// Ponovo podesavamo vrednosti brojaca i cistimo registre.
;// Promenljive M i N koristimo za smestanje vrednosti 100 i 10, jer nam dimenzije slike vise nisu potrebne.

mov ax,M
imul N
mov brojac,eax
xor eax,eax
xor ebx,ebx;// EBX je indeks od outbuffer BYTE, dok je EDI indeks od pixeli WORD.
xor edi,edi
mov ebx,outindex
mov M,100;//Ne moze neposredno div 100,i div 10 pa smestamo u M i N.
mov N,10

;// Petlja NoviIzlazni popunjava outbuffer novim 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.
;// Ako naidjemo na -1, signaliziramo kraj reda i upisujemo 0Ah u outbuffer. U tom slucaju brojac se ne smanjuje.
NoviIzlazni:

mov ax,pixeli[edi]
cmp ax,-1
je KrajReda

xor edx,edx
div M
cmp ax,0;// Proveravamo da li je broj dvocifren.
je Dvocifren
Trocifren:;// Trocifreni broj
mov cifra,3
add al,30h
mov broj[0],al;// Stotine
mov ax,dx
xor edx,edx
div N
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 N
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

;// Petlja kojom popunjavamo outbuffer.
Ispis:
xor edx,edx
xor eax,eax
mov eax,edi;// EAX sada cuva index od pixeli[] 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.
jmp Sledeci

KrajReda:
mov outbuffer[ebx],0ah
mov eax,edi;// U slucaju kada nije kraj reda, EAX cuva vrednost EDI, koju vracamo u labeli Sledeci.
            ;// Ako udjemo u labelu KrajReda smestanje se nece desiti pa to radimo ovde zbog kasnije zamene.
inc brojac;// Jer ne treba da se smanjuje za kraj reda.

Sledeci:
inc ebx
mov edi,eax
inc edi
inc edi
```

```
dec brojac
cmp brojac,0
jne NoviIzlazni

;// Upisujemo vrednost outbuffer u izlazni fajl i dobijamo izostrenu sliku.
mov  eax, outfileHandle
mov  edx, OFFSET outbuffer
mov  ecx,BUFFER_SIZE
call WriteToFile

ret
IzlazniFajl ENDP

;//MAIN
;// Ucitavamo originalnu sliku i izvrsavamo provere.
;// Zatim pozivamo procedure.
;// Na kraju zatvaramo originalnu i izostrenu sliku.
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 PixHis
call ObradaSlike
call IzlazniFajl

mov  eax,outfileHandle
call CloseFile
mov         eax,fileHandle
call CloseFile

quit:
exit
main ENDP
END main
```