IZVEŠTAJ ZA
PROJEKAT IZ RAČUNARSKE ELEKTRONIKE

Studenti:
Aleksa Srećković 2014/211
Milutin Marenović 2015/319

Predmetni professor:
prof Milan Prokin
Predmetni asistent:
prof Aleksandra Lekić

TEKST ZADATKA

Projekat 1 - skaliranje slike primenom bilinearne transformacije

Napisati program za skaliranje slike primenom bilinearne transformacije. 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 definiše se da li se vrši povećanje ili decimacija slike s puta.

REALIZACIJA PROJEKTA

Program je napisan u **asembleru**, tj programskom jeziku niskog nivoa. Korišćena je biblioteka **"Irvine32.inc"**. Za izradu projekta korišćena je literatura sa predmetnog sajta, kao i odličan sajt na kome su izlistane sve funkcije iz gore navedene biblioteke: http://programming.msjc.edu/asm/help/

Program ima dve sekcije: **.data** i **.code**

U **.data** sekciji su deklarisane sve potrebne promenljive, a u **.code** sekciji se nalazi glavni program, kao i pomoćne procedure koje se pozivaju u glavnom programu.

Tok programa

- Čitanje podataka sa standardnog ulaza - Poziva se procedura **readSTD** koja sa standardnog ulaza čita naziv ulaznog i izlaznog fajla, faktor skaliranja **s**, i proverava da li je u pitanju decimacija ili povećanje. Pročitanji parametri se smeštaju u promenljive **factorS** i **typeS**.
- Čitanje podataka iz ulaznog fajla – U promenljivu **bufferIn** smeštaju se podaci iz ulaznog fajla
- Prepisivanje karaktera P2 – Pošto se na početku svakog .pgm fajla nalazi "magični broj" P2, potrebno ga je učitati i prepisati u izlazni fajl.
- Čitanje komentara – Sledeći red datoteke predstavlja komentar koji počinje sa karakterom #. Kada se učita taj karakter, skače se na deo programa koji učitava ceo red, tako što proverava da li je pročitan karakter za nov red.
- Čitanje glavnih podataka o fajlu – Sledeći deo koda jeste čitanje sledeća tri broja koji predstavljaju širinu i visinu slike, kao i maksimalnu vrednost piksela. Napravljene su procedure **readNumber** i **printNumber** koje se pozivaju kada se čita odnosno upisuje broj u fajl.
- Određivanje širine i visine nove slike – Na osnovu promenljivih **factorS** i **typeS** određuje se nova širina odnosno visina slike. Koriste se standardne funkcije za množenje i deljenje **mul** i **div**

- Obrada – U zavisnosti od promenljive **typeS** bira se da li se skače na deo programa za decimaciju ili za povećanje. U daljem tekstu su objasnjene obe obrade.

Procedure

- **readSTD** – Procedura koja čita potrebne podatke sa standardnog ulaza
- **calculateG** – Procedura koja izračunava intenzitet novog piksela koji treba da se umetne na osnovu intenziteta najbližih piksela (**I1**, **I2**, **I3**, **I4**), faktora skaliranja **factorS**, i udaljenosti trenutnog piksela od tih najbližih piksela (**counterXr, counterYr**). Vrednost se smešta u promenljivu **Gxy** koja se kasnije upisuje u izlazni fajl.
- **getI** – Procedura koja vraća pokazivač na vrednost piksela koja se nalazi ispod piksela koji se trenutno obrađuje. Postiže se tako što se zna širina slike (**pixWidth**), i pravi se petlja koja preskoči **pixWidth** vrednosti i postavi pokazivač **buffPtr2** na tu vrednost.
- **printNumber** – Procedura koja konvertuje integer u string i ispisuje taj string u izlazni fajl
- **printSpaceString** i **printNewLine** – Kratke procedure koje ispisuju razmak / nov red u izlazni fajl. Napravljene su kako bi se smanjila veličina koda.
- **readNumber** – Procedura koja čita jedan broj iz ulaznog fajla i smešta ga u **edx** registar, a dužinu stringa smešta u **ecx** registar. U **ebx** registru treba da se nalazi pokazivač na broj koji treba da se pročita, neposredno pre poziva funkcije.

Obrada

- **Decimacija** – Postoji 4 brojača:
    - **counterHeight** – Pokazuje koliko je pročitano redova
    - **counterWidth** – Pokazuje koliko je pročitano brojeva u jednom redu. Kada **counterWidth** dostigne vrednost širine, resetuje se i **counterHeight** se poveća za 1
    - **counterSW** – Pokazuje koliko je piksela preskočeno u redu.
    - **counterSH** – Pokazuje koliko je redova preskočeno u ulaznom fajlu

  Kada **counterSW** i **counterSH** postanu jednaki faktoru **s**, to znači da smo stigli do piksela koji treba da se upiše u fajl, i tada se poziva funkcija **printNumber** za ispis intenziteta piksela u fajl

  Kada **counterHeight** postane jednak visini slike, to znači da smo stigli do kraja obrade slike i završava se program.

- **Povećanje** – Napravljeno je u vidu 4 for petlje:
    - Prva petlja prolazi kroz redove u ulaznom fajlu
    - Druga petlja u izlazni faj ispisuje **s** redova koji se umeću po visini
    - Treća petlja prolazi kroz jedan red ulaznog fajla

- Četvrta petlja umeće **s** brojeva u izlazni fajl (između dva broja koji su u ulaznom fajlu)

Dakle, složenost programa je: s^2 * width * height, odnosno newWidth * newHeight

Ispravnost programa

Ispravnost programa je proverena na 4 slike: **balloons.pgm**, **pepper.pgm**, **cassablanca.pgm** i **mona_lisa.pgm**.

Na sledećih par slika je prikazano nekoliko izvršavanja programa za različite vrednosti faktora S:


Slika mona_lisa.pgm smanjena 2 puta


Slika cassablanca.pgm povećana dva puta

KOD

```
INCLUDE Irvine32.inc
INCLUDE Macros.inc

BUFFER_SIZE = 256*256*20

.data
bufferIn BYTE BUFFER_SIZE DUP(?)
bufferOut BYTE 10 DUP(?)
bufferNumber BYTE 4 DUP(?)
bufferNum3 BYTE 4 DUP(?)
bufferNum2 BYTE 3 DUP(?)
bufferNum1 BYTE 2 DUP(?)
fileName1 BYTE 80 DUP(0)
fileName2 BYTE 80 DUP(0)
fileHandle1 HANDLE ?
fileHandle2 HANDLE ?
factorS DWORD 0
buffPtr DWORD 0
buffPtr2 DWORD 0
buffPtr3 DWORD 0
buffHelp DWORD 0
state DWORD 0       ; state = 0 na pocetku, state = 1 za citane sirine i visine,
state = 2 za citanje max vrednosti piksela,
                    ; state = 3 za upis nove sirine, visine i max vrednosti
piksela, state = 4 za glavnu obradu
typeS DWORD 0       ; typeS = 0 za povecanje, typeS = ostalo za decimaciju
pixWidth DWORD 0
newPixWidth DWORD 0
pixHeight DWORD 0
newPixHeight DWORD 0
pixMaxValue DWORD 0
string WORD 0
counterWidth WORD 0
counterHeight WORD 0
counterSW WORD 0
counterSH WORD 0
bytesRead DWORD 0
I1 DWORD 0
I2 DWORD 0
I3 DWORD 0
I4 DWORD 0
F1 DWORD 0
F2 DWORD 0
pom1 DWORD 0
Gxy DWORD 0
counterX0 DWORD 0
counterY0 DWORD 0
counterXr DWORD 0
counterYr DWORD 0
flag1 BYTE 0
flag2 BYTE 0

.code

; ********************************************************
; funkcija koja cita potrebne podatke sa standardnog ulaza,
```

```
; a to su: naziv ulaznog i izlaznog fajla, faktor skaliranja
; i da li je u pitanju decimacija ili povecanje.
; ********************************************************
readSTD PROC
    ; ucitavanje podataka o ulaznom i izlaznom fajlu
    mWrite <"Unesite naziv ulaznog fajla:", 0dh, 0ah>
    mov edx, OFFSET fileName1
    mov ecx, SIZEOF fileName1
    call ReadString
    mWrite <"Unesite naziv izlaznog fajla", 0dh, 0ah>
    mov edx, OFFSET fileName2
    mov ecx, SIZEOF fileName2
    call ReadString
    ; otvaranje ulaznog fajla preko OpenInputFile procedure
    mov edx, OFFSET fileName1
    call OpenInputFile
    mov fileHandle1, eax
    cmp eax, INVALID_HANDLE_VALUE
    jne input_ok
    mWrite <"Ne postoji ulazni fajl", 0dh, 0ah>
    jmp jmp_to_exit1
input_ok:
    ; stvaranje izlaznog fajla CreateOutputFile procedurom
    mov edx, OFFSET fileName2
    call CreateOutputFile
    mov fileHandle2, eax
    cmp eax, INVALID_HANDLE_VALUE
    jne output_ok
    mWrite <"Izlazni fajl se nije napravio", 0dh, 0ah>
    jmp jmp_to_exit1
output_ok:
    ; ucitavanje faktora s
    mWrite <"Unesite faktor skaliranja s:", 0dh, 0ah>
read_factor1:
    call ReadInt
    jno good_factor1
    mWrite <"Pogresan unos. Unesite ponovo: ", 0dh, 0ah>
    jmp read_factor1
good_factor1:
    mov factorS, eax
    ; provera da li je u pitanju povecanje ili decimacija
    mWrite <"Ako je u pitanju povecanje unesite 0, a ako je u pitanju decimacija
unesite neki broj razlicit od 0", 0dh, 0ah>
read_factor2:
    call ReadInt
    jno good_factor2
    mWrite <"Pogresan unos. Unesite ponovo: ", 0dh, 0ah>
    jmp read_factor2
good_factor2:
    mov typeS, eax
    ret
jmp_to_exit1:
    exit
readSTD ENDP

; ********************************************************
; funkcija koja u slucaju povecanja izracunava intenzitet novog
; piksela na osnovu sledecih promenljivih: I1, I2, I3, I4,
; counterXr, counterYr i factorS
; izracunati intenzitet se smesta u Gxy
```

```
; ************************************************************
calculateG PROC
    push eax
    push ecx
    push edx
    ; pom1 = f(x0, y0) * (s - yr)
    mov eax, factorS
    sub eax, counterYr
    mul I1
    mov pom1, eax
    ; pom1 = pom1 + f(x0, y0 + 1) * yr
    mov eax, counterYr
    mul I2
    add pom1, eax
    ; F1 = pom1 / s
    mov edx, 0
    mov eax, pom1
    div factorS
    mov F1, eax
    ; pom1 = f(x0 + 1, y0) * (s - yr)
    mov eax, factorS
    sub eax, counterYr
    mul I3
    mov pom1, eax
    ; pom1 = pom1 + f(x0 + 1, y0 + 1) * yr
    mov eax, counterYr
    mul I4
    add pom1, eax
    ; F2 = pom1 / s
    mov edx, 0
    mov eax, pom1
    div factorS
    mov F2, eax
    ; pom1 = F1 * (s - xr)
    mov eax, factorS
    sub eax, counterXr
    mul F1
    mov pom1, eax
    ; pom1 = pom1 + F2 * xr
    mov eax, counterXr
    mul F2
    add pom1, eax
    ; Gxy = pom1 / s
    mov edx, 0
    mov eax, pom1
    div factorS
    mov Gxy, eax
    pop edx
    pop ecx
    pop eax
    ret
calculateG ENDP

; ************************************************************
; funkcija vraca pokazivac (buffPtr2) na intenzitet piksela koji se nalazi
; ispod piksela na koji trenutno pokazuje buffPtr
; ************************************************************
getI PROC
    push eax
    push ecx
```

```
    push edx
    ; ecx je sirina slike, petlja se vrti dok ecx ne postane 0
    ; buffPtr2 je pokazivac koji ce pokazivati na piksel ispod
    ; piksela na koji pokazuje buffPtr
    mov ecx, pixWidth
    mov eax, buffPtr
    mov buffPtr2, eax
    ; petlja za prolazak kroz fajl i pronalazak zeljenog piksela
getI_loop:
    cmp ecx, 0
    je getI_end
    mov ebx, 0
getI_loop1:
    mov edx, buffPtr2
    mov al, [edx]
    call IsDigit
    jnz getI_end1
    inc buffPtr2
    jmp getI_loop1
getI_end1:
    ; sada je procitan intenzitet piksela, i sada preskacemo razmak i
    ; eventualno nov red
    inc buffPtr2
    mov edx, buffPtr2
    mov al, [edx]
    cmp al, 0ah
    je getI_skip
getI_skip_end:
    dec ecx
    jmp getI_loop
getI_skip:
    inc buffPtr2
    jmp getI_skip_end
getI_end:
    pop edx
    pop ecx
    pop eax
    ret
getI ENDP

; ************************************************************
; funkcija koja konvertuje integer u string i ispisuje na izlaz
; vrednost koja se nalazi u eax registru; koriste se
; registri ebx, ecx i edx koji se zbog toga stavljaju na stek.
; ************************************************************
printNumber PROC
    push ebx
    push ecx
    push edx
    mov ebx, 10
    mov ecx, 3
int_to_string:
    mov edx, 0
    div ebx
    add dl, 30h
    mov bufferNumber[ecx], dl
    dec ecx
    cmp eax, 0
    jnz int_to_string
    mov eax, fileHandle2
```

```
    mov ebx, 3
    sub ebx, ecx
    mov ecx, ebx
    mov ebx, 4
    sub ebx, ecx
    mov edx, OFFSET bufferNumber
    add edx, ebx
    call WriteToFile
    pop edx
    pop ecx
    pop ebx
    ret
printNumber ENDP

; ************************************************************
; funkcija koja ispisuje prazan string u izlaznu datoteku
; ************************************************************
printSpaceString PROC
    mov string, ' '
    mov eax, fileHandle2
    mov edx, OFFSET string
    mov ecx, 1
    call WriteToFile
    ret
printSpaceString ENDP

; ************************************************************
; funkcija koja ispisuje karakter za nov red u izlazni fajl
; ************************************************************
printNewLine PROC
    mov string, 0ah
    mov eax, fileHandle2
    mov edx, OFFSET string
    mov ecx, 1
    call WriteToFile
    ret
printNewLine ENDP

; ************************************************************
; funkcija koja cita jedan broj iz ulaznog fajla i smesta
; vrednost u edx registar, a duzinu stringa u ecx registar
; argument funkcije je pokazivac na broj koji treba da se procita
; a smesten je u ebx registar
; ************************************************************
readNumber PROC
    push eax
    mov ecx, 0
read_number_loop:
    mov edx, ebx
    mov al, [edx]
    call IsDigit
    jnz end_read_number_loop
    mov bufferNum3[ecx], al
    inc ecx
    inc ebx
    jmp read_number_loop
end_read_number_loop:
    ; proverava se da li je broj dvocifren
    cmp ecx, 2
    je smaller100
```

```
    cmp ecx, 1
    je smaller10
    mov edx, OFFSET bufferNum3
end_smaller:
    ; preskace se razmak, proverava se da li je sledeci karakter za novi red,
    ; i ako jeste preskace se i on
    inc ebx
    mov al, [ebx]
    cmp al, 0ah
    je skip_new_line
    jmp end_read_number
smaller100:
    mov al, bufferNum3[0]
    mov bufferNum2[0], al
    mov al, bufferNum3[1]
    mov bufferNum2[1], al
    mov edx, OFFSET bufferNum2
    jmp end_smaller
smaller10:
    mov al, bufferNum3[0]
    mov bufferNum1[0], al
    mov edx, OFFSET bufferNum1
    jmp end_smaller
skip_new_line:
    inc ebx
end_read_number:
    pop eax
    ret
readNumber ENDP

; ************************************************************
; main funkcija
; ************************************************************
main PROC
    ; poziva se funkcija koja cita potrebne podatke sa standardnog ulaza
    call readSTD

    ; citanje ulaznog fajla
    mov eax, fileHandle1
    mov edx, OFFSET bufferIn
    mov ecx, BUFFER_SIZE
    call readFromFile
    jnc read_ok
    mWrite <"Greska pri citanju fajla", 0dh, 0ah>
    jmp close_files
read_ok:

    ; prepisivanje karaktera 'P2'
    mov ecx, 3
    mov esi, OFFSET bufferIn
    mov edi, OFFSET bufferOut
    rep movsb
    mov eax, fileHandle2
    mov edx, OFFSET bufferOut
    mov ecx, 3
    call WriteToFile
    mov buffPtr, esi
    mov state, 1
    jc error
```

```
    ; loop koji proverava koji deo teksta se cita i/ili upisuje u fajl
check_state:
    ; provera da li je sledeci karakter '#' odnosno komentar i ako jeste, skace se
na deo programa
    ; za citanje komentara
    mov edx, buffPtr
    mov al, [edx]
    cmp al, '#'
    je comment_read
    ; provera da li je u pitanju citanje sirine i visine slike
    cmp state, 1
    je read_width
    ; provera da li je u pitanju citanje maksimalne vrednosti piksela
    cmp state, 2
    je read_max_value
    ; provera da li je vreme za pisanje sirine, visine slike i max vrednosti
piksela u output fajl
    cmp state, 3
    je write_width
    ; provera da li je u pitanju prelazak na obradu
    cmp typeS, 0
    je povecanje
    jmp decimacija

    ; ************************************************************
    ; povecanje
    ; ************************************************************
povecanje:
    ; priprema brojaca
    mov counterX0, 0
    mov counterY0, 0
    mov counterXr, 0
    mov counterYr, 0
    mov eax, buffPtr
    mov buffPtr3, eax
    ; glavna petlja
loop1:
    mov eax, pixHeight
    cmp counterY0, eax
    je close_files ; ako je counterY0 jednak visini slike, dosli smo do kraja
obrade
    mov eax, factorS
    cmp counterYr, eax
    je reset_counterYr ; ako je counterYr jednak broju s, dosli smo do novog reda u
ulaznom fajlu
return1:
    mov eax, pixWidth
    cmp counterX0, eax
    je reset_counterX0 ; ako je counterX0 jednak sirini slike, dosli smo do kraja
reda
return2:
    mov eax, factorS
    cmp counterXr, eax
    je reset_counterXr ; ako je counterXr jednak broju s, dosli smo do novog
piksela u jednom redu u ulaznom fajlu
return3:
    ; ako su counterXr i counterYr jednaki 0, onda ne moramo da racunamo
intenzitet, vec samo da prepisemo broj
    cmp counterXr, 0
    je print_number_condition1
```

```
    ; ako smo na pocetku reda, menjamo vrednosti buffPtr i buffPtr3
return5:
    cmp flag2, 1
    je change_buff

return4:
    ; pronalazenje najblizih piksela
    call getI
    mov ebx, buffPtr
    call readNumber
    call ParseDecimal32
    mov I1, eax
    mov buffHelp, ebx
    call readNumber
    call ParseDecimal32
    mov I3, eax
    mov ebx, buffPtr2
    call readNumber
    call ParseDecimal32
    mov I2, eax
    call readNumber
    call ParseDecimal32
    mov I4, eax
    call calculateG
    mov eax, Gxy
    call printNumber
    call printSpaceString
    mov eax, buffHelp
    inc counterXr
    jmp loop1

change_buff:
    mov flag2, 0
    cmp counterYr, 0
    je change_buff3
    mov eax, buffPtr3
    mov buffPtr, eax
label3:
    jmp return4

change_buff3:
    mov eax, buffPtr
    mov buffPtr3, eax
    jmp label3

reset_counterYr:
    call printNewLine
    inc counterY0
    mov counterYr, 0
    mov counterX0, 0
    mov counterXr, 0
    jmp loop1

reset_counterX0:
    inc counterYr
    mov counterX0, 0
    mov counterXr, 0
    mov flag2, 1
    jmp loop1
```

```
reset_counterXr:
    inc counterX0
    mov counterXr, 0
    mov ebx, buffPtr
    call readNumber
    mov buffPtr, ebx
    jmp loop1

print_number_condition1:
    cmp counterYr, 0
    je print_number
    jmp return5

    ; citanje broja i prepisivanje istog u izlazni fajl, jer nije
    ; potrebno racunanje intenziteta
print_number:
    mov ebx, buffPtr
    call readNumber
    mov eax, fileHandle2
    call WriteToFile
    call printSpaceString
    inc counterXr
    jmp loop1

    ; ************************************************************
    ; decimacija
    ; ************************************************************
decimacija:
    ; priprema brojaca
    mov counterSW, 0
    mov counterSH, 0
    mov eax, pixHeight
    mov counterHeight, ax
    mov eax, pixWidth
    mov counterWidth, ax
read_loop:
    ; citanje broja i provera da li broj treba da se ispise
    mov ebx, buffPtr
    call readNumber
    mov buffPtr, ebx
    cmp counterSW, 0
    je condition1
continue1:
    ; smanjuje se brojac
    dec counterSW
continue2:
    ; smanje se brojac counterWidth i proverava se da li je doslo do kraja reda
    dec counterWidth
    cmp counterWidth, 0
    je new_line
continue3:
    ; provera da li je doslo do kraja datoteke
    cmp counterHeight, 0
    je end_reading
    jmp read_loop

condition1:
    mov eax, factorS
    mov counterSW, ax
    cmp counterSH, 0
```

```
    je write_number
    jmp continue1

write_number:
    mov eax, fileHandle2
    call WriteToFile
    call printSpaceString
    mov ecx, factorS
    mov counterSW, cx
    jmp continue1

new_line:
    cmp counterSH, 0
    je set_counterSH
continue4:
    dec counterSH
    dec counterHeight
    mov eax, pixWidth
    mov counterWidth, ax
    jmp continue3

set_counterSH:
    mov eax, factorS
    mov counterSH, ax
    mov counterSW, ax
    jmp continue4

end_reading:
    jmp close_files

    ; citanje sirine i visine slike
read_width:
    mov ecx, 0 ; brojac koliko karaktera ima ucitana sirina
    ; loop za citanje sirine
width_loop:
    mov edx, buffPtr
    mov al, [edx]
    call IsDigit
    jnz end_width_loop
    mov bufferNumber[ecx], al
    inc ecx
    inc buffPtr
    jmp width_loop
    ; procitana je sirina, ali u vidu niza karaktera pa je potrebno konvertovati je
u integer
end_width_loop:
    mov edx, OFFSET bufferNumber
    call ParseDecimal32
    mov pixWidth, eax
    mov ecx, 0
    ; citanje praznog karaktera izmedju vrednosti za visinu i sirinu
    inc buffPtr
height_loop:
    mov edx, buffPtr
    mov al, [edx]
    call IsDigit
    jnz end_height_loop
    mov bufferNumber[ecx], al
    inc ecx
    inc buffPtr
```

```
    jmp height_loop
    ; procitana je visina, ali u vidu niza karaktera pa je potrebno konvertovati je
u integer
end_height_loop:
    mov edx, OFFSET bufferNumber
    call ParseDecimal32
    mov pixHeight, eax
    ; citanje karaktera za nov red
    inc buffPtr
    mov state, 2
    jmp check_state

    ; citanje maksimalne vrednosti piksela
read_max_value:
    mov ecx, 0
    ; loop za citanje maksimalne vrednosti piksela
max_value_loop:
    mov edx, buffPtr
    mov al, [edx]
    call IsDigit
    jnz end_max_value_loop
    mov bufferNumber[ecx], al
    inc ecx
    inc buffPtr
    jmp max_value_loop
    ; procitana je max vrednost ali u vidu niza karaktera pa je potrebno
konvertovati je u integer
end_max_value_loop:
    mov edx, OFFSET bufferNumber
    call ParseDecimal32
    mov pixMaxValue, eax
    ; citanje karaktera za novi red
    inc buffPtr
    mov state, 3
    jmp check_state

    ; upis sirine i visine slike, kao i max vrednosti piksela u output fajl
write_width:
    ; racunanje novih visina i sirina
    cmp typeS, 0
    je calculate
    mov edx, 0
    mov eax, pixWidth
    div factorS
    mov newPixWidth, eax
    mov edx, 0
    mov eax, pixHeight
    div factorS
    mov newPixHeight, eax
    jmp write_width1
calculate:
    mov eax, pixWidth
    mul factorS
    mov newPixWidth, eax
    mov eax, pixHeight
    mul factorS
    mov newPixHeight, eax
write_width1:
    ; ispis sirine u izlazni fajl
    mov eax, newPixWidth
```

```
    call printNumber
    call printSpaceString
    ; ispis duzine u izlazni fajl
    mov eax, newPixHeight
    call printNumber
    call printNewLine
    ; ispis maksimalne vrednosti piksela
    mov eax, pixMaxValue
    call printNumber
    call printNewLine
    mov state, 4
    jmp check_state

    ; loop za citanje komentara iz ulaznog fajla
comment_read:
    inc buffPtr
    mov edx, buffPtr
    mov al, [edx]
    cmp al, 0ah
    jne comment_read
    inc buffPtr
    jmp check_state

    ; greska pri citanju / upisu u fajl
error:
    mWrite <"Greska pri citanju/upisu u fajl", 0dh, 0ah>

    ; zatvaranje datoteka
close_files:
    mov eax, fileHandle1
    call CloseFile
    mov eax, fileHandle2
    call CloseFile

    ; izlazak iz programa
jmp_to_exit:
    exit
main ENDP
END main
```