УНИВЕРЗИТЕТ У БЕОГРАДУ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Катедра за електронику
Рачунарска електроника

Извештај – Пројекат 17

Александар Арсовић 0010/2014
Алекса Аничић 0339/2013

Садржај

1 Поставка пројекта **3**

2 Кратак опис пројектног кода **4**

3 Наше процедуре **5**
3.1 STRCAT 5
3.2 getFilename 5
3.3 stringToInt 5
3.4 intToString 5
3.5 FillArray 5
3.6 compareBuffers 5
3.7 OpenFile 5
3.8 ReadNumbersFromFile 5
3.9 printWinnerNumbers 5
3.10 printPobednik 5
3.11 printPreostalePartije 5

4 Код пројектног задатка **6**

1 Поставка пројекта

Потребно је направити игрицу *Win More* у којој је циљ да играчи погоде бројеве које ће компјутер да избаци. Игрица има 5 партија и у свакој компјутер избацује 10 насумичних бројева.

Сваки играч уноси 6 бројева и потребно је да погоди бар 5 да би добио неке поене. Када сви играчи унесу бројеве, компјутер избацује својих 10 бројева са периодом од 1s. После 5 партија рачунају се укупни освојени поени свих играча и одређује се ко је победник.

Све комбинације, међурезултате и укупне резултате рачунар чува у фајловима за сваког играча посебно.

2 Кратак опис пројектног кода

Прво се у конзоли исписује порука за унос броја играча. Учитава се број играча у бафер и пребацује у целобројну вредност из низа бајтова.

Прва петља је прављење фајлова. У њој се за сваког играча направи фајл и у њега упише за кога је тај фајл.

После тога иде петља за понављање партија. У једној њеној итерацији пролази се кроз све играче. Прво се пита сваки играч да унесе бројеве, а ако су у погрешном опсегу од њега се тражи да поново унесе бројеве. После уношења бројева у фајл сваког играча се уписује одговарајућа комбинација. Фајл се отвара позивањем процедуре *OpenFile* која је наша функција која отвара фајл за упис и ставља курсор на крај фајла.

Када сви играчи унесу своје бројеве, компјутер генерише свој низ и исписује га у конзолу. Исписује се коришћењем наше процедуре *printWinnerNumbers* која исписује низ од 10 бројева један по један са периодом од 1s. То је постигнуто коришћењем функције *Sleep*.

После добијања победничког низа бројева одређује се број освојених поена за сваког играча и уписује се у његов фајл заједно са добитном комбинацијом.

Када прођу све партије рачунају се укупни поени за сваког играча и одређује се победник. На крају се исписује који играч је имао највише поена.

3 Наше процедуре

3.1 STRCAT

Врши конкатенацију стрингова из меморије и уписује назад у меморију.

3.2 getFilename

У променљиву *filename* уписује назив фајла тренутног играча.

3.3 stringToInt

Пребацује стринг из *Buffer* у низ целих бројева у *intBuffer*.

3.4 intToString

Низ из променљиве *intBuffer10* уписује у стринг *stringBuffer*.

3.5 FillArray

Генерише насумичан низ 10 бројева.

3.6 compareBuffers

Одређује колико је корисник бројева погодио поредећи *intBuffer* и *intBuffer10*.

3.7 OpenFile

Отвара фајл за писање и поставља курсор на крај фајла.

3.8 ReadNumbersFromFile

Учитава или низ бројева које је корисник унео или број поена који је освојио у одређеној партији. У зависности од променљиве *pretraga* учитава поене или унесене бројеве.

3.9 printWinnerNumbers

Исписује извучене бројеве са периодом од 1s.

3.10 printPobednik

Исписује који играч је имао највише поена.

3.11 printPreostalePartije

Исписује преостале партије у горњем десном ћошку.

4 Код пројектног задатка

```
INCLUDE Irvine32.inc
INCLUDE macros.inc

BufSize = 80
FileBuffSize = 700

.data
endl EQU <0dh,0ah>                              ; end of line sequence

brojIgraca LABEL BYTE
        BYTE "Unesite broj igraca. ", endl
brojIgracaSize DWORD ($-brojIgraca)

igracTekst BYTE "Igrac "
brojeviIgraca LABEL BYTE
        BYTE " unesite svoje brojeve:", endl
brojeviIgracaSize DWORD ($-brojeviIgraca)

tajmer DWORD 1000
filename BYTE 0, 33 DUP(0)
filenameBase BYTE "PlayerSum_"
filenameEnd BYTE ".txt"
intBuffer BYTE 6 DUP(0)
intBuffer10 BYTE 10 DUP(0)
brojPogodaka BYTE 1 DUP(0)
osvojeniPoeni BYTE 1 DUP (0)
ukupniPoeni BYTE 0
pobednik BYTE 1
pobednikPoeni BYTE 0
nulaPoena BYTE 30h, 0dh, 0ah
desetPoena BYTE  31h, 30h, 0dh, 0ah
pedesetPoena BYTE 35h, 30h, 0dh, 0ah
stringBuffer BYTE 31 DUP(0)
stringBufferSize DWORD 0
idIgraca BYTE "ID igraca:"
crtice BYTE 20 DUP('-'), endl
brPartije BYTE "Broj partije:"
kombinacija BYTE "Kombinacija:"
izvuceniBrojevi BYTE "Izvuceni brojevi:"
dobijeniBrojevi BYTE "Izvuceni brojevi:", endl
brojPoena BYTE "Broj osvojenih poena:"
ukupnoPoena BYTE "Ukupno osvojenih poena:"
pobednikTekst BYTE "Pobednik je igrac broj "
preostalePartijeTekst BYTE "Preostale partije: "
preostalePartije BYTE ?
byteIspis BYTE ?
```

```
ponovoUnesiBrojeveTekst BYTE "Ponovo unesi brojeve:", endl
ponoviBrojeve BYTE 0


endline BYTE endl
pretraga BYTE 'K'
pretragaOffset DWORD 11


; Console handles
outputHandle HANDLE 0       ; handle to standard output device
inputHandle  HANDLE 0       ; handle to std input device
bytesWritten  DWORD ?        ; number of bytes written

ptrConsoleInfo LABEL WORD
consoleInfo CONSOLE_SCREEN_BUFFER_INFO <>
xCursor WORD ?
yCursor WORD ?
ptrCursorPosition LABEL WORD
cursorPosition COORD <>
kursorNaPocetak COORD <0,0>


buffer BYTE BufSize DUP(?)
fileBuffer BYTE FileBuffSize DUP(?)
bytesRead    DWORD ?

brojacIgraca LABEL BYTE
brIgr    DWORD ?
trenutniIgrac BYTE 0
brojPartije BYTE ?

; Files
fileHandle HANDLE ?


.code
main PROC
        ;call compareBuffers

  ; Get the console output and input handle:
        INVOKE GetStdHandle, STD_OUTPUT_HANDLE
        mov outputHandle,eax
        INVOKE GetStdHandle, STD_INPUT_HANDLE
        mov inputHandle,eax

        call Clrscr
    ; Write a string to the console:
```

```
        INVOKE WriteConsole,
          outputHandle,          ; console output handle
          ADDR brojIgraca,               ; string pointer
          brojIgracaSize,                        ; string length
          ADDR bytesWritten,     ; returns num bytes written
          0                                      ; not used

    ;Read number of players
        INVOKE ReadConsole, inputHandle, ADDR buffer,
          BufSize, ADDR bytesRead, 0

        ;kod za citanje broja igraca iz BUFFERA u ebx
        xor eax,eax
        xor ebx,ebx
        mov esi, OFFSET buffer
        LODSB
        mov ebx, eax
        sub ebx, 30h
        LODSB
        cmp eax, 0Dh
        je  skok1
        imul ebx, 10
        sub eax, 30h
        add ebx, eax
        ;prebacimo broj igraca iz ebx u brIgr
skok1:
        mov brIgr, ebx


        ;ovde loopujemo, svaki put kada pravimo novi fajl.
        ;Upisujemo ID igraca u svaki fajl
pravljenjeFajlova:
        inc trenutniIgrac

        call getFilename

        mov     edx,OFFSET filename
        call    CreateOutputFile
        mov     fileHandle,eax
        mov edx, OFFSET idIgraca
        mov ecx, SIZEOF idIgraca
        call WriteToFile
        mov eax, fileHandle
        add trenutniIgrac, 30h
        mov edx,  OFFSET trenutniIgrac
        mov ecx, SIZEOF trenutniIgrac
        call WriteToFile
        sub trenutniIgrac, 30h
```

```
        mov eax, fileHandle
        mov edx, OFFSET endline
        mov ecx, SIZEOF endline
        call WriteToFile
        mov eax, fileHandle
        call CloseFile


        mov al, trenutniIgrac
        cmp al, brojacIgraca
        jne pravljenjeFajlova

        ;upis brojeva u fajlove
        mov brojPartije, 1h

sledecaPartija:
        mov trenutniIgrac, 0h
        call Clrscr
        call printPreostalePartije

;; Upisujemo brojeve igraca u njihove fajlove
upisivanjeUnetihBrojeva:
        inc trenutniIgrac
        call getFilename

        INVOKE WriteConsole, outputHandle, ADDR igracTekst,
           LENGTHOF igracTekst ,       ADDR bytesWritten, 0

        add trenutniIgrac, 30h
        INVOKE WriteConsole, outputHandle, ADDR trenutniIgrac,
           1,       ADDR bytesWritten, 0
        sub trenutniIgrac, 30h

        INVOKE WriteConsole, outputHandle, ADDR brojeviIgraca,
           brojeviIgracaSize ,       ADDR bytesWritten, 0
        ;Read number of players

ponoviUnos:
        INVOKE ReadConsole, inputHandle, ADDR buffer, BufSize,
           ADDR bytesRead, 0
        call stringToInt
        cmp ponoviBrojeve, 1
        je ponoviUnos

        call OpenFile
        mov eax, fileHandle
        mov edx, OFFSET crtice
        mov ecx, SIZEOF crtice
```

```
        call WriteToFile
        mov eax, fileHandle
        mov edx, OFFSET brPartije
        mov ecx, SIZEOF brPartije
        call WriteToFile
        mov eax, fileHandle
        add brojPartije, 30h
        mov edx, OFFSET brojPartije
        mov ecx, SIZEOF brojPartije
        call WriteToFile
        sub brojPartije, 30h
        mov eax, fileHandle
        mov edx, OFFSET endline
        mov ecx, SIZEOF endline
        call WriteToFile
        mov eax, fileHandle
        mov edx, OFFSET kombinacija
        mov ecx, SIZEOF kombinacija
        call WriteToFile
        mov eax, fileHandle
        mov edx, OFFSET buffer
        mov ecx, bytesRead
        call WriteToFile
        mov eax, fileHandle
        call CloseFile

        mov al, trenutniIgrac
        cmp al, brojacIgraca
        jne upisivanjeUnetihBrojeva

        ;generisemo random brojeve
        call Randomize
        call FillArray
        call intToString
        mov  trenutniIgrac, 0h


        ;;; Ispisujemo izvucenu kombinaciju
        call printWinnerNumbers


        ; za svakog igraca trazimo njegovu kombinaciju iz fajla
        ; poredimo i upisujemo bodove i izvucenu komb.
upisOsvojenihPoena:
        inc trenutniIgrac
        call readNumbersFromFile
        call compareBuffers
```

```
        call getFilename

        call OpenFile

        mov eax, fileHandle
        mov edx, OFFSET izvuceniBrojevi
        mov ecx, SIZEOF izvuceniBrojevi
        call WriteToFile


        mov eax, fileHandle
        mov edx, OFFSET stringBuffer
        mov ecx, stringBufferSize
        call WriteToFile

        mov eax, fileHandle
        mov edx, OFFSET brojPoena
        mov ecx, SIZEOF brojPoena
        call WriteToFile

        mov eax, fileHandle
        mov edx, OFFSET nulaPoena
        mov ecx, 3
        xor ebx, ebx
        mov bl, osvojeniPoeni
        cmp bl, 10
        jne pedeset
        mov edx, OFFSET desetPoena
        mov ecx, 4
        jmp ispisaniOsvojeniPoeni
pedeset:
        jna ispisaniOsvojeniPoeni
        mov edx, OFFSET pedesetPoena
        mov ecx, 4

ispisaniOsvojeniPoeni:
        call WriteToFile

        mov eax, fileHandle
        call CloseFile

        mov al, trenutniIgrac
        cmp al, brojacIgraca
        jne upisOsvojenihPoena


        inc brojPartije
        mov al, brojPartije
```

```
        cmp al, 6
        jne sledecaPartija



;;; Podesavanja za proceduru readNumersFromFile za pretragu
    rezultata iz pojedinacnih rundi
        mov al, 's'
        mov pretraga, al
        mov eax, 14
        mov pretragaOffset, eax

        mov trenutniIgrac, 1
        mov pobednik, 1

racunaj2:
        mov brojPartije, 0
        mov ukupniPoeni, 0
        mov intBuffer10, 0


racunaj:
        inc brojPartije
        call readNumbersFromFile
        mov al, intBuffer
        add ukupniPoeni,al
        add intBuffer10, al
        mov eax, 5
        cmp brojPartije, al
        jne racunaj

        call getFilename
        call OpenFile
        mov eax, fileHandle
        mov edx, OFFSET crtice
        mov ecx, SIZEOF crtice
        call WriteToFile

        mov eax, fileHandle
        mov edx, OFFSET ukupnoPoena
        mov ecx, SIZEOF ukupnoPoena
        call WriteToFile

        call intToString
        mov eax, fileHandle
        mov edx, OFFSET stringBuffer
        mov ecx, 2
        call WriteToFile
```

```
        mov eax, fileHandle
        call CloseFile

        ;provjeri ko ima najvise
        mov bl, pobednikPoeni
        mov al, ukupniPoeni
        cmp al, bl
        jna     nijeVece
        mov pobednikPoeni, al
        xor ebx, ebx
        mov bl, trenutniIgrac
        mov pobednik, bl
nijeVece:
        inc trenutniIgrac
        mov al, trenutniIgrac
        cmp al, brojacIgraca
        jna racunaj2



        ;;Ispisivanje pobednika
        call printPobednik



        INVOKE ExitProcess,0
main ENDP




;procedures
;-----------------------------------------------------------------
;KONKATENACIJA
STRCAT PROC USES eax ecx

                ; Dodji do kraja stringa
                mov eax, 0
                mov ecx, LENGTHOF filename
                cld
                repne SCASB
                dec edi
                mov ecx, ebx

                rep MOVSB
```

```
                mov bl,0
                mov [edi],bl

ret
STRCAT ENDP


;-----------------------------------------------------------------
getFilename PROC USES edi esi ebx eax ecx

        mov filename, 0
        mov edi, offset filename
        mov esi, offset filenameBase
        mov ebx, LENGTHOF filenameBase
        call STRCAT

        add trenutniIgrac, 30h
        mov edi, offset filename
        mov esi, offset trenutniIgrac
        mov ebx, LENGTHOF trenutniIgrac
        call STRCAT
        sub trenutniIgrac, 30h

        mov edi, offset filename
        mov esi, offset filenameEnd
        mov ebx, LENGTHOF filenameEnd
        call STRCAT

ret
getFilename ENDP

;-----------------------------------------------------------------
stringToInt PROC USES  eax ebx esi edi ecx
        mov ponoviBrojeve, 0
        mov esi, OFFSET buffer
        mov edi, OFFSET intBuffer

pocetak:
        xor eax,eax
        xor ebx,ebx

        LODSB
        cmp al, 0Dh
        je kraj2
        cmp al, 20h
        je pocetak

        mov ebx, eax
```

```
        sub ebx, 30h
        LODSB
        cmp eax, 20h
        je upis
        cmp al, 0Dh
        je kraj

        imul ebx, 10
        sub eax, 30h
        add ebx, eax

upis:
        ;; radi se provera brojeva za unete brojeve korisnika
        mov al, 's'
        cmp al, pretraga
        je neProveravajVelicinu
        cmp bl, 0
        jl ponovoUnesiBrojeve
        cmp bl,77
        jg ponovoUnesiBrojeve

neProveravajVelicinu:
        mov [edi], bl
        inc edi
        jmp pocetak

        kraj:
        mov [edi], bl


        kraj2:
        ret

ponovoUnesiBrojeve:
        INVOKE WriteConsole, outputHandle, ADDR
           ponovoUnesiBrojeveTekst, LENGTHOF
           ponovoUnesiBrojeveTekst ,     ADDR bytesWritten, 0
        mov ponoviBrojeve, 1
        ret


stringToInt ENDP

;-----------------------------------------------------
intToString PROC USES eax ecx esi edi ebx
        mov stringBufferSize,0
        mov ecx, 10
        xor eax,eax
```

```
        mov bl, 0Ah
        mov edi, OFFSET stringBuffer
        mov esi, OFFSET intBuffer10
sledeciBroj:
        xor eax, eax
        LODSB
        div bl

        cmp al, 0
        je      manjeOdDeset
        add al, 30h
        mov [edi], al
        inc edi
        inc stringBufferSize
manjeOdDeset:
        add ah, 30h
        mov [edi], ah
        inc edi
        inc stringBufferSize
        mov ah, 20h
        mov [edi], ah
        inc edi
        inc stringBufferSize
        dec ecx
        jnz sledeciBroj

        dec edi
        mov ah, 0Dh
        mov [edi],ah
        inc edi
        inc stringBufferSize
        mov ah, 0Ah
        mov [edi],ah
        ;inc stringBufferSize

ret
intToString ENDP


;-----------------------------------------------------
FillArray PROC USES eax edi ecx edx
;       pArray:PTR DWORD, ; pointer to array;
;       Count:DWORD, ; number of elements;
;       LowerRange:SDWORD, ; lower range
;       UpperRange:SDWORD ; upper range
```

```
;
; Fills an array with a random sequence of 32-bit signed
; integers between LowerRange and (UpperRange - 1).
; Returns: nothing
;-----------------------------------------------------------
        mov edi, OFFSET intBuffer10 ; EDI points to the array
        mov ecx,10 ; loop counter
        mov edx,1
        sub edx,0    ; EDX = absolute range (0..n)
        cld ; clear direction flag
L1: mov eax,edx ; get absolute range
        call RandomRange
        stosb ; store EAX into [edi]
        loop L1
ret
FillArray ENDP



;-----------------------------------------------------------
compareBuffers PROC USES eax esi edi ebx ecx
        mov brojPogodaka, 0                                ;clear
            brojPogodaka
        mov ecx, 7
            ;counter to go trough 6 buffer members, starts from 7
             because it dec's at the start of the loop
        mov esi, OFFSET intBuffer                  ;we use esi as a
             pointer to intBuffer(6)
        dec esi
            ;we dec here, so that we can inc in loop from start

proveriSledeciOd6:
        inc esi
            ;here we inc
        mov ebx, 10
            ;counter to help go through all of the buffer10,
            resets for every int of buffer6
        dec ecx
        jz  gotovaProvera
        mov edi, OFFSET intBuffer10                ;set EDI to show
             at the start of buffer10
proveriSledeciOd10:
        CMPSB
            ;compares [ESI] & [EDI], and increments them both, we
             dont want esi incremented tho
        je      postoji
        dec esi
        dec ebx
```

```
        jz  proveriSledeciOd6
        jmp proveriSledeciOd10

postoji:
        dec esi
            ;because CMPSB auto inc'd esi, we dec it here
        inc brojPogodaka
        jmp proveriSledeciOd6

gotovaProvera:
        mov osvojeniPoeni,0
        mov bl, 5
        cmp brojPogodaka, bl
        jne sest
        mov osvojeniPoeni, 10
        jmp odredjenBrojPoena
sest: jna odredjenBrojPoena
        mov osvojeniPoeni, 50
odredjenBrojPoena:



ret
compareBuffers ENDP


;-----------------------------------------------------
OpenFile PROC USES ebx edx ecx edi eax
        ;Opens existing file for appending
        INVOKE CreateFile,ADDR filename,GENERIC_WRITE,
            DO_NOT_SHARE,NULL,OPEN_EXISTING,
            FILE_ATTRIBUTE_NORMAL,0
        cmp eax,INVALID_HANDLE_VALUE
        je izadji
        mov     fileHandle,eax
        INVOKE SetFilePointer,  fileHandle,     0,      0,
            FILE_END
ret
izadji:
        ;;; Ovde nikada ne treba da udje
        mov eax, OFFSET filename
        mov ebx, LENGTHOF filename
        INVOKE ExitProcess,0

OpenFile ENDP


;-----------------------------------------------------
```

```
ReadNumbersFromFile PROC USES ebx edx ecx edi eax

        call getFilename
        mov edx, OFFSET filename
        call OpenInputFile
        mov fileHandle, eax

        mov bl, brojPartije
        mov edx,OFFSET fileBuffer
        mov ecx,FileBuffSize
        call ReadFromFile

        mov eax, fileHandle
        call CloseFile


        xor eax,eax
        mov al, pretraga
        mov edi, OFFSET fileBuffer
        mov ecx, SIZEOF fileBuffer
sledece:
        repne SCASB
        dec bl
        jnz sledece

        add edi, pretragaOffset
        mov esi, edi
        mov edi, OFFSET buffer

nijeEnter:
        MOVSB
        mov eax, 0Ah
        cmp [esi], eax
        jne nijeEnter

        call stringToInt
        ret
ReadNumbersFromFile ENDP


printWinnerNumbers PROC USES esi eax
        ;Postavljanje kursora na sredinu (otprilike)
        INVOKE GetConsoleScreenBufferInfo, outputHandle, ADDR
            consoleInfo
        mov esi, OFFSET ptrConsoleInfo
        mov ax, [esi]
        shr ax, 1
```

```
        sub ax, 15
        mov ptrCursorPosition, ax
        mov ax, [esi+6]
        mov esi, OFFSET ptrCursorPosition
        mov [esi+2], ax
        INVOKE SetConsoleCursorPosition, outputHandle,
            cursorPosition

        INVOKE WriteConsole, outputHandle, ADDR dobijeniBrojevi,
            LENGTHOF dobijeniBrojevi,      ADDR bytesWritten, 0
        ;Postavljanje kursora na sredinu (otprilike)
        INVOKE GetConsoleScreenBufferInfo, outputHandle, ADDR
            consoleInfo
        mov esi, OFFSET ptrConsoleInfo
        mov ax, [esi]
        shr ax, 1
        sub ax, 15
        mov ptrCursorPosition, ax
        mov ax, [esi+6]
        mov esi, OFFSET ptrCursorPosition
        mov [esi+2], ax
        INVOKE SetConsoleCursorPosition, outputHandle,
            cursorPosition


        mov esi, OFFSET stringBuffer
sledeciKarakter:
        LODSB
        mov byteIspis, al
        INVOKE WriteConsole, outputHandle, addr byteIspis, 1,
            ADDR bytesWritten, 0
        mov al, byteIspis
        cmp al, 0Ah
        je kraj
        cmp eax, 20h
        jne sledeciKarakter
        ;invoke sleep, 1000
        jmp sledeciKarakter
kraj:
;INVOKE WriteConsole, outputHandle, addr byteIspis, 1, ADDR
    bytesWritten, 0
ret
printWinnerNumbers ENDP

printPobednik PROC USES eax esi
        ;Postavljanje kursora na sredinu (otprilike)
        INVOKE GetConsoleScreenBufferInfo, outputHandle, ADDR
            consoleInfo
```

```
        mov esi, OFFSET ptrConsoleInfo
        mov ax, [esi]
        shr ax, 1
        sub ax, 15
        mov ptrCursorPosition, ax
        mov ax, [esi+6]
        mov esi, OFFSET ptrCursorPosition
        mov [esi+2], ax
        INVOKE SetConsoleCursorPosition, outputHandle,
            cursorPosition

        INVOKE SetConsoleTextAttribute, outputHandle, 004h

        INVOKE WriteConsole, outputHandle, ADDR pobednikTekst,
            LENGTHOF pobednikTekst, ADDR bytesWritten, 0
        add pobednik,30h
        INVOKE WriteConsole, outputHandle, ADDR pobednik, 1,
            ADDR bytesWritten, 0

ret
printPobednik ENDP

printPreostalePartije PROC USES eax esi
        ;Postavljanje kursora na sredinu (otprilike)
        INVOKE GetConsoleScreenBufferInfo, outputHandle, ADDR
            consoleInfo
        mov esi, OFFSET ptrConsoleInfo
        mov ax, [esi]
        sub ax, 20
        mov ptrCursorPosition, ax
        mov ax, [esi+6]
        mov esi, OFFSET ptrCursorPosition
        mov [esi+2], ax
        INVOKE SetConsoleCursorPosition, outputHandle,
            cursorPosition

        INVOKE WriteConsole, outputHandle, ADDR
            preostalePartijeTekst, LENGTHOF preostalePartijeTekst
            , ADDR bytesWritten, 0

        mov al, brojPartije
        mov bl, 5
        sub bl,al
        add bl,30h
        mov preostalePartije, bl
```

```
        INVOKE WriteConsole, outputHandle, ADDR preostalePartije
            , 1,     ADDR bytesWritten, 0

        INVOKE WriteConsole, outputHandle, ADDR endline,
           LENGTHOF endline,       ADDR bytesWritten, 0

        INVOKE SetConsoleCursorPosition, outputHandle,
            kursorNaPocetak


ret
printPreostalePartije ENDP

END main
```