Elektrotehnički fakultet u Beogradu
Katedra za elektroniku

Ivan Beljić, 0299/2013
Dragana Milojević, 0522/2013

Računarska elektronika

Projektni izveštaj
Tic Tac Toe

mentori:
prof. dr Milan Prokin
asis. Aleksandra Lekić

Beograd, Jul 2017.

Sadržaj:

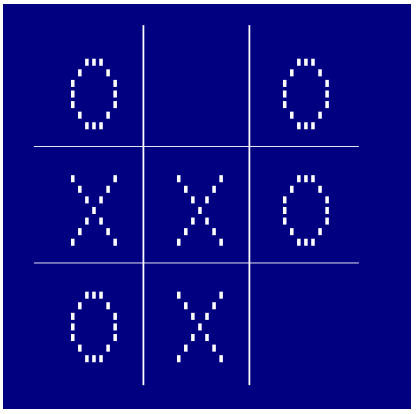
1. Uvod i tekst zadatka

Projektni zadatak podrazumeva programiranje procesora 8085 u asembleru prema zadatim specifikacijama. Kao projektno okruženje korišćen je *Microsoft Visual Studio 2017*, kao i pomoćna biblioteka *Irvine32*.

Konkretno u ovom projektu, bilo je potrebno napraviti igricu *Tic Tac Toe* za igranje u dve osobe. U igrici se 9 praznih polja popunjava tako što se pritiska jedan od 9 tastera rezervisanih za igranje igrice, čime se upisuje u dato polje X ili O. Odabrani su tasteri numeričke tastature 1-9.

Po pravilu prvom igraču odgovara O, a drugom X. Igrači igraju naizmenično. Igra se završava kada jedan od igrča sakupi 3 ista simbola (O ili X) u vrsti, koloni ili dijagonali. Tada se ispisuje da je dati igrač pobedio.

Igrica može da se prekine i pritiskom tastera *ESC*. Prilikom realizacije programa bilo je potrebno voditi računa o izgledu glavnog prozora igrice. Na slici 1 je prikazan primer izgleda prozora prilikom igranja igrice.



Slika 1: Izgled ekrana

2. Opis programa

Ceo program je koncipiran u 4 celine koje su smeštene u **main** proceduru.

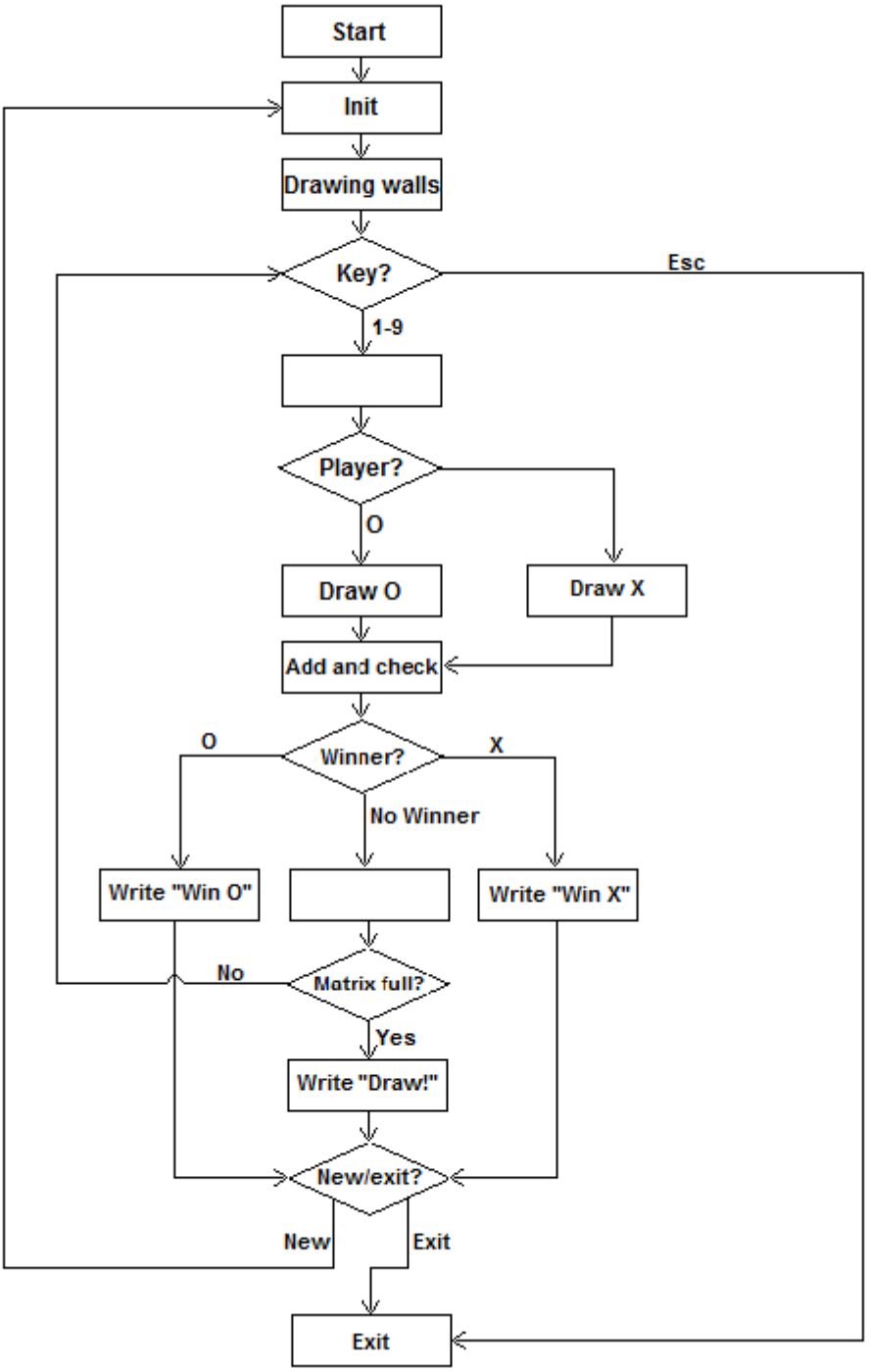
Prva celina predstavlja početni ekran u kome se iscrtava prepoznatljiva mreža. To se postiže tako što se na unapred proračunata mesta iscrtavaju karakteri iz ASCI tabele.

Tada se ulazi u drugu celinu programa, gde se očekuje pritisak na numeričkoj tastaturi ili pritisak tastera *ESC*. U zavisnosti od odabranog tastera, program se završava ili se na određeno mesto u mreži iscrtava O ili X, u zavisnosti od toga koji igrač je na redu za igranje. Pored iscrtavanja, menja se vrednost određenog elementa niza **matrix** koji govori da je na tom mestu upisan simbol, tako se program štiti od višestrukog upisa na istu lokaciju u mreži (ukoliko igrač pokuša da upiše na isto mesto simbol, program će zanemariti taj pokušaj i igrač će ispočetka moći da bira polje). Pored promene niza **matrix**, menjaju se i određeni elementi niza **sums**, u kome se nalazi 8 suma, 3 sume po vrstama, 3 sume po kolonama i 2 po dijagonalama. Vrednost O simbola je 1, a X simbola je 4.

Iscrtavanje je treći deo programa. Postoje dve labele, jedna za iscrtavanje simbola X, dok je druga za iscrtavanje simbola O. Prva labela iscrtava simbole po dijagonali, tako što inkrementira vrednost registara **dl** i **dh**, koji određuju poziciju kursora na konzolnoj liniji, i na to mesto iscrtava ASCII znak broj 0xFEh. Pored iscrtavanja na glavnoj dijagonali, iscrtava se i sporedna. Za nju je korisćena osobina da je zbir indeksa kolone i vrste na sporednoj dijagonali uvek konstantan. Druga labela iscrtava prvo dijagonalne elemente simbola O, za to iscrtavanje korišćena je osobina da se elementi susednih dijagonala nalaze u istoj koloni a da se njihovo rastojanje uvećava za 2 za gornje dve dijagonale, a smanjuje za 2 za donje dve dijagonale. Središnji elementi su iscrtani tako što se nalaze u istom redu tj. koloni, koja se nalazi na pola polja u kome se iscrtava zadati simbol.

Poslednji deo programa, proverava stanje niza **sums**, i ako je neki od elementa jednak 3, to znači da se u tom redu, koloni ili dijagonali nalaze 3 simbola O, a ako je on jednak 12, to znači da se u tom redu, koloni ili dijagonali nalaze 3 simbola X. Time se ta partija završava. Takođe, u ovom delu programa se proverava da li je došlo do pat pozicije, u kojoj su sva mesta u matrici popunjena, a ni u jednoj koloni, vrsti ili dijagonali ne postoje 3 ista simbola. Ukoliko je doslo do neka od gore tri pomenuta slučaja, na ekranu se ispisuje poruka o tome, igrači se pitaju da unesu enter za novu partiju, odnosno ESC za izlazak.

3. Algoritamski dijagrami toka



Slika 3: Dijagram toka main programa

4. Programski kod

```
; Solution program.
; PROGRAM TIC TAC TOE

INCLUDE Irvine32.inc
INCLUDE Macros.inc
; dl = current x
; dh = current y

; bl = next x
; bh = next y

drawDelay = 20   ;milliseconds between redrawing the ball

; Define the wall
wallTop = 23      ;top row number
wallBottom = 56         ;bottom row number
wallLeft = 25           ;left column number
wallRight = 70          ;right column number
wall_X1 = 40      ;x position (column number)
wall_X2 = 55      ;x position (column number)
wall_Y1 = 34      ;x position (column number)
wall_Y2 = 45      ;x position (column number)

; Define X
x1X = 5
x2X = 11
y1X = 3
y2X = 9
xWidth = 14

; Define the window size
xmin = 0          ;left edge
xmax = 95         ;right edge
ymin = 0          ;top
ymax = 79         ;bottom

; Define positions of top left corner of each feild
x11 = 25
y11 = 23
x12 = 40
y12 = 23
x13 = 55
y13 = 23

x21 = 25
y21 = 34
x22 = 40
y22 = 34
x23 = 55
y23 = 34

x31 = 25
y31 = 45
x32 = 40
y32 = 45
```

```
x33 = 55
y33 = 45

; Define O
x1O = 5
x2O = 11
y1O = 3
y2O = 9
xCentralO = 8
yCentralO = 6
; sum for winning
winX =  12
winO = 3
;value of x and o
xvalue = 4
ovalue = 1

BufSize = 80

.data

ddx BYTE 1        ;start position of current feild on Ox
ddy BYTE 1        ;start position of current feild on Oy
player BYTE 0     ; current player 0-O 1-X
matrix BYTE 9 DUP (0)
sums BYTE 9 DUP (0)
counter BYTE 0

buffer BYTE BufSize DUP(?)
stdInHandleo HANDLE ?
stdInHandlex HANDLE ?
bytesRead   DWORD ?

.code

main PROC
;---------- intro stuff, just for my demo
            call Clrscr


; PROGRAM STARTS HERE
;----------------------------------------------------------------
            mov eax,white + (blue * 16)
            call SetTextColor
            call Clrscr

;----- hides the cursor ----------------------------------------
.data
cursorInfo CONSOLE_CURSOR_INFO <>
outHandle  DWORD ?
.code
            INVOKE GetStdHandle, STD_OUTPUT_HANDLE
            mov  outHandle,eax
            INVOKE GetConsoleCursorInfo, outHandle, ADDR cursorInfo
            mov  cursorInfo.bVisible,0
            INVOKE SetConsoleCursorInfo, outHandle, ADDR cursorInfo
;-----------------insert names for players----------------------
startsHere:
            call Clrscr
            mov counter,0
```

```
        mov ecx,9
        mov player,0
insterts0:
        mov matrix[ecx],0
        mov sums[ecx],0
        loop insterts0


;---------------------------------------------------------------

;------ Draw the Wallx1 ---------------------------------------
; from (40,5) -- to (40,19)
        call Clrscr
        mov  dl,wall_X1
        mov  dh,wallTop
        mov  ecx,wallBottom - wallTop + 1
        mov  al,0B3h       ; solid block character

DrawWallx1:
        call Gotoxy
        call WriteChar
        inc  dh
        loop DrawWallx1
;---------------------------------------------------------------

;------ Draw the Wallx2----------------------------------------
; from (40,5) -- to (40,19)

        mov  dl,wall_X2
        mov  dh,wallTop
        mov  ecx,wallBottom - wallTop + 1
        mov  al,0B3h       ; solid block character

DrawWallx2:
        call Gotoxy
        call WriteChar
        inc  dh
        loop DrawWallx2
;---------------------------------------------------------------

;------ Draw the Wally1----------------------------------------
; from (40,5) -- to (40,19)

        mov  dl,wallLeft
        mov  dh,wall_Y1
        mov  ecx,wallRight - wallLeft + 1
        mov  al,0C4h       ; solid block character

DrawWally1:
        call Gotoxy
        call WriteChar
        inc  dl
        loop DrawWally1
;---------------------------------------------------------------

;------ Draw the Wally2----------------------------------------
; from (40,5) -- to (40,19)

        mov  dl,wallLeft
        mov  dh,wall_Y2
```

```
        mov  ecx,wallRight - wallLeft + 1
        mov  al,0C4h       ; solid block character

DrawWally2:
        call Gotoxy
        call WriteChar
        inc  dl
        loop DrawWally2
;-----------------------------------------------------------

;------ Draw corners  ---------------------------------------
        mov al, 0C5h
        mov dl, wall_X1
        mov dh, wall_Y1

        call Gotoxy
        call WriteChar

        mov dl, wall_X1
        mov dh, wall_Y2

        call Gotoxy
        call WriteChar

        mov dl, wall_X2
        mov dh, wall_Y1

        call Gotoxy
        call WriteChar

        mov dl, wall_X2
        mov dh, wall_Y2

        call Gotoxy
        call WriteChar

        jmp L0
;----------------------------------------

;------ Draw X  ---------------------------------------

DrawX:
        mov al, 0FEh
        mov dl, ddx
        add dl, x1X
        mov dh, ddy
        add dh, y1X
        mov ecx, y2X - y1X + 1

DrawXLoop:
        call Gotoxy
        call WriteChar
        mov bl, ddx
        add bl, ddy
        add bl, xWidth
        sub bl, dh
        mov bh, dl
        mov dl, bl

        call Gotoxy
```

```
        call WriteChar
        mov dl, bh
        inc dl
        inc dh
        loop DrawXLoop
        jmp checkWinner

;-----------------------------------------------------

;------ Draw O  ---------------------------------------


DrawO:
;drawing central pixels of O
        mov al, 0FEh
        mov dl, ddx
        add dl, xCentralO
        mov dh, ddy
        add dh, y1O
        call Gotoxy
        call WriteChar

        mov dh, ddy
        add dh, y2O
        call Gotoxy
        call WriteChar

        mov dh, ddy
        add dh, yCentralO
        mov dl , ddx
        add dl, x1O
        call Gotoxy
        call WriteChar

        mov dl , ddx
        add dl, x2O
        call Gotoxy
        call WriteChar


;drawing rest of pixels of O
        mov bl, 2
        mov al, 0FEh
        mov dl, ddx
        add dl, x1O
        add dl, bl
        mov dh, ddy
        add dh, y1O
        mov ecx, 3

DrawOLoop:
        call Gotoxy
        call WriteChar
        add dl, bl
        call Gotoxy
        call WriteChar
        add dh, 8
        sub dh, bl
        call Gotoxy
        call WriteChar
```

```
        sub dl, bl
        call Gotoxy
        call WriteChar
        add dh, bl
        sub dh, 8
        inc dh
        dec dl
        add bl, 2
        loop DrawOLoop
        jmp checkWinner

;-----------------------------------------------------
;check winner
checkWinner:    mov ecx,8
checking: mov bl, sums[ecx]
        cmp sums[ecx],winX
        jz xWin
        cmp sums[ecx],winO
        jz oWin
        loop checking
        mov bl, counter
        cmp bl,9
        jz refresh
        jmp L0

;-----------------------------------------------------
; wait for input

L0:     mov     eax,10    ; delay for msg processing
        call    Delay
        call    ReadKey         ; wait for a keypress
        jz      L0
Lesc: cmp dl,27
        jnz L1
        call Clrscr
        exit

;-----------------------------------------------------
; case 1-9
;-----------------------------------------------------
L1:     cmp     dl,97
        jnz     L2
        cmp matrix[1],0
        jnz L0
        mov ddx, x31
        mov ddy, y31
        cmp player, 0
        jnz L1x
L1o:mov matrix[1],ovalue
        mov al, sums[1]
        add al, ovalue
        mov sums[1],al
        mov al, sums[6]
        add al, ovalue
        mov sums[6],al
        mov al, sums[8]
        add al, ovalue
        mov sums[8],al
        mov player, 1
        mov bl , counter
```

```
        inc bl
        mov counter, bl
        jmp drawO
L1x:mov matrix[1],xvalue
        mov al, sums[1]
        add al, xvalue
        mov sums[1],al
        mov al, sums[6]
        add al, xvalue
        mov sums[6],al
        mov al, sums[8]
        add al, xvalue
        mov sums[8],al
        mov player , 0
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawX

L2: cmp dl,98
        jnz        L3
        cmp matrix[2],0
        jnz L0
        mov ddx, x32
        mov ddy, y32
        cmp player,0
        jnz L2x
L2o:mov matrix[2],ovalue
        mov al, sums[2]
        add al, ovalue
        mov sums[2],al
        mov al, sums[6]
        add al, ovalue
        mov sums[6],al
        mov player, 1
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawO
L2x:mov matrix[2],xvalue
        mov al, sums[2]
        add al, xvalue
        mov sums[2],al
        mov al, sums[6]
        add al, xvalue
        mov sums[6],al
        mov player, 0
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawX

L3: cmp dl,99
        jnz        L4
        cmp matrix[3],0
        jnz L0
        mov ddx, x33
        mov ddy, y33
        cmp player,0
        jnz L3x
```

```
L3o:mov matrix[3],ovalue
        mov al, sums[3]
        add al, ovalue
        mov sums[3],al
        mov al, sums[6]
        add al, ovalue
        mov sums[6],al
        mov al, sums[7]
        add al, ovalue
        mov sums[7],al
        mov player, 1
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawO
L3x:mov matrix[3],xvalue
        mov al, sums[3]
        add al, xvalue
        mov sums[3],al
        mov al, sums[6]
        add al, xvalue
        mov sums[6],al
        mov al, sums[7]
        add al, xvalue
        mov sums[7],al
        mov player, 0
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawX

L4: cmp dl,100
        jnz         L5
        cmp matrix[4],0
        jnz L0
        mov ddx, x21
        mov ddy, y21
        cmp player,0
        jnz L4x
L4o:mov matrix[4],ovalue
        mov al, sums[1]
        add al, ovalue
        mov sums[1],al
        mov al, sums[5]
        add al, ovalue
        mov sums[5],al
        mov player, 1
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawO
L4x:mov matrix[4],xvalue
        mov al, sums[1]
        add al, xvalue
        mov sums[1],al
        mov al, sums[5]
        add al, xvalue
        mov sums[5],al
        mov player, 0
        mov bl , counter
```

```
        inc bl
        mov counter, bl
        jmp drawX

L5: cmp dl,101
        jnz     L6
        cmp matrix[5],0
        jnz L0
        mov ddx, x22
        mov ddy, y22
        cmp player,0
        jnz L5x
L5o:mov matrix[5],ovalue
        mov al, sums[2]
        add al, ovalue
        mov sums[2],al
        mov al, sums[5]
        add al, ovalue
        mov sums[5],al
        mov al, sums[7]
        add al, ovalue
        mov sums[7],al
        mov al, sums[8]
        add al, ovalue
        mov sums[8],al
        mov player, 1
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawO
L5x:mov matrix[5],xvalue
        mov al, sums[2]
        add al, xvalue
        mov sums[2],al
        mov al, sums[5]
        add al, xvalue
        mov sums[5],al
        mov al, sums[7]
        add al, xvalue
        mov sums[7],al
        mov al, sums[8]
        add al, xvalue
        mov sums[8],al
        mov player, 0
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawX
L6: cmp dl,102
        jnz     L7
        cmp matrix[6],0
        jnz L0
        mov ddx, x23
        mov ddy, y23
        cmp player,0
        jnz L6x
L6o:mov matrix[6],ovalue
        mov al, sums[3]
        add al, ovalue
        mov sums[3],al
```

```
        mov al, sums[5]
        add al, ovalue
        mov sums[5],al
        mov player, 1
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawO
L6x:mov matrix[6],xvalue
        mov al, sums[3]
        add al, xvalue
        mov sums[3],al
        mov al, sums[5]
        add al, xvalue
        mov sums[5],al
        mov player, 0
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawX

L7: cmp dl,103
        jnz         L8
        cmp matrix[7],0
        jnz L0
        mov ddx, x11
        mov ddy, y11
        cmp player,0
        jnz L7x
L7o:mov matrix[7],ovalue
        mov al, sums[1]
        add al, ovalue
        mov sums[1],al
        mov al, sums[4]
        add al, ovalue
        mov sums[4],al
        mov al, sums[7]
        add al, ovalue
        mov sums[7],al
        mov player, 1
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawO
L7x:mov matrix[7],xvalue
        mov al, sums[1]
        add al, xvalue
        mov sums[1],al
        mov al, sums[4]
        add al, xvalue
        mov sums[4],al
        mov al, sums[7]
        add al, xvalue
        mov sums[7],al
        mov player, 0
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawX
```

```
L8: cmp dl,104
        jnz         L9
        cmp matrix[8],0
        jnz L0
        mov ddx, x12
        mov ddy, y12
        cmp player,0
        jnz L8x
L8o:mov matrix[8],ovalue
        mov al, sums[2]
        add al, ovalue
        mov sums[2],al
        mov al, sums[4]
        add al, ovalue
        mov sums[4],al
        mov player, 1
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawO
L8x:mov matrix[8],xvalue
        mov al, sums[2]
        add al, xvalue
        mov sums[2],al
        mov al, sums[4]
        add al, xvalue
        mov sums[4],al
        mov player, 0
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawX

L9: cmp dl,105
        jnz         L0
        cmp matrix[9],0
        jnz L0
        mov ddx, x13
        mov ddy, y13
        cmp player,0
        jnz L9x
L9o:mov matrix[9],ovalue
        mov al, sums[3]
        add al, ovalue
        mov sums[3],al
        mov al, sums[4]
        add al, ovalue
        mov sums[4],al
        mov al, sums[8]
        add al, ovalue
        mov sums[8],al
        mov player, 1
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawO
L9x:mov matrix[9],xvalue
        mov al, sums[3]
        add al, xvalue
        mov sums[3],al
```

```
        mov al, sums[4]
        add al, xvalue
        mov sums[4],al
        mov al, sums[8]
        add al, xvalue
        mov sums[8],al
        mov player,0
        mov bl , counter
        inc bl
        mov counter, bl
        jmp drawX
; X wins
xWin : call Clrscr
        mWrite <"X player is winner",0dh,0ah>
        mWrite <"Enter esc key for exit, and enter key if you want a new game.",0dh,0ah>
        jmp startOrEnd
; O wins
oWin : call Clrscr
        mWrite <"O player is winner!",0dh,0ah>
        mWrite <"Enter esc key for exit, and enter key if you want a new game.",0dh,0ah>
        jmp startOrEnd
; DRAW
refresh:
        call Clrscr
        mWrite <"It is draw!",0dh,0ah>
        mWrite <"Enter esc key for exit, and enter key if you want a new game.",0dh,0ah>
        jmp startOrEnd
; ESC or Continue
startOrEnd:
        mov     eax,10    ; delay for msg processing
        call    Delay
        call    ReadKey         ; wait for a keypress
        jz      startOrEnd
Lesc1: cmp dl,27
        jnz Lenter
        call Clrscr
        exit
Lenter:
        cmp dl,13
        jz startsHere
        jmp startOrEnd
main ENDP

END main
```