# UNIVERZITET U BEOGRADU
# ELEKTROTEHNIČKI FAKULTET

***Računarska elektronika***


## Projekat 14

## *Maze race*

Projekat radili:

| Ime | Prezime | broj indeksa |
|---|---|---|
| Raško | Kruščić | 468/2013 |
| Milena | Josipović | 236/2014 |

## Funkcija zadatka

Cilj igre je pobediti računar odnosno doći do izlaza iz lavirinta pre nego što on to učini. Izlaz iz lavirinta obeležen je putačom ×, računar je obeležen znakom plus +, dok je igrač kojim upravljate obeležen zvezdicom. Kroz lavirint se kreće upotrebom strelica na tastaturi. Poželjno je naći što kraći put do izlaza jer ukoliko računar prvi dođe do izlaza igra je gotova. Broj poena zavisi od vremena koje vam je bilo potrebno da pronađete izlaz. Što je traganje za izlazom duže trajalo, igrač će dobiti manje poena.

## Realizacija programa

Prvo je potrebno definisati sam izgled prozora (izgled lavirinta, putanju koja vodi do izlaza...) i osnovne promenljive kao što su početni položaj igrača, vreme koje je proteklo od početka igre, trenutni skor igrača.... Definišemo i stringove koji se ispisuju u slučaju pobede i poraza.
Pri izradi koda u glavnom programu koristili smo procedure kao i programski dostupne registre.
U kodu pored instrukcija nalaze se komentari koji objašnjavaju logiku koja stoji iza svakog dela koda zadatka.

# Kod programa

```
TITLE Maze Game

INCLUDE Irvine32.inc
;
;Wall = 219
;Exit = 088
;Player = 042
;AI = 043
;Blank = 000

.data

Maze DWORD    219,219,219,219,219,219,219,219,219,219,219,219,219,219,219,219,219,219,219
rowSize = ($ - Maze)
       DWORD  219,000,219,000,000,000,219,000,219,088,000,000,000,000,219,000,219  ; Exit
       DWORD  219,000,219,000,219,000,219,000,219,000,219,219,219,000,219,000,219
       DWORD  219,000,000,000,219,000,219,000,219,000,219,000,000,000,219,000,219
       DWORD  219,000,219,000,219,000,219,000,219,000,219,219,219,000,219,000,219
       DWORD  219,000,219,000,219,000,000,000,219,000,219,000,000,000,219,000,219
       DWORD  219,000,219,000,219,000,219,219,219,000,219,000,219,219,219,000,219
       DWORD  219,000,219,000,219,000,219,000,000,000,219,000,219,000,000,000,219
       DWORD  219,000,219,000,219,000,219,000,219,219,219,000,219,000,219,219,219
       DWORD  219,000,219,000,219,000,219,000,219,000,219,000,219,000,219,000,219
       DWORD  219,000,219,000,219,219,219,000,219,000,219,000,219,000,219,000,219
       DWORD  219,000,219,000,000,000,000,000,219,000,219,000,000,000,219,000,219
       DWORD  219,000,219,219,219,219,219,219,219,000,219,219,219,000,219,000,219
       DWORD  219,000,000,000,219,000,000,000,219,000,000,000,219,000,000,000,219
       DWORD  219,219,219,000,219,000,219,219,219,000,219,219,219,219,219,000,219
       DWORD  219,000,000,042,219,000,000,043,000,000,000,000,000,000,000,000,219  ; Player, AI
       DWORD  219,219,219,219,219,219,219,219,219,219,219,219,219,219,219,219,219

       MazeSolution  DWORD  4,4,4,4,4,4,4,4,-68,-68,-4,-4,-68,-68,-4,-4,-68,-68
                     DWORD  -68,-68,-68,-68,4,4,-68,-68,-68,-68,-4,-4,-4,-4, 0
                                                                  ; 0- AI has no more moves
                                                                  ; AI is winner
       mazeSolPos    DWORD  0
       playerPosition  DWORD       1032
       AIPosition DWORD 1048
       AITime DWORD 0              ; last time AI made a move
       AIMoveTime DWORD 0          ; time between moves (difficulty)
       counter DWORD 17
       counter1 DWORD 68           ; 17*4
       endOfMaze DWORD 289
       MOVE DWORD 68
       messageDirections BYTE "Use the arrow keys to move", 0dh, 0ah, 0
       messageTime BYTE "Time Past", 0dh, 0ah, 0
       messageScore BYTE "Your Score is ", 0dh, 0ah, 0
       msgWin BYTE "You are the winner!", 0dh, 0ah, 0
       msgLose BYTE "You lost the game", 0dh, 0ah, 0
       startTime DWORD 0
       time DWORD 0
       divisor DWORD 1000
       prizeScore DWORD 99         ; starting score is 99
       cursorInfo CONSOLE_CURSOR_INFO <>
       outHandle DWORD ?
```

```
.code
;*************************************************************************
;*****************************START**MAIN*********************************
;*************************************************************************

main PROC

	INVOKE GetStdHandle, STD_OUTPUT_HANDLE
	mov outHandle, eax
	INVOKE GetConsoleCursorInfo, outHandle, ADDR cursorInfo
	mov cursorInfo.bVisible, 0
	INVOKE SetConsoleCursorInfo, outHandle, ADDR cursorInfo       ; making cursor invisible

	mov  eax, white+(blue*16)          ; set blue screen
	call SetTextColor
	call Clrscr

	mov edx, OFFSET Maze
	add AIPosition, edx                ; address of AI in Maze array
	add PlayerPosition, edx            ; address of player in Maze array
	mov edx, OFFSET MazeSolution
	mov MazeSolPos, edx                ; address of current AIMove

	xor edx, edx                       ; EDX = 0
	INVOKE GetTickCount
	mov startTime, eax                 ; starting point
	mov AITime, eax
                                ; DRAW MAZE
	call Draw
	call Crlf
	call Crlf
	mov edx, OFFSET MessageDirections
	call WriteString                   ; print directions message
	mov edx, OFFSET MessageTime
	call WriteString
	call Crlf
	mov edx, OFFSET MessageScore
	call WriteString
GameLoop:
	cmp ebx, 99          ; end game value equals 99
	je EndGame
	call TimerScore
	mov  eax, 50         ; sleep, to allow OS to time slice
	call Delay                 ; (otherwise, some key presses are lost)
	call ReadKey         ; look for keyboard input
	jz   GameLoop        ; no key pressed yet

	cmp ah, 72           ; Up arrow key
	je Up
	cmp ah, 80           ; Down arrow key
	je Down
	cmp ah, 75           ; Left arrow key
	je Left
	cmp ah, 77           ; Right arrow key
	je Right
```

```
Up:
	mov eax, -68
	mov MOVE, eax
	call MoveProc
	jmp GameLoop
Down:
	mov eax, 68
	mov MOVE, eax
	call MoveProc
	jmp GameLoop
Left:
	mov eax, -4
	mov MOVE, eax
	call MoveProc
	jmp GameLoop
Right:
	mov eax, 4
	mov MOVE, eax
	call MoveProc
	jmp GameLoop
EndGame:
	call WaitMsg
	exit
main ENDP

;*****************************************************************************
;*****************************END**MAIN*****************************************
;*****************************************************************************


TimerScore PROC
	INVOKE GetTickCount
	sub eax, startTime
	div divisor                ; time[ms] / 1000, EAX = timeTaken
	sub eax, time
	cmp eax, 0
	je waiting                 ; wait for 1s to pass
	add eax, time              ; set new time
	mov time, eax

	mov edx, time
	sub edx, AITime
	cmp edx, AImoveTime        ; AIMoveTime=2 => move enemy every 2 secs
	jb skip

	mov edx, MazeSolPos
	mov eax, [edx]
	cmp eax, 0
	je gameOver
	call AImove

	skip:
	mov dl, 0
	mov dh, 20
	call gotoXY
	mov eax, time
	call WriteInt              ; print time
	mov ebx, prizeScore
	sub ebx, eax               ; calculating new score
```

```
        mov eax, ebx                ; WriteInt takes value from EAX
        mov dh, 22
        call gotoXY
        call WriteInt               ; print score
waiting:
        ret
gameOver:
        mov dl, 0
        mov dh, 23
        call gotoXY
        mov ebx, 99
        mov edx, OFFSET msgLose
        call writeString
        ret

TimerScore ENDP

;=============================================================================
;=============================================================================
AImove PROC
        xor eax, eax
        mov edx, AIposition         ; reading and writing to mem must be with EDX reg
        mov [edx], eax              ; delete old position

        mov edx, MazeSolPos
        mov eax, [edx]              ; read move from MazeSolution
        add AIposition, eax         ; add it to old position (this is position in Maze)

        mov eax, 43
        mov edx, AIposition
        mov [edx], eax              ; write new position
        mov eax, time
        mov AITime, eax             ; set new time
        call Draw
        add MazeSolPos, 4
        mov eax, MazeSolPos
        ret
AImove ENDP

;=============================================================================
;=============================================================================
MoveProc PROC
        mov ebx, MOVE
        add PlayerPosition, ebx             ; new position
        mov edx, PlayerPosition
        mov eax, [edx]                      ; read from maze
        cmp eax, 0                          ; Open Spot
        je ValidMove
        cmp eax, 219                        ; Wall
        je Wall
        cmp eax, 88                         ; Exit
        je ExitGame
ValidMove:
        mov ebx, MOVE                       ; EDX = MOVE
        sub PlayerPosition, ebx             ; return player to old position
        mov edx, PlayerPosition
        xor eax, eax                        ; 0 is space in ASCII, EAX = 0
        mov [edx], eax                      ; removes player from old position in maze
        add edx, ebx                        ; move player to new position
```

```
        mov PlayerPosition, edx
        mov eax, 42
        mov [edx], eax                  ; set player to new position in maze
        call Draw
        jmp MoveDone
Wall:
    mov eax, MOVE
        sub PlayerPosition, eax         ; invalid position, return to old one
        jmp MoveDone
ExitGame:
        mov dl, 0
        mov dh, 23
        call gotoXY
        mov edx, OFFSET msgWin
        call writeString
        mov ebx, 99
MoveDone:
        ret
MoveProc ENDP

;=============================================================================
;=============================================================================
Draw PROC

        mov dh, 0                   ; Set maze position X
        mov dl, 0                   ; Set maze position Y
        call Gotoxy                 ; Call Go to X Y
        mov ebx, OFFSET Maze        ; Move the maze 2D array into ebx
        xor ecx, ecx                ; reset ECX counter
PrintLoop:
        mov eax, [ebx]              ; read character
        add ebx, 4                  ; Move to the next character in maze
        inc ecx                     ; Increment the counter
        call WriteChar              ; Write Character

        cmp ecx, endOfMaze          ; check if maze is drawn
        je return
        xor edx, edx                ; EDX = 0
        mov eax, ecx                ; EAX / 20 = EAX, ostatak u EDX
        div counter                 ; EAX / 20 = EAX, remainder in EDX
        cmp edx, 0                  ; compare for end of row
        je NextLine
        jmp PrintLoop

NextLine:
        call Crlf
        jmp PrintLoop
return:
        ret
Draw ENDP

;=============================================================================
;=============================================================================
END main
```