Project Report

Computer Organization & Assembly Language

Project name:

A ball moving around the screen

**GROUP MEMBERS**

MUHAMMAD AHSAN YOUNUS (CSC-18F-130)

AHTESHAM-UL-HAQ (CSC-18F-126)

**SUMITTED TO: MISS SAHAR ZAFAR JUMANI**

**Introduction:**

**The assembly language is a low-level programming language,** which offers an efficient way for writing programs. Under the efficient, we assume that the programmer has a complete control over the code organization. This is possible because there is no compiler which translates and organizes the code. Thus, writing the program and the code organization is a full responsibility of the programmer. The programmer has access to each memory address and the full control of each byte in memory. This feature made assembly a very popular in writing fast interrupt procedures in the world of embedded systems over the years.

**The project is about a ball moving around the screen and touching the edges .**

**Source Code:**

STACK SEGMENT PARA STACK

DB 64 DUP (' ')

STACK ENDS

DATA SEGMENT PARA 'DATA'

WINDOW\_WIDTH DW 140h ;the width of the window (320 pixels)

WINDOW\_HEIGHT DW 0C8h ;the height of the window (200 pixels)

WINDOW\_BOUNDS DW 6 ;variable used to check collisions early

TIME\_AUX DB 0 ;variable used when checking if the time has changed

BALL\_X DW 0Ah ;X position (column) of the ball

BALL\_Y DW 0Ah ;Y position (line) of the ball

BALL\_SIZE DW 04h ;size of the ball (how many pixels does the ball have in width and height)

BALL\_VELOCITY\_X DW 05h ;X (horizontal) velocity of the ball

BALL\_VELOCITY\_Y DW 02h ;Y (vertical) velocity of the ball

DATA ENDS

CODE SEGMENT PARA 'CODE'

MAIN PROC FAR

ASSUME CS:CODE,DS:DATA,SS:STACK ;assume as code,data and stack segments the respective registers

PUSH DS ;push to the stack the DS segment

SUB AX,AX ;clean the AX register

PUSH AX ;push AX to the stack

MOV AX,DATA ;save on the AX register the contents of the DATA segment

MOV DS,AX ;save on the DS segment the contents of AX

POP AX ;release the top item from the stack to the AX register

POP AX ;release the top item from the stack to the AX register

CALL CLEAR\_SCREEN

CHECK\_TIME:

MOV AH,2Ch ;get the system time

INT 21h ;CH = hour CL = minute DH = second DL = 1/100 seconds

CMP DL,TIME\_AUX ;is the current time equal to the previous one(TIME\_AUX)?

JE CHECK\_TIME ;if it is the same, check again

;if it's different, then draw, move, etc.

MOV TIME\_AUX,DL ;update time

CALL CLEAR\_SCREEN

CALL MOVE\_BALL

CALL DRAW\_BALL

JMP CHECK\_TIME ;after everything checks time again

RET

MAIN ENDP

MOVE\_BALL PROC NEAR

MOV AX,BALL\_VELOCITY\_X

ADD BALL\_X,AX ;move the ball horizontally

MOV AX,WINDOW\_BOUNDS

CMP BALL\_X,AX

JL NEG\_VELOCITY\_X ;BALL\_X < 0 + WINDOW\_BOUNDS (Y -> collided)

MOV AX,WINDOW\_WIDTH

SUB AX,BALL\_SIZE

SUB AX,WINDOW\_BOUNDS

CMP BALL\_X,AX ;BALL\_X > WINDOW\_WIDTH - BALL\_SIZE - WINDOW\_BOUNDS (Y -> collided)

JG NEG\_VELOCITY\_X

MOV AX,BALL\_VELOCITY\_Y

ADD BALL\_Y,AX ;move the ball vertically

MOV AX,WINDOW\_BOUNDS

CMP BALL\_Y,AX ;BALL\_Y < 0 + WINDOW\_BOUNDS (Y -> collided)

JL NEG\_VELOCITY\_Y

MOV AX,WINDOW\_HEIGHT

SUB AX,BALL\_SIZE

SUB AX,WINDOW\_BOUNDS

CMP BALL\_Y,AX

JG NEG\_VELOCITY\_Y ;BALL\_Y > WINDOW\_HEIGHT - BALL\_SIZE - WINDOW\_BOUNDS (Y -> collided)

RET

NEG\_VELOCITY\_X:

NEG BALL\_VELOCITY\_X ;BALL\_VELOCITY\_X = - BALL\_VELOCITY\_X

RET

NEG\_VELOCITY\_Y:

NEG BALL\_VELOCITY\_Y ;BALL\_VELOCITY\_Y = - BALL\_VELOCITY\_Y

RET

MOVE\_BALL ENDP

DRAW\_BALL PROC NEAR

MOV CX,BALL\_X ;set the initial column (X)

MOV DX,BALL\_Y ;set the initial line (Y)

DRAW\_BALL\_HORIZONTAL:

MOV AH,0Ch ;set the configuration to writing a pixel

MOV AL,0Fh ;choose white as color

MOV BH,00h ;set the page number

INT 10h ;execute the configuration

INC CX ;CX = CX + 1

MOV AX,CX ;CX - BALL\_X > BALL\_SIZE (Y -> We go to the next line,N -> We continue to the next column

SUB AX,BALL\_X

CMP AX,BALL\_SIZE

JNG DRAW\_BALL\_HORIZONTAL

MOV CX,BALL\_X ;the CX register goes back to the initial column

INC DX ;we advance one line

MOV AX,DX ;DX - BALL\_Y > BALL\_SIZE (Y -> we exit this procedure,N -> we continue to the next line

SUB AX,BALL\_Y

CMP AX,BALL\_SIZE

JNG DRAW\_BALL\_HORIZONTAL

RET

DRAW\_BALL ENDP

CLEAR\_SCREEN PROC NEAR

MOV AH,00h ;set the configuration to video mode

MOV AL,13h ;choose the video mode

INT 10h ;execute the configuration

MOV AH,0Bh ;set the configuration

MOV BH,00h ;to the background color

MOV BL,00h ;choose black as background color

INT 10h ;execute the configuration

RET

CLEAR\_SCREEN ENDP

CODE ENDS

END