

Computer Architecture Lab – CS322

Name: M. Maheeth Reddy

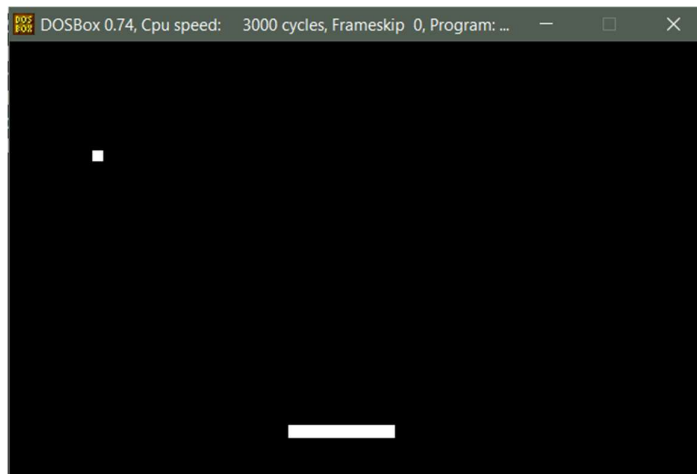
Roll No.: 1801CS31

Date: 23 September 2020

Lab 4 – Assembly Language Programming

A Simple Bat and Ball game in 8086 assembly language

I have designed a simple game in the lines of the famous DX-Ball game, but with no bricks. The game consists of a bat at the bottom of the screen and a ball. A player can move the bat towards left or right using the designated keys. The aim is to prevent the ball from hitting the bottom of the screen. This game is intended to be a stressbuster only, so there are no scores. If the player fails to prevent the ball hitting the bottom of the screen, the game restarts.



Gameplay

Controls:

Move Bat Left



Move Bat Right



Design Overview:

The game comprises of two moving objects: a **Ball** and a **Bat**. So, the program is configured to video mode. The motion of these two objects depends on:

1. their **Velocity** (constant during runtime)
 - a. **Velocity of ball** is 4 pixels per 0.01 second in both x and y directions.
 - b. **Velocity of bat** is 8 pixels for one keypress.
2. Keyboard Input
3. Boundaries of the screen

So, the program must keep checking for keyboard input at every instant of time and make sure that both the objects are within the bounds of the screen.

For this purpose, the program keeps track of the system time. It calculates the positions of the ball and the bat at most once in 0.01 seconds (This is the least time we can get using the interrupt). It clears the screen and draws the objects at their new positions. Clearing the screen removes any traces formed by the moving objects.



Updating the position of the ball:

The new position of the ball is calculated based on its current position and velocity. To check whether the ball is within the window or colliding with the edges, the following conditions are checked and accordingly velocity of the ball is changed.

1. Does the ball collide with left edge of the screen ?
 - a. If yes, negate the x-direction velocity of the ball
 - b. Otherwise, check next condition
2. Does the ball collide with right edge of the screen ?
 - a. If yes, negate the x-direction velocity of the ball
 - b. Otherwise, check next direction
3. Does the ball collide with top edge of the screen ?
 - a. If yes, negate the y-direction velocity of the ball
 - b. Otherwise, check next condition
4. Does the ball collide with bottom edge of the screen ?
 - a. If yes, restart the game
 - b. Otherwise, check next condition
5. Does the ball collide with the bat ?
 - a. If yes, negate the y-direction velocity of the ball
 - b. Otherwise, return

Updating the position of the bat:

The new position of the bat is calculated based on its current position and velocity. The bat is restricted to the window area by checking the following conditions.

1. Is there any keyboard input ?
 - a. If yes, check next condition
 - b. Otherwise, don't change position of bat
2. Which key ?
 - a. If it is **Comma**  , move the bat left and make sure it is within the window.
 - b. If it is **Full Stop**  ., move the bat right and make sure it is within the window.

Drawing ball/bat on the screen:

A ball is a square with the dimensions 4 x 4 pixels. Therefore, there are 4 rows of pixels with each row having 4 pixels. To draw the ball on the screen:

1. A pixel is placed on each column of the first row.
2. Move to the next row and repeat until all rows are drawn.

Same procedure is followed for drawing the bat (dimensions: 48 x 5 pixels).

The repeated execution of the procedures explained above gives a feel of playing the game.

Exiting the Game: Player can exit the game by closing DOSBox.

Interrupts Used

Interrupt	Parameters	Purpose
INT 21H / AH = 2CH	nil	<u>Get System Time</u> CH = hour CL = minute DH = second DL = 1/100 seconds
INT 16H / AH = 01H	nil	Check for any keystroke in Keyboard buffer ZF = 1 if keystroke is not available ZF = 0 Otherwise
INT 16H / AH = 00H	nil	Read Keyboard Input (no echo) to AL register
INT 10H / AH = 0CH	AL = 0FH (white colour) BH = 00H (page number) CX for column DX for row	Change colour of pixel at location specified by CX, DX
INT 10H / AH = 00H	AL = 13H	Set graphical video mode 320 x 200 pixels, 1 page
INT 10H / AH = 0BH	BH = 00H BL = 00H (black colour)	Set black background

Note: A 30-second gameplay video ([*gameplay.mkv*](#)) has been included. Please check that too.