COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE LAB

<u>PROJECT REPORT</u>

"PONG GAME"

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCS KARACHI CAMPUS

Group Members:

Muhammad Ali 23K-0730

Ayan Hussain 23K-0608

Umer Khan 23K-0798

Submitted to: Engr Nadeem Ghouri

Introduction

This project uses the Irvine32 library to construct a classic "Pong" game in assembly language. The user controls a paddle on the right side of the screen while playing the game inside a rectangular box. The ball keeps bouncing off the paddle and walls. Preventing the ball from hitting the right wall after passing the paddle is the aim of the game. The game is over, and the score is shown if the player misses the ball.

Objective

- Use assembly language to create a Pong game.
- For handling text-based graphics and input/output, use the Irvine32.inc library.
- Use game logic to move the ball, detect collisions, track scores, and move the paddles based on user input

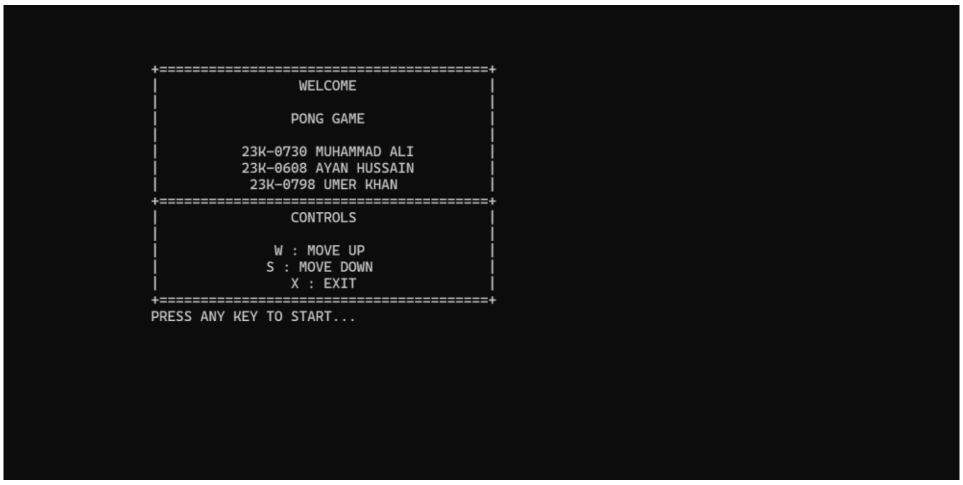
Technologies and Tools Employed

- Assembly Language (MASM x86) is the programming language.
- IDE: Visual Studio 2022 with the MASM plugin, or any IDE and assembler that is compatible with x86.
- Libraries: Irvine32 library (used for input/output tasks such as text writing and character reading).
- System specifications: a system that can run software based on 32-bit DOS. The Irvine32 library and the MASM assembler a console or text-based terminal to play the game on.

DESIGN AND FUNCTIONALITY

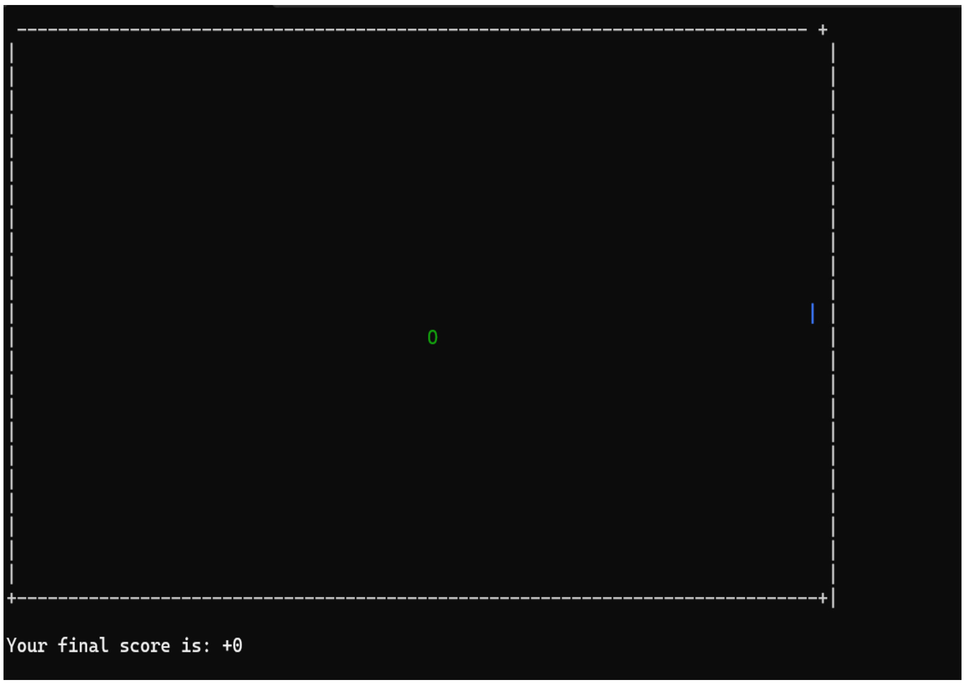
The program is divided into several procedures that handle different aspects of the game:

1. **Initialization:**
    - Displays a welcome message and prompts the user to press any key to start the game.
    - Clears the screen and draws the initial game box using the DrawBox procedure.
    - Draws the initial paddle and ball positions.



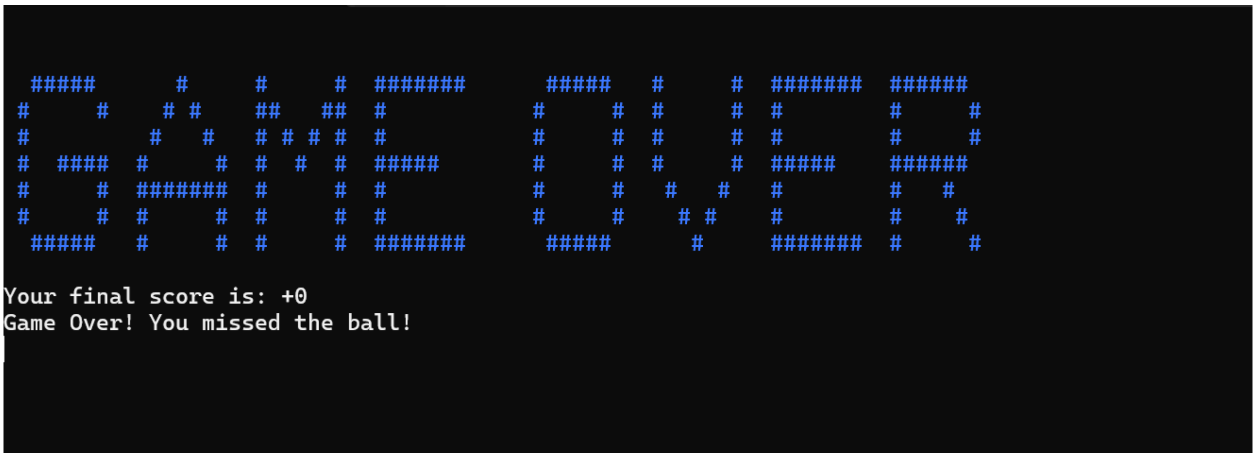
```
+=========================================+
|                 WELCOME                 |
|                                         |
|                PONG GAME                |
|                                         |
|          23K-0730 MUHAMMAD ALI          |
|          23K-0608 AYAN HUSSAIN          |
|           23K-0798 UMER KHAN            |
+=========================================+
|                CONTROLS                 |
|                                         |
|              W : MOVE UP                |
|             S : MOVE DOWN               |
|                X : EXIT                 |
+=========================================+
PRESS ANY KEY TO START...
```

2. **Game Loop:**
   - The game runs in a loop, with the ball's position being updated based on its direction (ballDX and ballDY).
   - The paddle can be moved up or down based on user input (W to move up, S to move down).
   - The ball bounces off the top and bottom walls and the paddle. If the ball reaches the right wall without hitting the paddle, the game ends.
   - The score is updated each time the ball hits the paddle, and it is displayed at the bottom of the screen.
3. **Collision Detection:**
   - The CheckCollision procedure checks if the ball collides with the walls or the paddle. If the ball touches the top or bottom walls, it changes direction.
   - If the ball collides with the paddle, it reverses direction and increases the score.
   - If the ball crosses the left side of the screen (without hitting the paddle), the game ends.
4. **Drawing the Game Elements:**
   - **DrawBox:** Draws the rectangular box that forms the boundary of the game.
   - **DrawPaddle:** Draws the player's paddle on the right side of the screen and updates its position based on user input.
   - **DrawBall:** Draws the ball and updates its position after each move.



```
Your final score is: +0
```

5. **Game Over:**
   - If the ball crosses the left boundary and the player misses it, the game ends and displays the final score.



```
 #####      #     #     #  #######      #####   #     #  #######  ######
#     #    # #    ##   ##  #           #     #  #     #  #        #     #
#         #   #   # # # #  #           #     #  #     #  #        #     #
#  ####  #     #  #  #  #  #####       #     #  #     #  #####    ######
#     #  #######  #     #  #           #     #   #   #   #        #   #
#     #  #     #  #     #  #           #     #    # #    #        #    #
 #####   #     #  #     #  #######      #####      #     #######  #     #

Your final score is: +0
Game Over! You missed the ball!
```

CODE BREAKDOWN

- **Data Section:**
  - The .data section contains all the strings used for displaying messages like the welcome message, start message, game over message, and score message.
  - Variables for game elements such as the paddle and ball positions, direction (velocity), and the score are also defined here.

    - The game's boundary box (top, bottom, and sides) is defined using strings of characters.
- **Main Procedure:**
    - The main procedure coordinates the flow of the game, starting with displaying the welcome message, reading user input, and then entering the game loop.
- **Game Loop:**
    - The gameLoop procedure is a continuous loop that updates the ball position, checks for collisions, draws the paddle and ball, and updates the score.
- **Drawing Functions:**
    - The DrawBox procedure is responsible for rendering the boundary of the game screen.
    - The DrawPaddle and DrawBall procedures handle the drawing of the paddle and ball on the screen and clearing the previous positions.
- **Collision and Input:**
    - The CheckCollision procedure checks if the ball has hit the top, bottom, or paddle. If the ball reaches the right side without hitting the paddle, the game ends.
    - The CheckInput procedure checks if the user has pressed 'W' to move the paddle up or 'S' to move it down.

KEY FEATURES

- **Paddle Movement:** The player can move the paddle up or down using the 'W' and 'S' keys, respectively. The paddle cannot move beyond the top or bottom boundaries of the screen.
- **Ball Movement:** The ball moves automatically, bouncing off the top and bottom walls. The ball's direction is reversed when it collides with the paddle or the top/bottom walls.
- **Score System:** Each time the ball hits the paddle, the score increases. If the player misses the ball, the game ends and the final score is displayed.
- **Game Over Condition:** The game ends if the ball misses the paddle and reaches the left wall. The final score is displayed at this point.

FLOW CHART

1. **Start:**
    - Display welcome message and wait for user input.
2. **Game Setup:**
    - Clear the screen, draw the game box, and initialize the ball and paddle positions.
3. **Game Loop:**
    - Move ball.
    - Check for collisions (top wall, bottom wall, paddle).
    - Update score and draw elements.
    - Handle user input for paddle movement.
4. **Game Over:**
    - Display game over message and final score.

CONCLUSION

This Pong game in assembly demonstrates the use of basic game programming techniques such as collision detection, movement, and simple graphics within a text-based interface. The project provides an excellent introduction to assembly language programming, game logic, and using the Irvine32 library for input/output operations in an assembly environment.

While the game is relatively simple, it offers a solid foundation for creating more advanced games and understanding how low-level hardware interactions work in a DOS environment.

Future Improvements:

- **Game Difficulty:** Add difficulty levels by increasing the speed of the ball as the player progresses.
- **Enhanced Graphics:** Use more colors or symbols to improve the visual appearance of the game.
- **Sound Effects:** Implement sound effects for ball bouncing, scoring, and game over events.

REFERENCES

- Irvine, K. (2015). **Irvine32 Library**. Retrieved from [Irvine32 Library Documentation].
- Microsoft (n.d.). **MASM x86 Assembler**. Retrieved from [MASM x86 Documentation].