ProJect Report

Arkanoid

Mehrdad Varmaziar
Amir Mahdi Blohari

—

Microprocessor & Assembly Language

—

Professor Vakilian

Febuary 20, 2026

INTRODUCTION

The purpose of this project is to design and implement a simple 2D game based on the classic Arkanoid using the Intel 8086 microprocessor and Assembly language.

The project demonstrates how low-level programming can be used to control graphics, keyboard input, memory, and real-time game logic.

This implementation runs in VGA graphics mode and includes paddle movement, ball animation, collision detection, and win/lose conditions.

The game is fully controlled by the keyboard and is executed in a continuous loop, similar to real game engines.

THE PROCESS

Project Objective

The main objectives of this project are:

- To use Intel 8086 Assembly language for real-time programming
- To switch and control VGA graphics mode
- To draw game objects (ball, paddle, blocks)
- To detect keyboard input (A / D keys)
- To implement collision detection algorithms
- To manage game states (playing, win, game over)

Game Description

The game is a simplified version of Arkanoid:

- The player controls a paddle at the bottom of the screen
- The ball moves continuously and bounces off walls, the paddle, and blocks
- The goal is to destroy all blocks
- If the ball leaves the bottom off the screen, the player loss

System Setup

Graphic Mode :

The program uses BIOS interrupt 10h to set VGA graphics:

```
mov ax, 0013h
int 10h
```

This enables 320×200 resolution with 256 colors.

THE PROCESS

Program Sctructure

The Program is divided into logical subroutines :

1. Initialization - sets graphics mode and initializes variables
2. Input Handling - reads keyboard (A & D keys)
3. Ball Movement – updates ball position
4. Collision Detection – checks ball vs walls , paddle , block
5. Rendering – redraws all objects
6. Game Loop – repeats all steps cotinuously

Data and Variables

Variable	Description
ball_x	Ball horizontal position
ball_y	Ball vertical position
ball_dx	Ball horizontal velocity
ball_dy	Ball vertical velocity
paddle_x	Paddle horizontal position
blocks[]	Block status array(1 = active , 0 = destroyed)

THE PROCESS

Main Game Loop

This loop ensures smooth animation and real-time control :

```
main_loop:
        call read_input
        call move_ball
        call check_collision
        call draw_scene
        jmp main_loop
```

Collision Detection

- Wall collision:
  reverses horizontal or vertical direction
- Paddle collision:
  reverses vertical direction
- Block collision:
  removes the block and changes ball direction

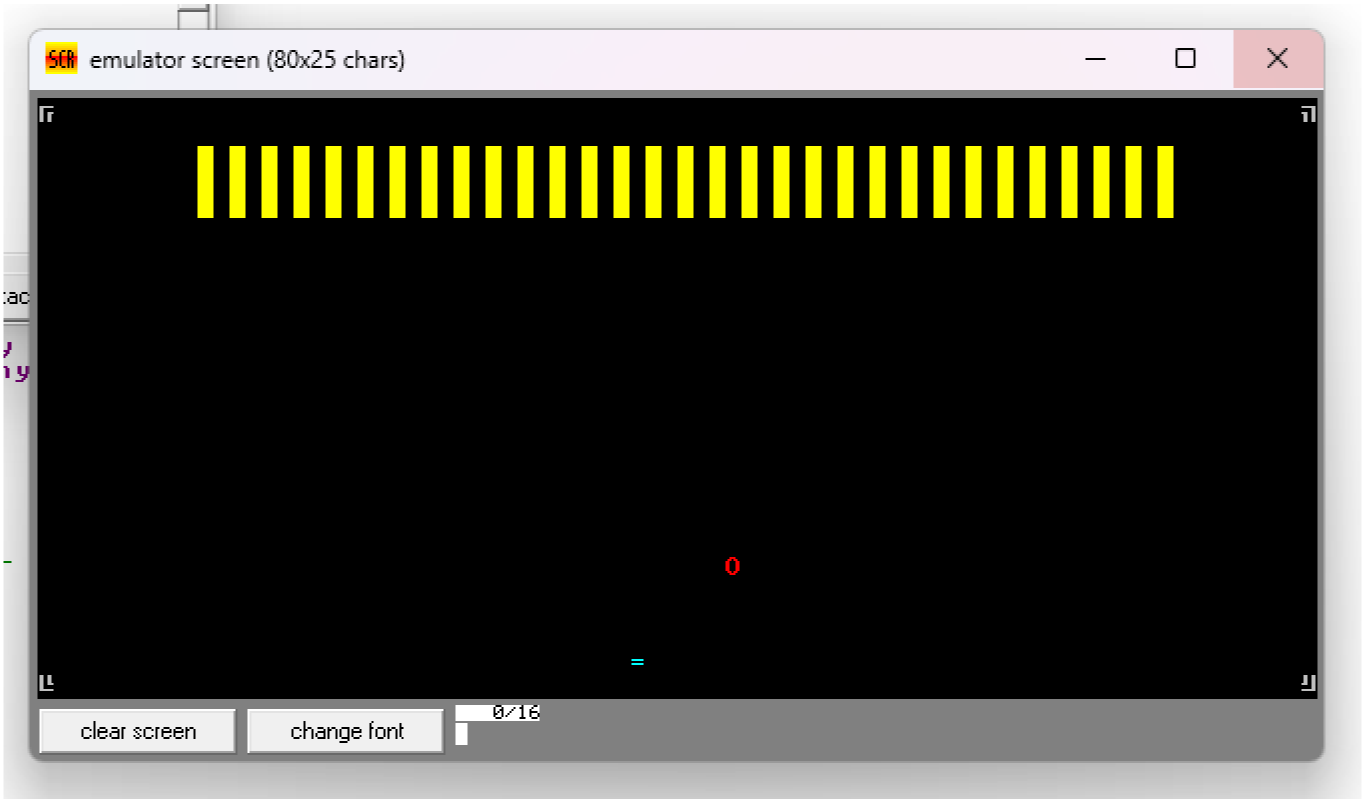
All collisions are detected by comparing the ball coordinates with the object boundaries.

Win & Game Over Condition

- Win **:**
  when all block values become zero
- Game Over :
  when the ball passes the bottom edges

Messages are displayed accordingly.

Export image



Conclusion

This project proves that even a low-level microprocessor such as the Intel 8086 can be used to build an interactive graphical game.

It demonstrates important concepts such as:

- Direct hardware control
- Low-level graphics programming
- Real-time input processing
- Game loop architecture
- Collision detection logic

This project helped strengthen understanding of both microprocessor architecture and assembly programming techniques.

Finish.