

Submitted By:

**Bilal Naveed (500)**

Subject:

**Computer Organization &**

**Assembly Language**

Section:

**B**

**DEPARTMENT OF COMPUTER SCIENCE**

**UNIVERSITY OF ENGINEERING AND TECHNOLOGY, LAHORE NAROWAL CAMPUS**

**Project Report: Coin Collector Game in Assembly Language**

## ****1. Introduction****

The **Coin Collector Game** is a simple yet engaging game developed in **x86 Assembly Language** using DOS interrupts. The game allows players to control characters on the screen to collect randomly placed coins while avoiding collisions with walls and each other. The project demonstrates fundamental concepts of **low-level programming**, including **input handling, screen manipulation, and random number generation**.

## ****2. Project Overview****

### **2.1 Game Description**

* **Single-Player Mode**: One player controls a character ("A") to collect coins.
* **Two-Player Mode**: Two players ("A" and "B") compete to collect coins while avoiding collisions.
* **Scoring System**: Each collected coin increments the player's score.
* **Game Over Conditions**:
  + Colliding with the game border.
* **Winner Determination**: The player with the highest score wins.

### **2.2 Key Features**

* **Dynamic Input Handling**: Supports keyboard inputs for movement.
* **Random Coin Generation**: Coins appear at random positions.
* **Score Tracking**: Real-time score updates.
* **Game Over Screen**: Displays the winner and final scores.

## ****3. Technical Implementation****

### **3.1 Data Section**

The game uses predefined strings and variables:

* **Player Names** (P1, P2): Stored as strings.
* **Player Positions** (XPOS, YPOS, XPOS2, YPOS2): Track coordinates.
* **Coin Position** (XCOINPOS, YCOINPOS): Randomly generated.
* **Score Variables** (SCORE, SCORE2): Track points.

### **3.2 Main Procedures**

#### **3.2.1 Game Initialization**

* Clears the screen (CLEAR\_SCREEN).
* Displays the game title (C\_DISPLAY).
* Asks for the number of players (1 or 2).

#### **3.2.2 Player Name Input**

* Uses DOS interrupt 0Ah to read player names.
* Stores names in buffers (P1\_BUFFER, P2\_BUFFER).

#### **3.2.3 Game Loop**

1. **Draws the Game Border** (Draw\_Border):
   * Uses asterisks (\*) to create boundaries.
2. **Handles Player Movement** (GetInput\_single/GetInput\_multi):
   * **Player A Controls**: W (Up), Z (Down), A (Left), D (Right).
   * **Player B Controls**: I (Up), M (Down), J (Left), L (Right).
3. **Coin Collection Logic** (CheckCoinCollection\_single/CheckCoinCollection\_multi):
   * Checks if a player’s position matches the coin’s position.
   * Increments score and generates a new coin (CreateRandomCoin).
4. **Collision Detection** (CheckCollisions):
   * Detects wall collisions and player collisions (in two-player mode).

#### **3.2.4 Game Over & Winner Determination**

* Displays scores and declares the winner (GameOverScreen\_multi).
* Asks if players want to replay (PLAY\_AGAIN\_MSG).

## ****4. Key Challenges & Solutions****

### **4.1 Random Number Generation**

* **Challenge**: Generating random positions for coins.
* **Solution**: Used system time (int 1Ah) to seed pseudo-random positions.

### **4.2 Input Handling**

* **Challenge**: Detecting key presses without blocking the game loop.
* **Solution**: Used int 16h with AH=01h to check for keypresses.

### **4.3 Screen Manipulation**

* **Challenge**: Updating player positions without flickering.
* **Solution**: Used cursor positioning (int 10h) to overwrite characters.

## ****5. Future Enhancements****

1. **Difficulty Levels**: Introduce speed variations or obstacles.
2. **Sound Effects**: Add beeps for coin collection and collisions.
3. **High Score System**: Save scores to a file.
4. **Graphical Improvements**: Use extended ASCII characters for better visuals.

## ****6. Conclusion****

This project successfully demonstrates **low-level game development in Assembly**, highlighting:

* **Efficient memory usage** (registers, variables).
* **Direct hardware interaction** (keyboard, screen).
* **Structured programming** (procedures, loops).

The **Coin Collector Game** serves as an excellent learning tool for understanding **x86 Assembly** and **DOS programming**. Future work could expand the game with additional features while maintaining its simplicity.