Logo, company name

Description automatically generated

**COMSATS University Islamabad (CUI)**

**Project Proposal**

**For**

**Snake Game**

**Kulsoom Khurshid CIIT/SP20-BCS-044/ISB**

***Course***

**Microprocessor and Assembly Language**

***Professor*Taimur Shahzad**

***Bachelor of Science in Computer Science (2020-2024)***

**Table of Contents**

1. Introduction 3

2. General Overview 3

3. Background 4

4. Description 4

5. Related Work 6

6. Flow Chart 6

7. Advantage 9

8. Code 9

9. Conclusion 19

# Introduction

This project is about a game that we all have been playing in our childhood, ‘the snake game’. It is not the best game to develop, but you can have an idea of how to code a simple game in assembly language using the Irvine library. This game has no special name but people call it snake game as the player maneuvers a line that grows in length if the player eats the fruit which is represented by a dot. If that line touches the boundary walls or it’s body the snake dies indicating the end of the game. The concept for this game was initiated by Arcade Game Blockade in 1976. There are many developers in the world who have developed this game leading to many versions of this game on different platforms. In 1988, it’s variant was released on Nokia Mobiles that attracted a large number of audience towards it.

The aim of this project is to have simplicity and fun. Before starting the game it asks the user for the speed and once the game is started the user moves the line i.e. snake left and right, up and down.

To successfully complete the project, the user need to have a strong understanding of the device on which it is being deployed, the platform on which it is created and to have the strong concepts of the assembly language. An algorithm is to be designed for handling various task such as asking the user for the speed, handling the line or snake according to that speed, for calculating the total score, time, etc.

# General Overview

The snake game programmed in assembly language is a program that eats a fruit that leads to the growth of snake body and increases players score. The program is controlled by the microprocessor that executes the program written in the assembly language to carry out the various task such as snake movement, eating the fruit and eventually increasing the players score. The hardware components that are used are the keyboard keys such as W, A, S, D, display screen and the game mechanism that is controlled by microprocessor and assembly language program that allows the user to interact with the hardware and the program.

The assembly language program includes the instruction for the handling various task such as moving the snake, score count, time count, etc. The program also handle the situations when the snake hits the wall or touches it’s body itself.

The program is designed in a way that every time the snake touches the dot the score gets increased by 1.

# Background

Assembly language is a low-level programming language that is used to write programs that are executed directly by the processor. It is considered a "low-level" language because it is closer to the machine language that the processor can execute directly and is more difficult to read and understand than higher level languages like C or Python. Assembly language is often used in systems where speed and memory efficiency are important, such as in embedded systems or real-time systems. Snake game is an example of such program that might benefit from using assembly language because they often have limited resources and require fast response times to

Handle flow of the program.

The snake game is one of the game that was introduced in mid 90’s and played by almost all of us. The game was first introduced in Nokia phones which helped them to gain many more customers. The game looks quite simple but it is not that simple to develop many efficient algorithms were used for it’s working. Hence the use of assembly language to program this game increases its efficiency and reliability which is important for the seamless user experience.

# Description

The snake game by its name it is obvious that its about a snake that moves in the boundary and looks for its food that is displayed in the form of a dot. As the snake hits this dot, its length increases and the score get incremented by 1. More hit means the difficulty level increase as the snake body itself is the biggest obstacle in the game.

The snake movement is controlled by the keyboard key. W is used to move the snake forward, S for moving the snake downwards, A to turn left and D to turn right. These are constants and you can program according to will. For instance, you can use the keyboard arrow keys for the movement as well but that totally depends on the structure of the program.

The aim of the game is to collect as many dots as possible without hitting the boundary walls or the snake body itself as the leads to the end of the game. As the snake collects the dots the body increases along with the score more chances of crashing to its body itself. After a certain time you have collected the enough food the game moves to next level which is harder than the previous as the snake is long and the speed is also increase and the amount of food to collect to progress through the level gets larger.

Player gets score on the amount of food or dots it took. The speed depends on the user as the user decides in the beginning whether he wants fast, medium or slow. In my program, 1 is for fast, 2 is for medium and 3 is for slow. There is no option for retrieve if once the game is end. As you hit the obstacle the game ends the score is displayed to the user and ask the user if he wants to play again if yes press 1 else 0 to finish the program execution. Playing again option doesn’t means the game begins from that position it starts a totally new game and the score is reset to 0. Make sure the capslock is off as it doesn’t work for those keys.

Since this game is developed in the assembly language, I have used procedures for implementing the task. Following are some of the procedures that are used in the program.

|  |  |
| --- | --- |
| **DrawWall** | It is the part of the code that is called when the program is moved to the line call DrawWall is executed it automatically moves the flow to this label where the boundary wall for the game is created. |
| **DrawSnake** | It is responsible for drawing the snake. In this procedure another procedure called UpdatePlayer is called that is responsible for updating the snake body. |
| **CreateRandomCoin** | This procedure is responsible for initializing the coins at random positions for the snake in the board. |
| **CheckSnake** | This procedure is responsible to evaluate if the snake head collides with the wall or its body. If it collides the other procedure is called. |
| **EatingCoin** | If the user eats the dot/coin this procedure is called that increments the body of snake and the score is incremented by 1. |
| **YouDied** | This procedure is called as the head collides with body or wall. |
| **ReinitializeGame** | This procedure initialize the game from beginning where the score is reset, the snake body turns to original length. |

Apart from these procedure many built-in Irvine functions are used. The most important one is call function, flag registers are used to jump to specific part of code. Overall the goal of this program is to develop an efficient game for the users. This requires a careful planning and attention to detail, as well as a strong understanding of assembly language and the hardware components of the snake game. In this project, I have tried to apply the knowledge of assembly language that we learnt in the subject computer organization and architecture to complete all the tasks required. The main task of this project is to develop a simulation program of a snake game.

# Related Work

There are many versions of this game on different platforms. Even in the assembly language we can find many versions of it but with different libraries. I have used the Irvine32 library for this project.

* Snake.io
* Worms Zone.io
* Snake ’97:retro phone classic
* Snake Rival
* Snake game
* Slither.io
* Hungry snake

The names differ but all these games are same, its purpose is same as well as working. The only thing that differs is the language in which it is written and the platform on which it is being deployed.

# Flow Chart

1. **Initialize**

This initialize the game by drawing the board and asking the user for the desired speed.

1. **Draw snake**

After the speed, the snake is drawn on the board

1. **Move Snake and the availability of coin**

The snake moves within the board and looks for the coin as the head collides with the coin the score gets incremented and the length of the snake body is also increased.

1. **Colliding with the wall**

If the snake collide its head with the wall or with itself the game ends at that spot and users total score is displayed.

1. **Reinitialize**

If the user opt for new game the game is reset to its original setting and the process continues.

**Start**

**Select the speed**

**Press D to start the game**

**Move**

**Press a?**

**Press d?**

**Press w?** **Press s?**

Yes Yes

**Move left Move right**

Yes Yes

**Move straight Move down**

**Did snake get food?**

No

No Yes

**Touch wall?** **Add into score**

Yes

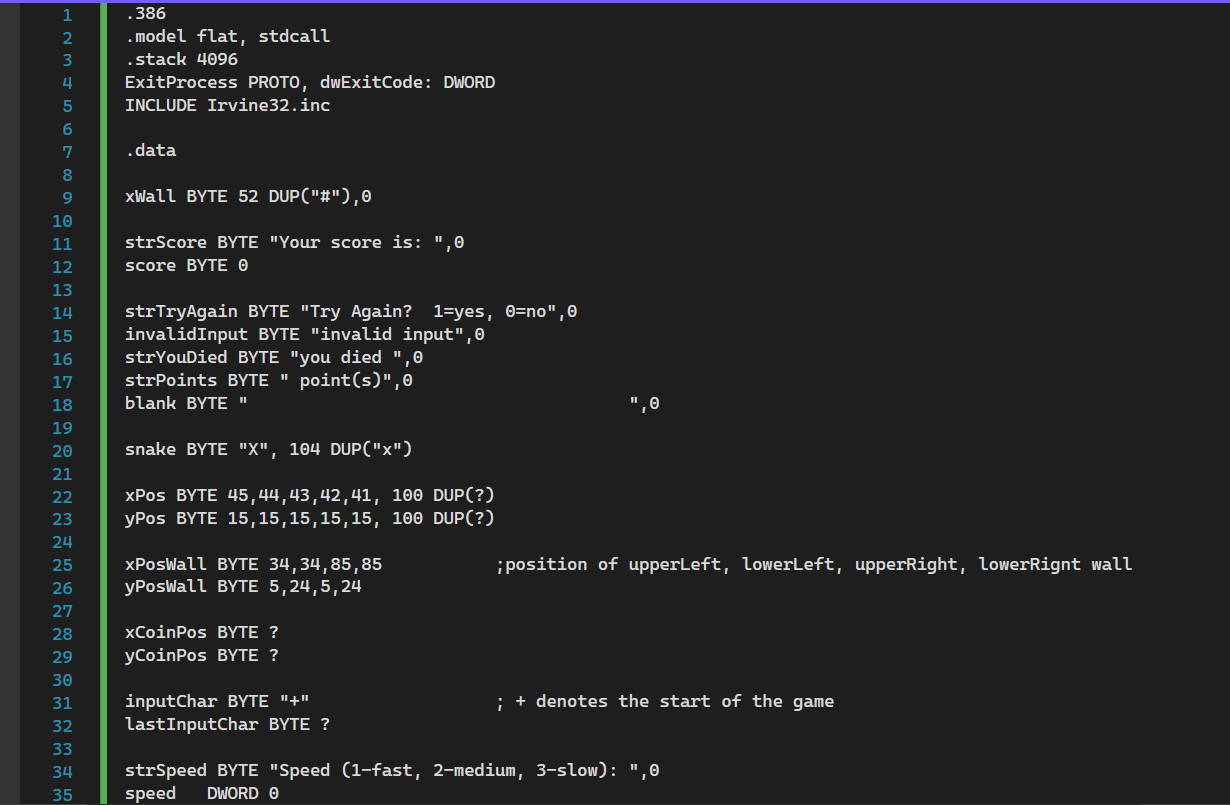
**End**

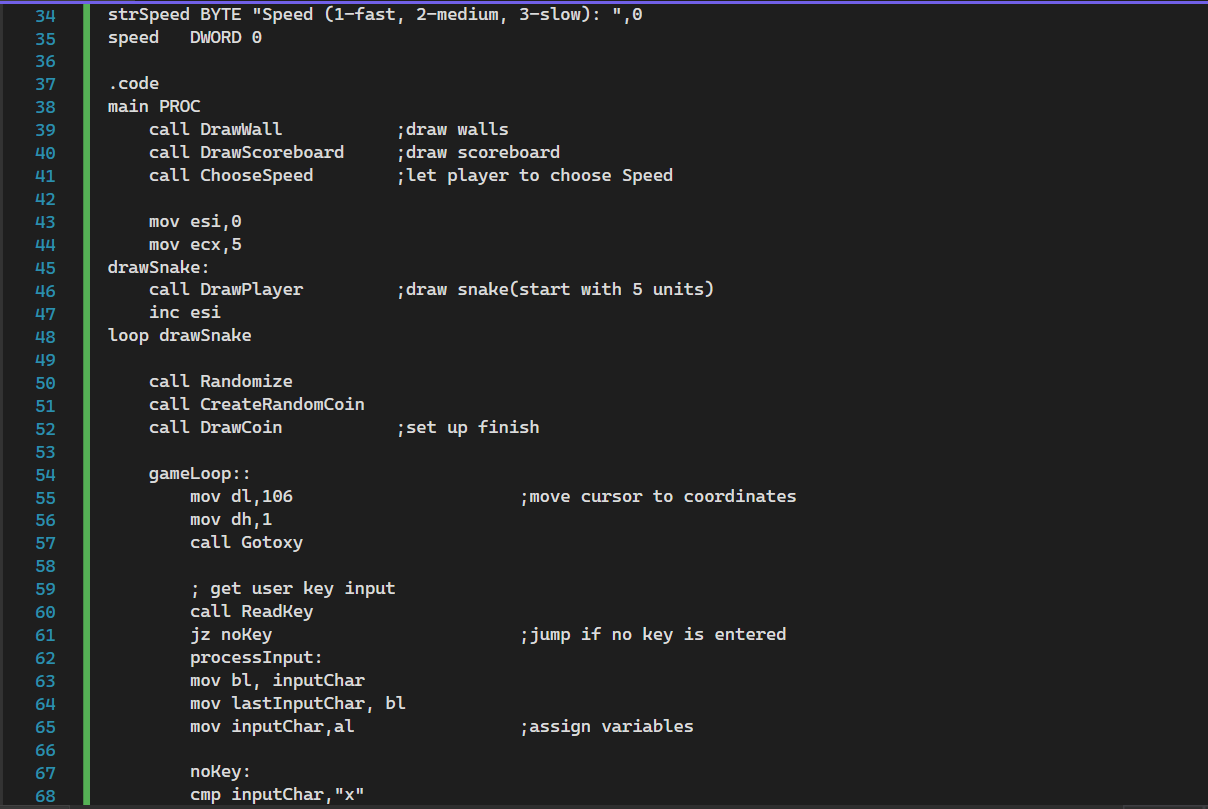
# Advantage

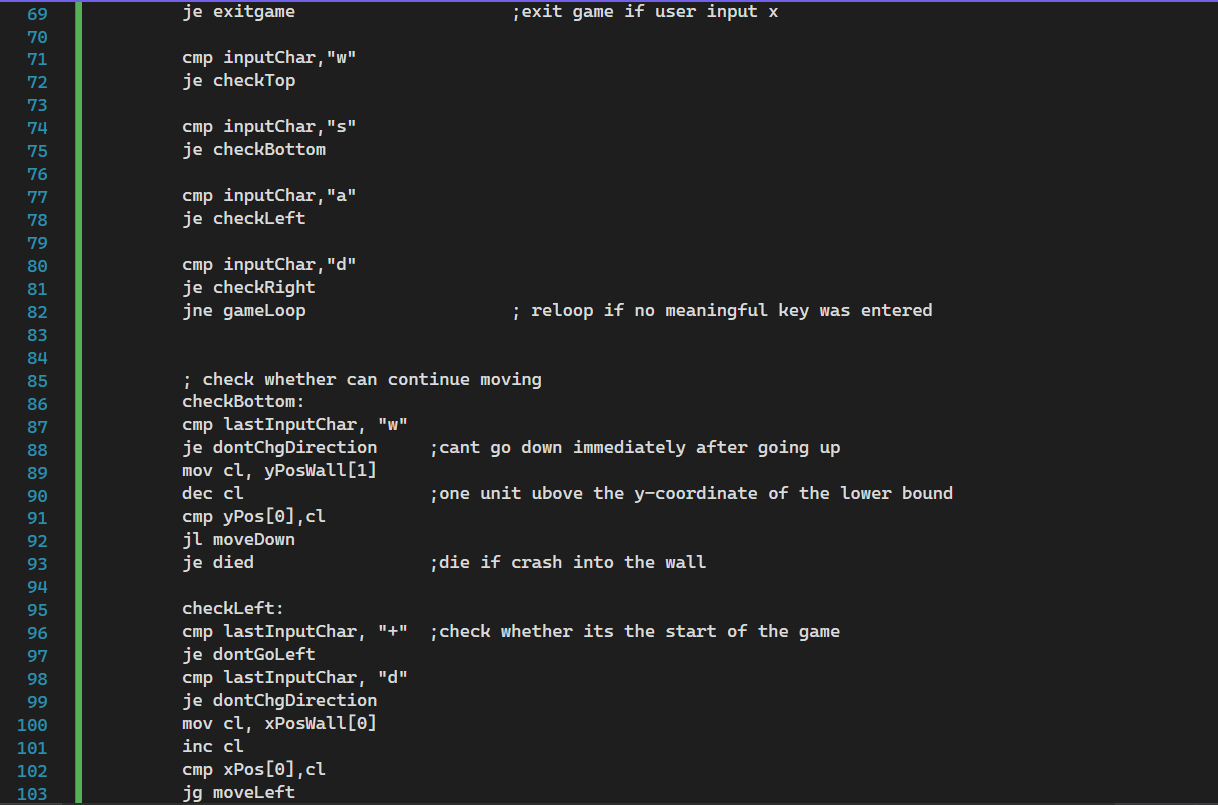
Following are some advantages of this project.

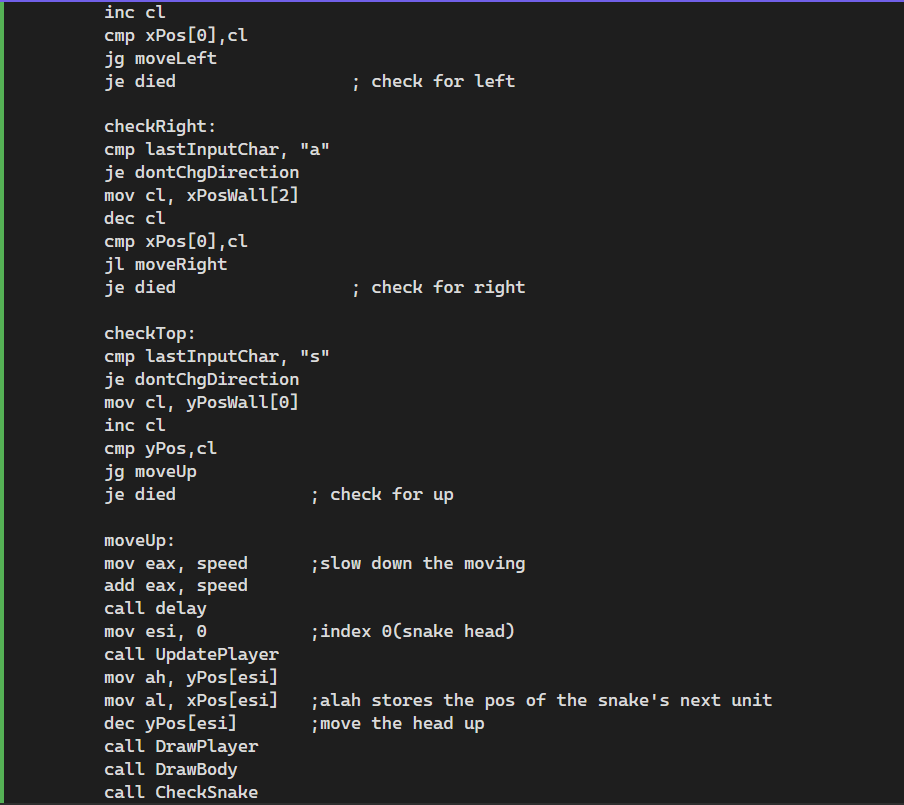
* It is an exercise of brain and improvement to problem solving.
* Relax and relief from anxiety and stress.
* Improve the skills.
* Mind training.
* Important for children for learning purposes.
* Strong memory.
* Power to decide.
* Powerful tool for children to have certain life skills.
* Memory efficiency as the assembly program requires less memory than high level languages which can be very beneficial for the systems with the limited memory resources,
* Control over the program. As the assembly language has much more control over the hardware and the code can be optimize for specific features.
* Debugging assembly languages are way easier than high level languages.

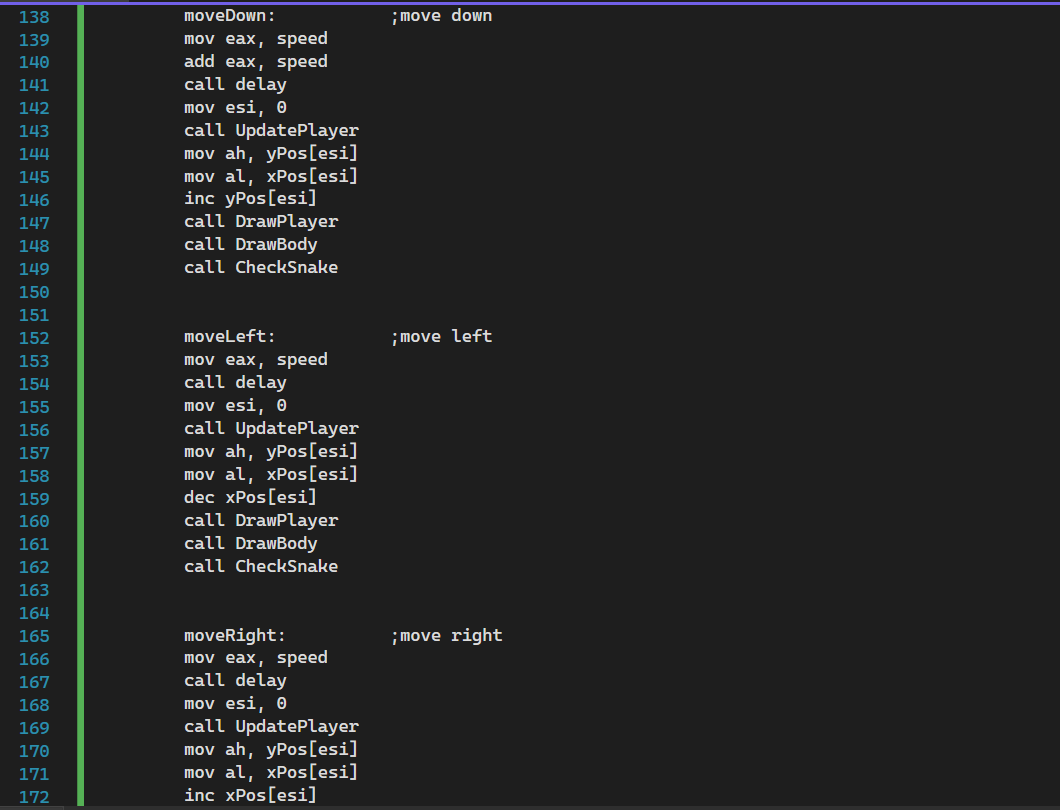
# Code

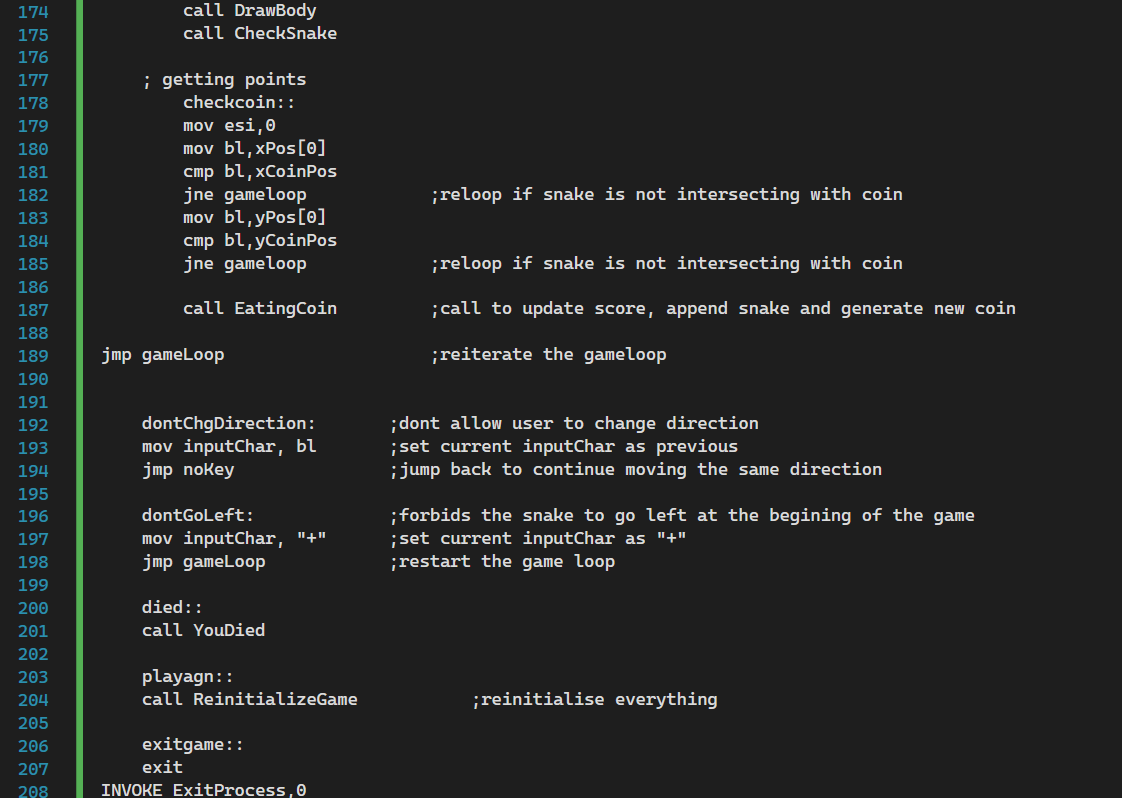


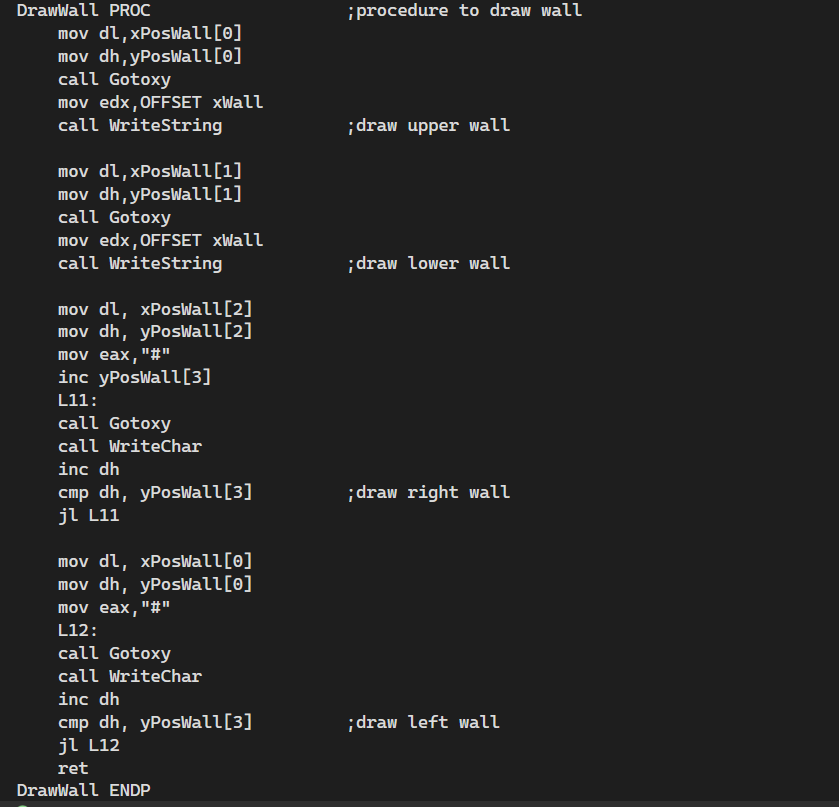


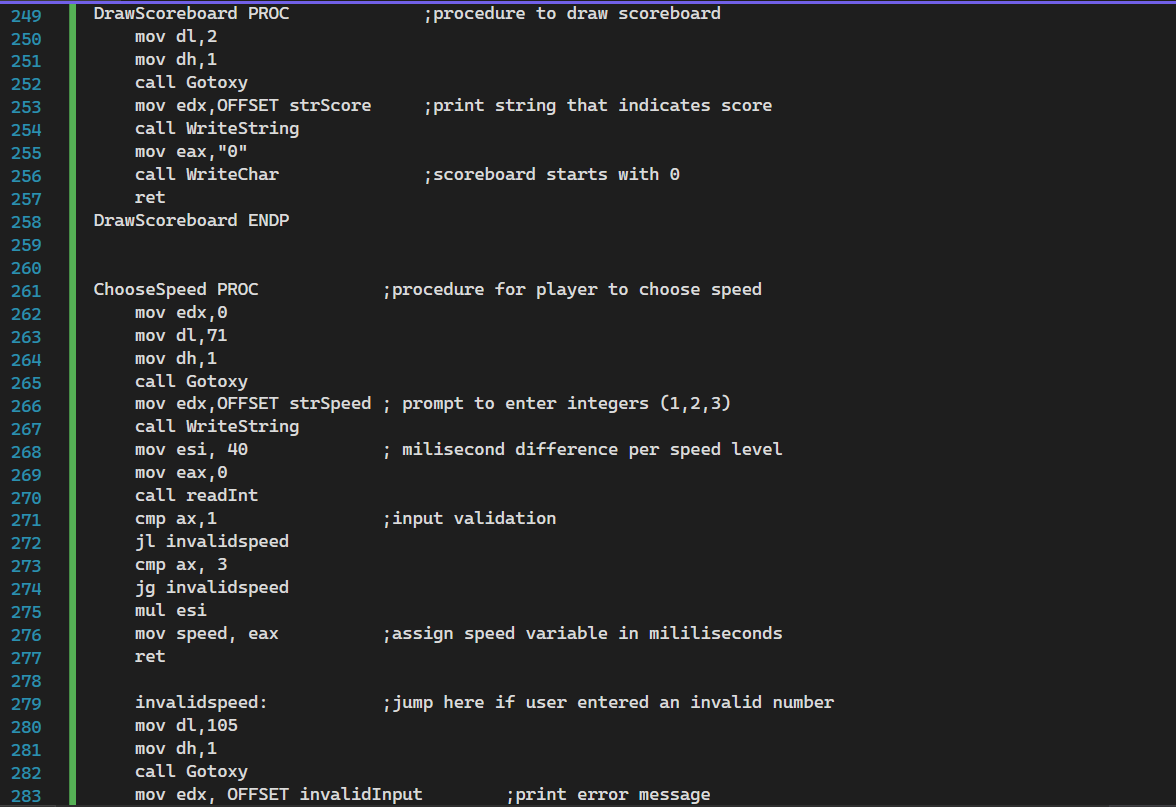


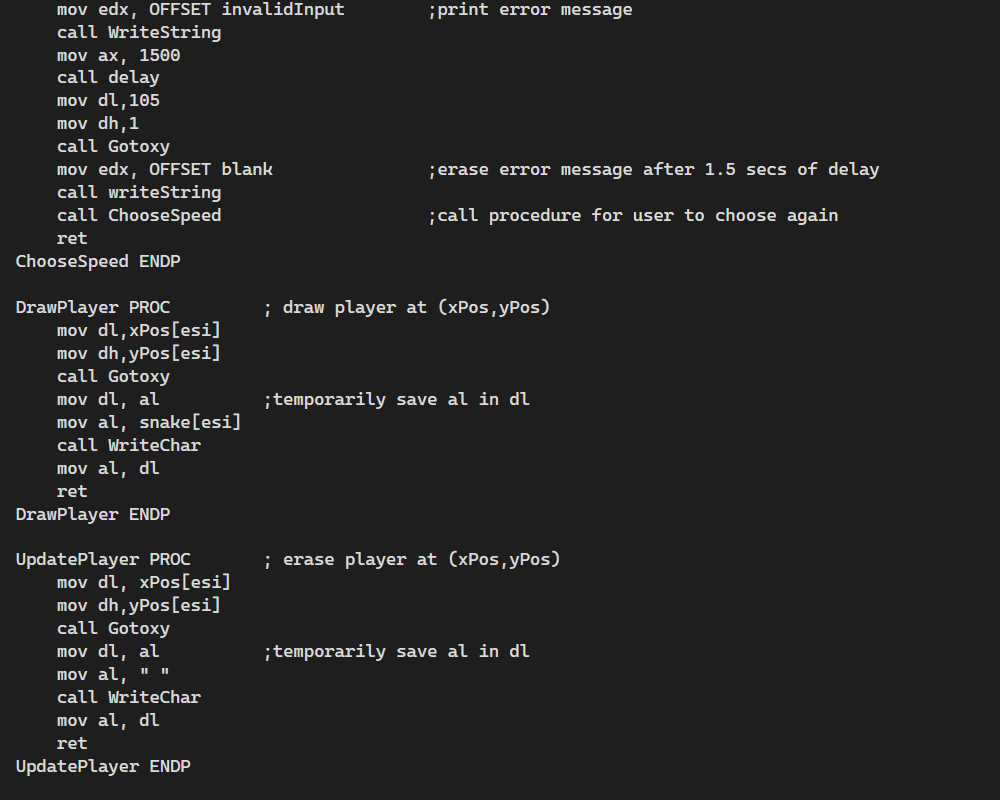


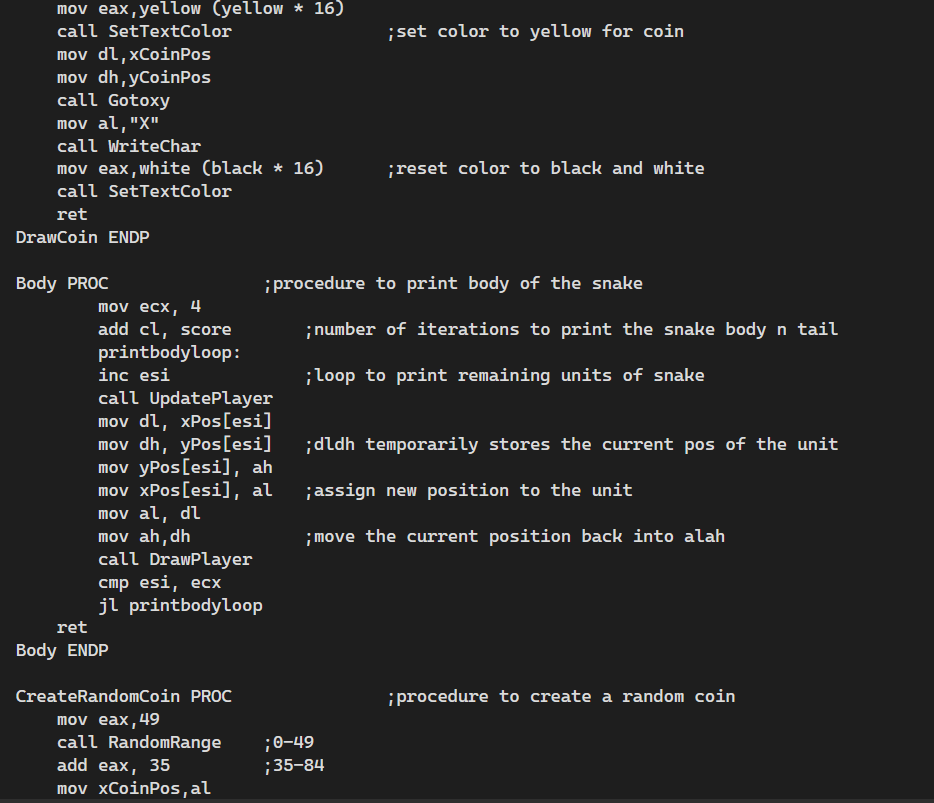


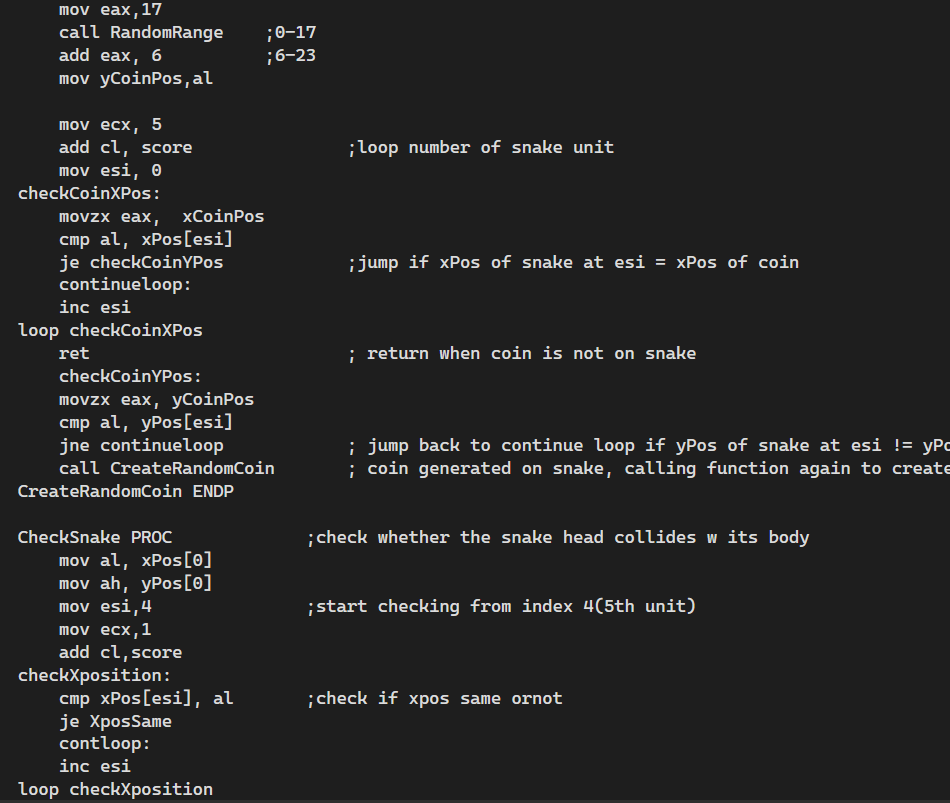


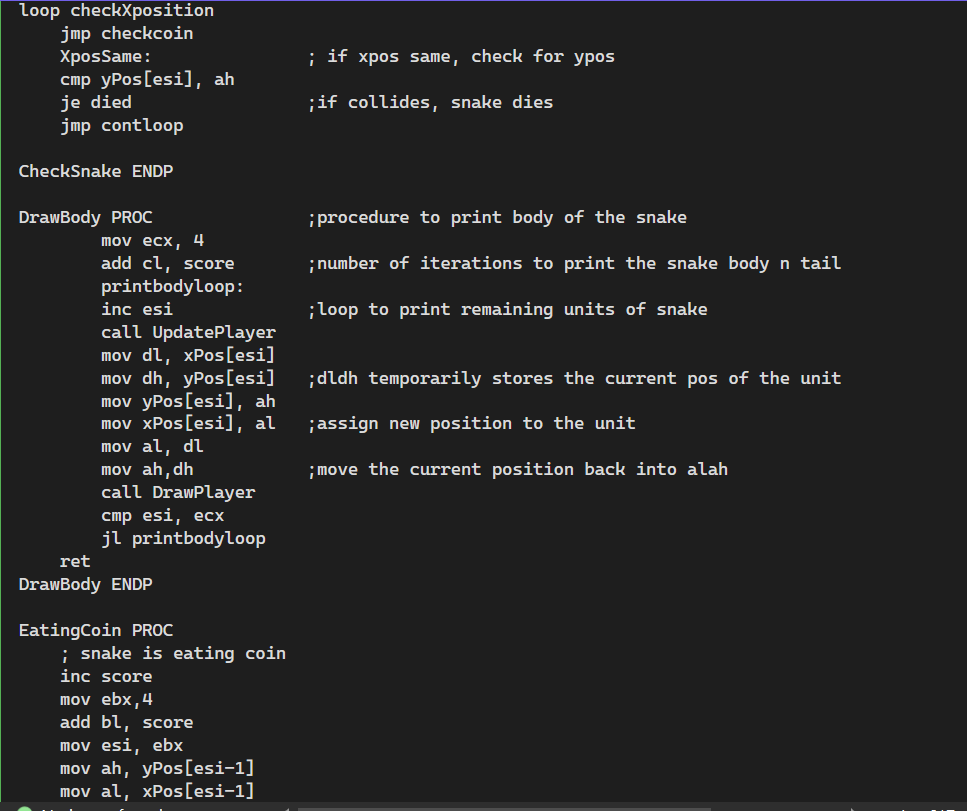


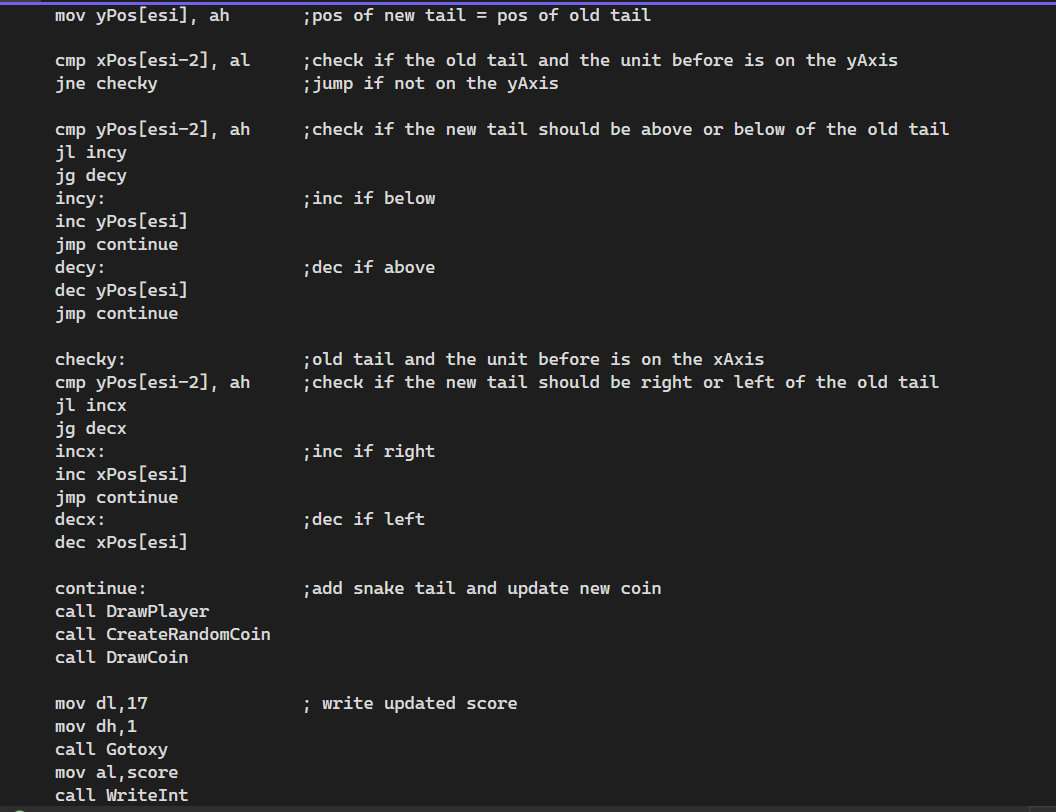


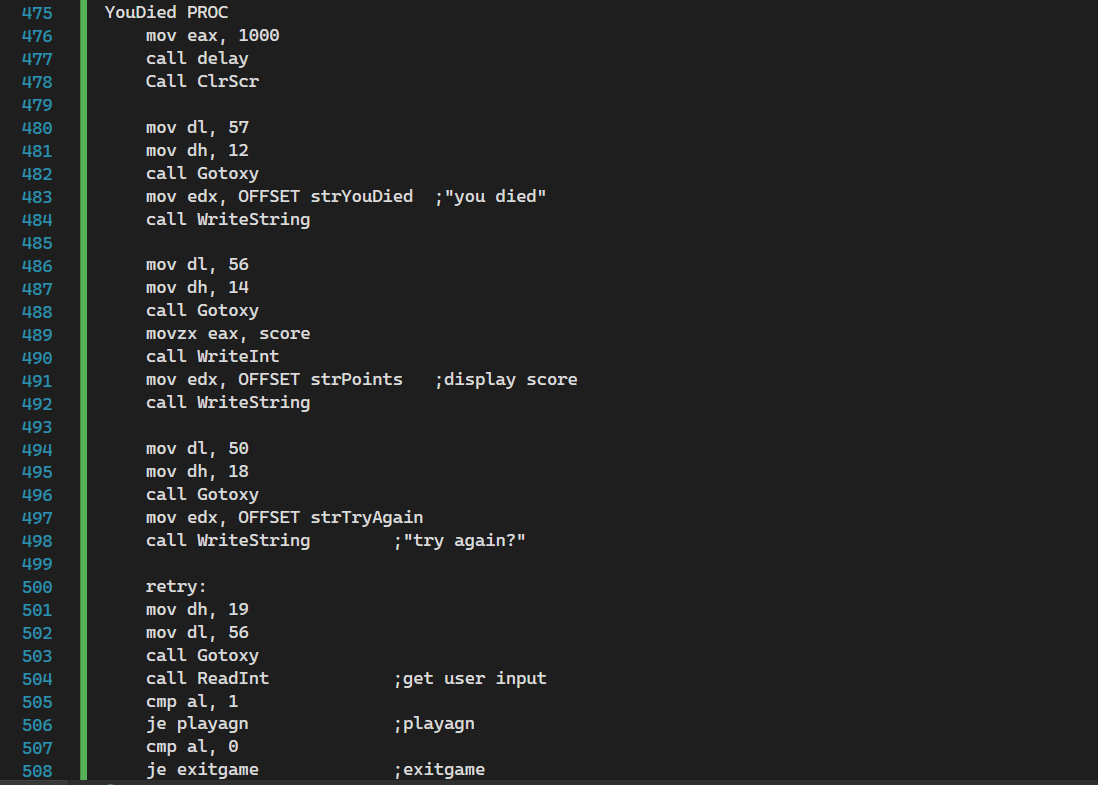


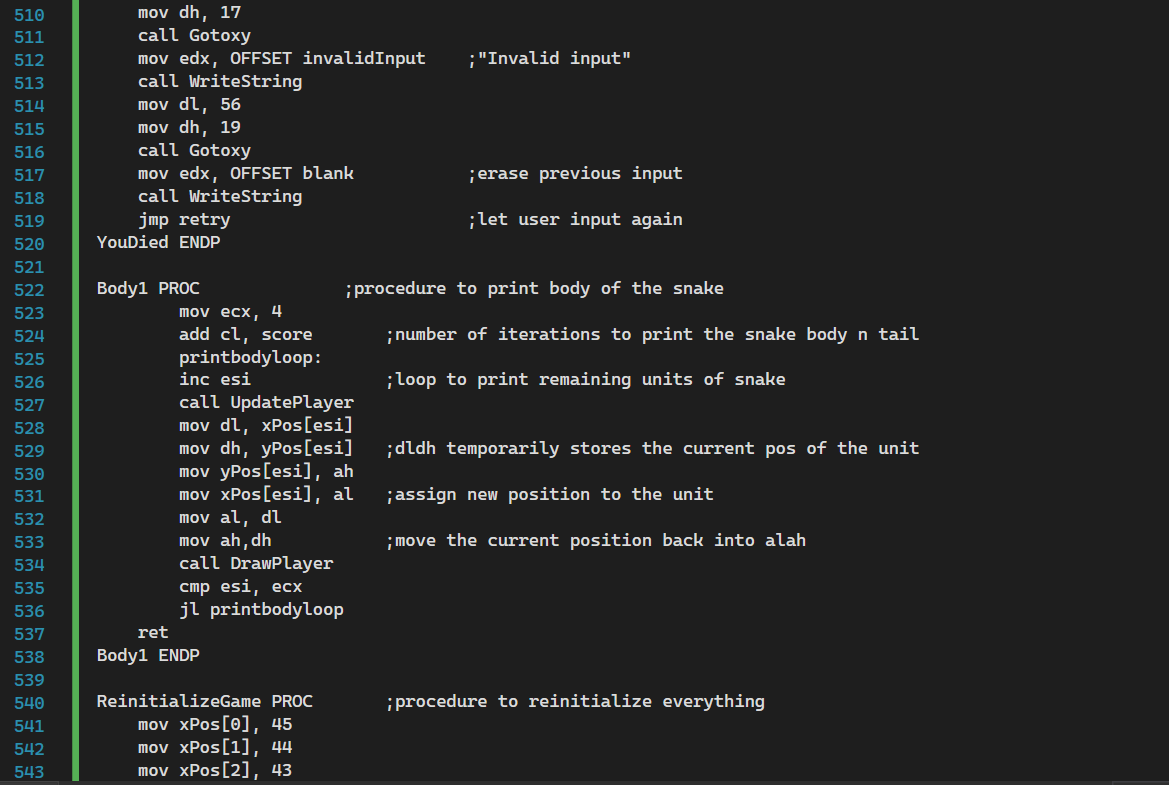


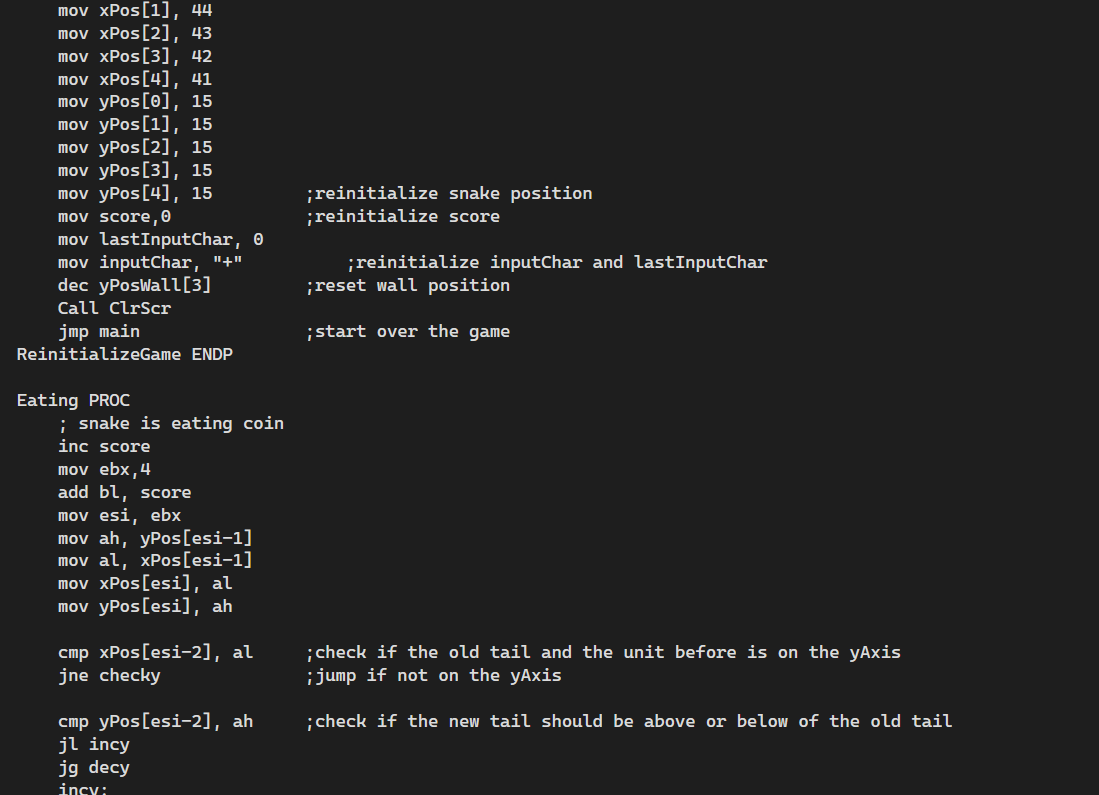


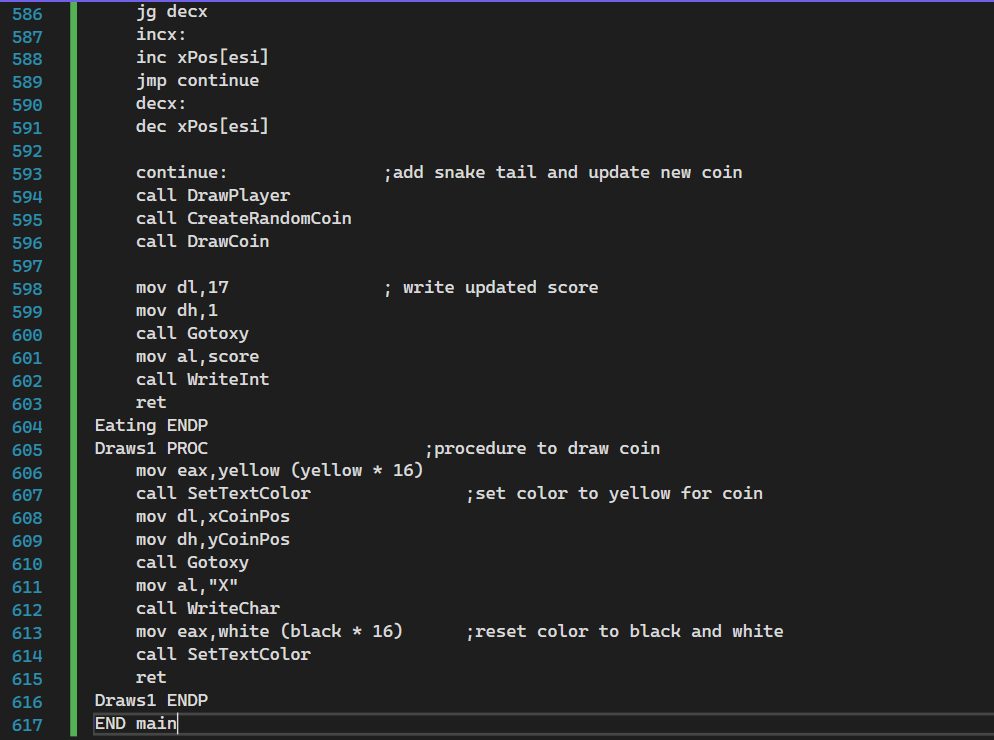




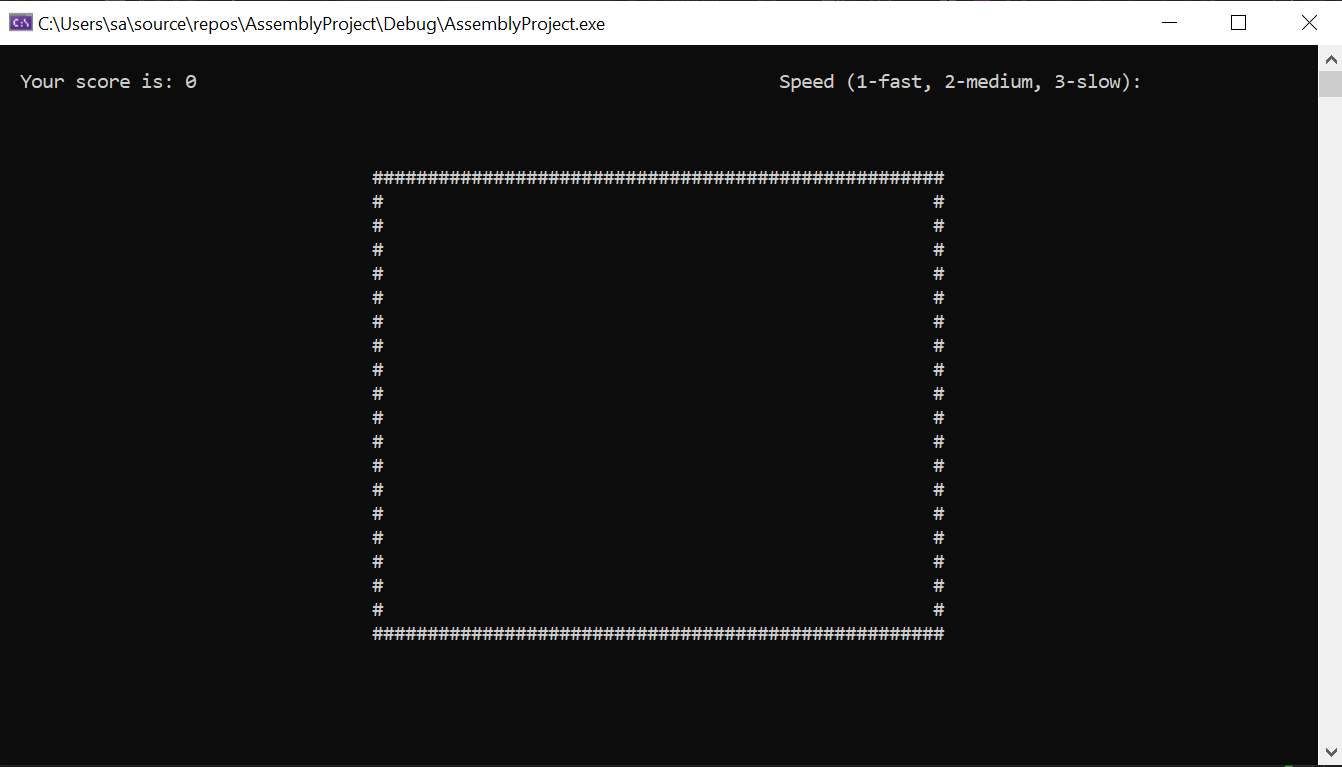




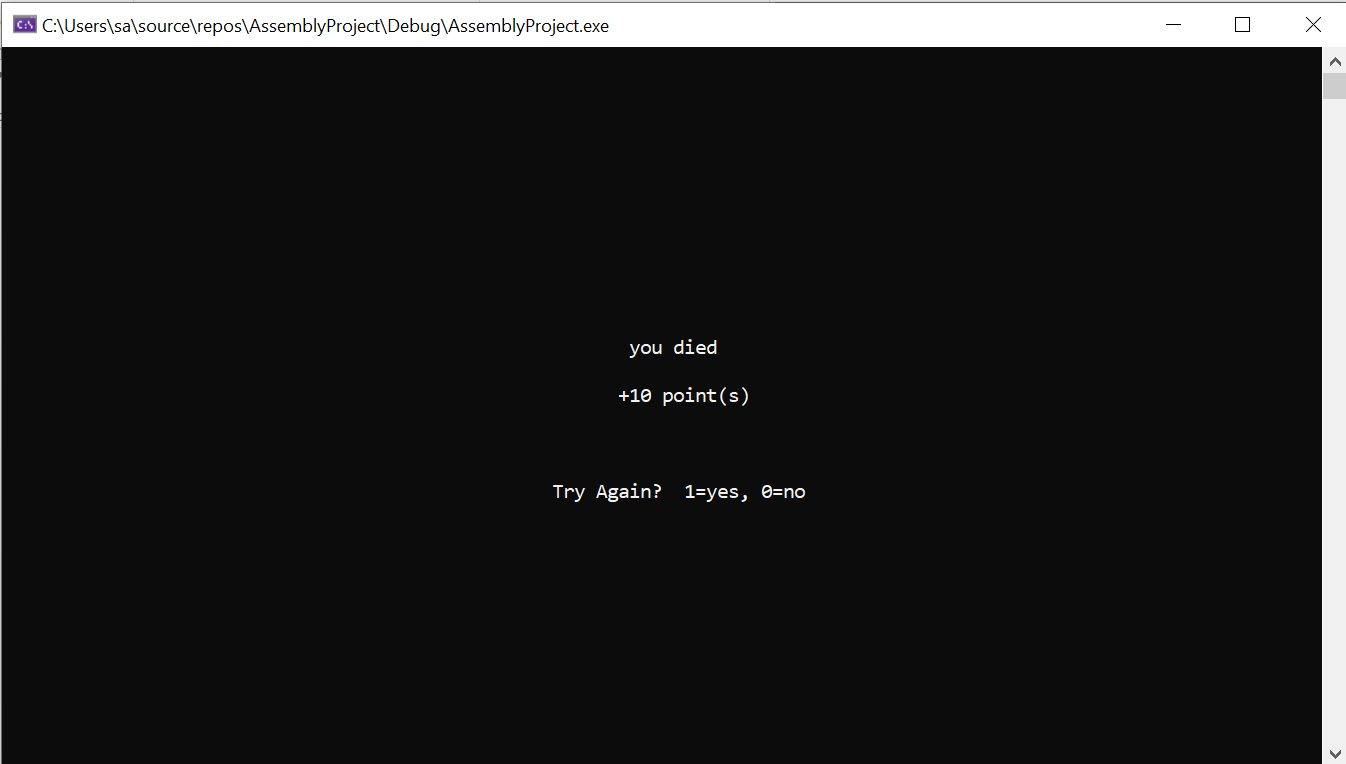




Output







# Conclusion

This project was about the snake game on assembly language. Though the graphics of the game is not that great but still the logic and algorithm that we have used works perfectly sine as any other snake game on any other platform. Procedures are used for each task be it initializing the board for the game, taking the speed on which the game will be proceeded, initializing the snake for the game, randomly assigning the coins in the board for the snake, the snake movement, the score count, the died procedure that executes when the snake hit the wall or its body, the final score display and the reinitializing function. This project is completed on the Irvine library.