Developing Space Invaders on a PIC18 Microprocessor

Laksh Gehani

Abstract: This is a proposal for implementing Space Invaders on a PIC18 microprocessor programmed in assembly. The focus of this project is to optimise the performance within a resource-constrained system. The game will feature real-time player movement, enemy AI, shooting mechanics and collision detection. The system will be evaluated based on several criteria such as the frame rate to ensure engaging gameplay and optimal performance.

I. Introduction

Embedded systems play a crucial role in modern computing, enabling low-power and high-efficiency applications ranging from industrial automation to consumer electronics. Recent advancements in processor technology have led to innovations in the gaming industry. A notable example is Intel's Core Ultra 200S processors [1]. These are designed to improve gaming performance through artificial intelligence.

One classic game that presents an ideal challenge for embedded implementation is Space Invaders, a popular arcade game first released in 1978 [2]. The game involves the player controlling a cannon to shoot down descending alien invaders while avoiding enemy fire. This project aims to implement Space Invaders on PIC18F86K22 microprocessor [3], programmed entirely in assembly language. The system will integrate a 128×64 LCD for graphical output, keypad for user input, and a buzzer for sound effects. The game logic, including player movement, enemy AI, projectile handling, and collision detection will be optimised to run efficiently within the limited resources of the microcontroller. The motivation for this project stems from the growing interest in developing cost-effective and low power gaming systems. The techniques used to produce this can be extended to further advancements in energy-efficient digital entertainment systems.

II. High Level Overview

A. Hardware Design

The game would consist of the following key hardware components:

1) PIC18F87K22 Microprocessor: The microprocessor serves as the central processing unit, coordinating the interactions between the input from the keypad and the output on the LCD and the buzzer.

2) 4x4 Keypad: The push buttons are the primary input method for controlling player movement (left, right) and shooting. After button press, a signal is sent to the microprocessor which processes the input and updates the game state accordingly. The size of the keypad can be reduced as only 4 buttons are required to play the game.

3) 128x64 Graphical LCD: The display is responsible for rendering the game elements such as the player, enemies, bullets and the score. The microprocessor communicates with the LCD via serial peripheral interface (SPI).

4) Buzzer: This provides the sound output for in-game actions such as shooting, enemy destruction and game-over events. The microprocessor generates sound by modulating pulse width modulated (PWM) signals to produce distinct sounds for different game events.

5) Power Supply: The system is powered by a stable 15V power source to run all components smoothly.

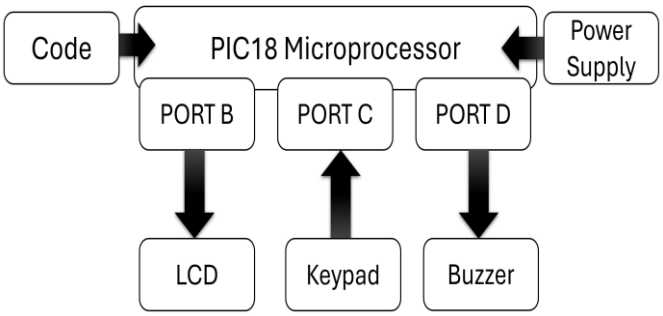

Figure 1: Hardware interaction diagram illustrating how the microprocessor interacts with its peripherals.

B. Software Design

The software for the system is written using MPLAB X IDE, purely in assembler. It is designed using a modular approach to help with code maintainability and debugging. The system follows a structured sequence of operations:

1) Initialisation: The microprocessor initialises all necessary peripherals. The system sets up interrupt service routines for handling user input.

2) User input handling: The input module detects button presses for player movement and shooting.

3) Game logic: The logic module updates the state of the game based on player actions and projectile interactions. It continuously checks for collision detection between enemies and bullets to ensure accurate scoring. It also includes conditions for game-over events based on enemies or player lives.

4) Graphics Rendering: Display module updates the LCD with new sprite positions for the player, enemies and bullets. It also displays the score and lives remaining.

5) Sound Generation: The audio module generates game sound effects using PWM based buzzer control.

6) Game continuity: The main game loop continuously checks for player input, updates enemy positions, checks for collisions, updates the display and triggers relevant sound effects.

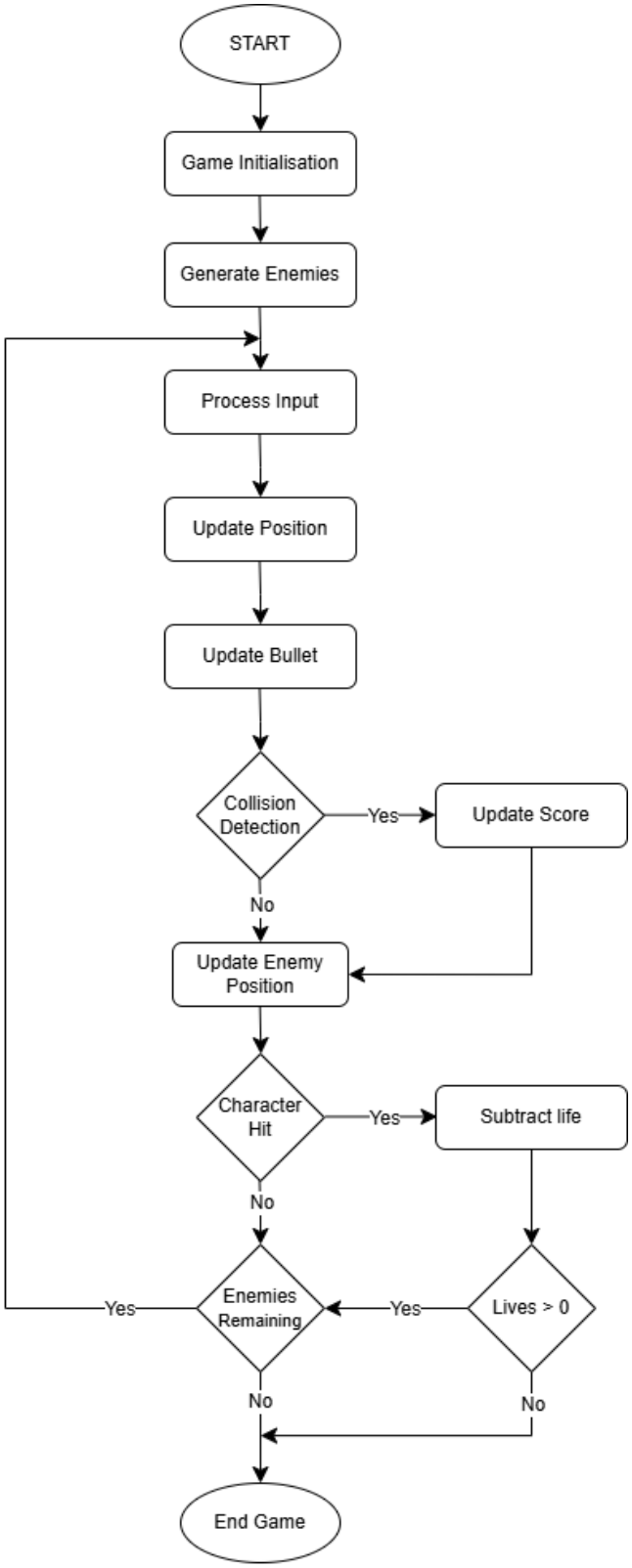

Figure 2: High level flowchart of the game loop showing the execution sequence of the game.

III. Constraints & Optimisations

Implementing Space Invaders on a microprocessor presents several hardware and software constraints that must be addressed to ensure efficient performance. Firstly, the microprocessor has a limited memory capacity. Storing all sprites may cause memory overflow leading to corrupted graphics. To address this, reusable graphical elements are used instead of storing unique sprites for each frame. Furthermore, the microprocessor has a limited clock speed and can only execute one instruction per cycle. The real-time input handling may cause delays due to excessive CPU polling, making the game feel unresponsive. To counteract this, interrupt driven input will be implemented, allowing the system to immediately process player commands. Another potential challenge is button debouncing, where rapid button presses may cause unwanted repeated actions. This will be mitigated using interrupt-based filtering to ensure each button press is processed correctly. Finally, the game is displayed on an LCD, which has limited refresh rate, meaning that full-screen updates can cause slow rendering. This may degrade the gameplay experience. To overcome this, only the necessary portions of the display will be updated rather than refreshing the entire screen.

IV. Performance Criteria

The success of the project will be evaluated by the following quantifiable criteria:
1) Frame update rate: The game should maintain a consistent frame refresh rate on the LCD to ensure flicker-free rendering.
2) Input latency: Player inputs should be processed instantly ensuring that there is minimal delay between the button press and the corresponding actions.
3) Memory usage efficiency: The software should be designed to ensure that all sprite storage, game logic and execution loop are optimised to operate within the limited memory capacity.
4) Collision detection accuracy: The game should accurately detect bullet and enemy collisions in real time. Every hit or miss of a bullet should be correctly processed within the game loop.

Additionally, there are some qualitative aspects that contribute to the overall performance. Firstly, the system should be reliable and stable over extended periods of operation. The game should function without any crashes or logic errors. Furthermore, the system should have seamless interactions between the microprocessor and the peripherals. SPI communication with the LCD and the GPIO input handling will be evaluated for proper synchronisation. Finally, the game controls should feel intuitive and responsive, giving the player a smooth and engaging experience.

V. Conclusion

This report demonstrates the feasibility of implementing Space Invaders on a PIC18 microprocessor, addressing challenges such as memory constraints and processing limitations. The system integrates the microprocessor with a LCD, keypad and buzzer to provide a fully functional game. The game will be tested on key performance criteria, including frame rate, input latency, memory efficiency and collision accuracy. The final project aims to utilise the optimisations mentioned above to produce smooth gameplay and real-time responsiveness.

REFERENCES

[1] Campbell, M. (2025, January 6). Intel extends its Core Ultra 200S lineup with 12 new CPUs. Overclock3D.
[2] Britannica, Space Invaders, Video game series, Britannica.com
[3] Microchip Technology Inc, PIC18F86K22/PIC18F87K22 Data Sheet

APPENDIX

Cycle 1 feedback: There was good background research done on energy costs and use, providing strong motivation for the work An area that can be improved is to ensure that your conclusions are quantifiable (where possible), very clearly. “Power can’t sustain a house”. Does that mean “solar energy is a good alternative” or not.