National University Of Emerging Sciences and Technologies

Bit Blaster

Instructor - Mr. Misbah Haider Malik

Section - BCS-3J

Group Members -

Muhammad Mufeez - 23K-0800

Haseeb Asif - 23K-0539

Azaan Hussain Shah - 23K-0568

Introduction:

"Bit Blaster" is a space-themed arcade game, inspired by the classic Space Invaders, developed entirely in assembly language. What sets this project apart is the inclusion of a custom bootloader, which will initialize the system and load the game directly into memory, eliminating the need for an operating system. This approach challenges our understanding of both system-level programming and game development, as we take full control of the hardware from the moment the machine powers on. "Bit Blaster" combines the technical complexity of bootloading with the creativity of game design, offering a unique opportunity to explore low-level programming, optimization, and system management while delivering a fast-paced and nostalgic gaming experience.

Problem Statement:

Most game development projects in high-level languages take place within an established operating system, bypassing the intricate process of booting up the system and directly interacting with hardware. This limits the understanding of how a computer initializes and how games can operate in a minimalistic environment. To fully appreciate the process of game development on a lower level, it's necessary to build both a bootloader and the game itself, providing end-to-end control from system start-up to gameplay. The challenge is to demonstrate how a game can be created and executed without relying on an existing OS, showcasing the power of assembly language for both system initialization and game functionality.

Existing Solution:

Classic space shooter games like *Space Invaders* have been widely implemented in high-level programming languages such as C++, Java, and Python, with support from modern libraries and game engines. These implementations simplify development with rich tools and features but often rely on an existing operating system environment. However, they abstract away crucial hardware interactions, providing limited exposure to low-level system operations. The role of bootloaders and direct hardware management is often overlooked in such solutions, focusing only on game mechanics rather than system initialization and bare-metal control.

Proposed Solution:

Bit Blaster will be a complete, low-level solution, consisting of both a custom bootloader and a fully functional space invader-style game. The bootloader will initialize the system, prepare the hardware, and load the game into memory, demonstrating full control over the boot process. The game itself, written entirely in assembly language, will highlight efficient use of system resources, optimized performance, and direct hardware interaction. By combining the development of a bootloader with the game, this project will showcase the complete journey from power-on to gameplay, emphasizing the educational value of low-level programming and system management.

Salient Features:

- Custom Bootloader: The project includes a custom bootloader that initializes the system and loads the game directly into memory, bypassing the need for an operating system.
- Low-level Optimization: Developed entirely in assembly language, "Bit Blaster" demonstrates efficient CPU instruction-level optimization and memory management for smooth gameplay and performance.
- Complete Control from Power-On: From system startup to gameplay, the project showcases full control over hardware, giving insight into bootloading and direct hardware interaction.
- Retro Aesthetics: The game features classic, pixel-based graphics reminiscent of vintage arcade games, capturing the nostalgic feel of early space shooters.
- Compact Codebase: Assembly language allows for a highly compact and efficient codebase, resulting in minimal memory usage without sacrificing gameplay quality.
- Progressive Difficulty: The game increases in complexity as players advance, featuring faster enemies and more challenging attack patterns.
- Responsive Controls: Designed for quick player input, the controls allow for immediate action, making gameplay engaging and intense.

Tools and Technologies:

- Microsoft Visual Studio
- MASM (Microsoft Macro Assembler)
- VirtualBox/QEMU
- MakeFile (Optional)