Project Specification

Nemu: A Nintendo Entertainment System Emulator

Lucaz Lindgren

2025-06-13

Contents

1 Overview

This document provides an overview of the project development, establishing its objectives and goals. It includes some technical details and an outline of the system architecture, but it is not a technical specification.

2 Aim

This project aims to produce a cycle-accurate *Nintendo Entertainment System* (NES) emulator. This allows consumers to experience the vast library of classic games developed for the system, even without access to the deprecated hardware. Cycle-accurate precision ensures that games relying on precise timing between different system components will be compatible. This aims to reduce inaccuracies and graphical anomalies.

Recognizing that many later NES games utilized memory management controllers (mappers) to circumvent the console's limited address space, the emulator will prioritize support for the most frequently used mappers. Specifically, it will support mappers 0 (NROM), 1 (MMC1), 2 (UxROM), 3 (CNROM), and 4 (MMC3) [2].

The following peripheral input devices will be supported: original NES controller, Xbox controller, and keyboard.

The emulator will support both the Linux and Windows operating systems. This will be achieved by using platform-agnostic code libraries.

The emulator will not involve the distribution of any copyrighted game software. Users will be required to provide their game ROMs, which they are legally entitled to possess. The emulator is simply a tool for running legally obtained software on modern hardware.

2.1 Secondary Goals

Upon the implementation and verification of the core system, development will proceed to implement a series of secondary objectives. These are designed to improve the end-user experience and broaden game compatibility.

2.1.1 Graphical User Interface

A *Graphical User Interface* (GUI) will be developed to serve as the front-end for the emulator. The GUI aims to provide an accessible and user-friendly experience by abstracting technical complexities, helping users with limited technical knowledge. The GUI will allow users to manage their game library, configure settings, and display game-specific metadata.

The GUI will be engineered using *Clay*, an immediate mode C library based on the widely adopted Dear ImGui framework. Clay offers official binding for Odin, ensuring seamless integration with the codebase. This integration will involve developing a custom rendering backend for Clay.

2.1.2 Game Save State

To provide a modern gameplay experience that transcends the capabilities of the original hardware, a robust save system will be implemented. This will enable the user to capture a complete snapshot of the emulator state at any point during gameplay. This involves serializing the entire state of the emulator to a file on disk. This user can then reload this file at a later time to restore the system state.

2.1.3 Additional Mappers

Support for additional mappers to provide broader game compatibility.

3 System Overview

3.1 Background

The Nintendo Entertainment System (NES) is an 8-bit home video game console first released in the North American market in 1985. It is widely regarded as one of the most significant video game consoles ever produced, and its launch is frequently credited with revitalizing the American video game industry following the market crash of 1983.[1] The console has become a popular target for emulation due to its influential legacy and simple architecture.

The core system architecture of the NES is composed of four primary components: the *Central Processing Unit* (CPU), the *Picture Processing Unit* (PPU), the *Audio Processing Unit* (APU), and 2 kilobytes of onboard *Random Access Memory* (RAM).

3.1.1 Central Processing Unit

The NES is based on the NMOS version of the *6502 processor* running at 1.79 MHz. It is a little-endian 8-bit processor with a 16-bit address bus, allowing the processor to access up to 64 KB of memory space. The memory space is split into 256 pages, each consisting of 256 bytes. The stack is permanently fixed in page 1 ($0100-$01FF) and controlled by the stack pointer (S) register. All i/o accesses are memory-mapped [3].

3.1.2 Picture Processing Unit

The PPU is a specialized co-processor responsible for generating the composite video signal designed to be received by a television. It operates independently of the CPU and manages its dedicated 16 kilobyte address space for graphical data. The address space consists of a combination of internal VRAM and cartridge-based ROM and RAM. In addition to its main address space, the PPU contains two distinct internal memory regions. It features 256 bytes of Object Attribute Memory (OAM) used to store the attributes for up to 64 on-screen sprites. It also contains 32 bytes of palette RAM, which defines the available on-screen colors [4].

3.1.3 Audio Processing Unit

The APU is responsible for generating all of the audio waveforms. It is comprised of five channels: two pulse wave generators, a triangle wave, noise, and a data modulation channel for playing DPCM samples. Control over the channel parameters is managed through a series of internal registers. These are memory-mapped and therefore accessed through specific addresses of the CPU's address space [5].

3.1.4 Mappers

NES games come in cartridges, which contain various circuits and hardware. The configuration and capabilities of such cartridges are commonly referred to as their *mapper*. These are designed to circumvent the NES limitations, such as adding RAM to the cartridge or extra audio channels [2].

3.2 Technical Prerequisites

This project will be developed using the *Odin* programming language. As a modern derivative of C, Odin is designed for low-level systems development, offering manual memory management which grants fine-grained control over resource allocation. Its strong emphasis on high performance and reliability makes it an exemplary choice for building an efficient emulator.

To handle platform-specific operations, the emulator will leverage *Sokol*. Sokol is a cross-platform suite of libraries that provides a unified API for managing window contexts, graphics hardware (via OpenGL, Direct3D, Metal), audio, and user input. It has official Odin language bindings, which ensure seamless integration into the codebase. Where direct interoperability is required, particularly for interfacing Sokol with other C-based dependencies, the C language will be utilized.

The build system will be managed through Bash scripts, providing a flexible and transparent method for compiling the source code and managing dependencies across target platforms. Version control will be managed through Git and hosted on GitHub. The goal is to maintain a working version of the project on the main branch and create diverging branches for experimental features.

The emulator will be compatible with both Linux and Windows operating systems.

3.3 Architecture

The architecture will be developed with a strict separation between the core emulation logic and the platform-specific frontend. This modular design is intended to facilitate portability, allowing the emulator to be adapted for additional platforms, like macOS and WebAssembly.

The emulator will employ a multithreaded execution model to improve performance. A dedicated thread will be allocated for the primary emulation loop, while a separate thread will manage the rendering processes, including the game itself and the user interface (if included).

4 Milestones

Milestones represent discrete stages and deadlines within the development lifecycle. Table 1 outlines the planned milestones and their projected completion dates. Multiple milestones will be worked towards simultaneously.

Nr	Description	Date
1	Project specification complete	2025-06-13
2	Sample project including Sokol and Clay	2025-06-13
3	Stable CPU implementation	2025-06-20
4	NES ROM and mapper 0 support	2025-06-20
5	Partial PPU support	2025-06-27
6	Running simple NES games	2025-06-27
8	PPU scrolling support	2025-07-08
9	APU support	2024-07-14
10	Support for additional supported mappers	2024-07-18
11	Support for multiple platforms	2024-07-23
12	Fully passing test suite	2024-07-23
14	Secondary goals	2025-07-31
15	Code complete	2025-08-01
16	Quality assurance finished	2025-08-06
17	Project report complete	2025-08-13
18	Project presentation complete	2025-08-15

Table 1: Project milestones ordered from earliest to latest due date.

5 Assignments

The total time allocated for the project is 280 hours distributed among the assignments listed in table 2. The table also shows the estimated completion time for each assignment.

Description	Estimated Time
Project specification	7.5 h
Project report	25 h
Project presentation	7.5 h
Basic project setup including Clay and Sokol	10 h
Core emulation implementation	135 h
Support for multiple platforms	15 h
Secondary goals	55 h
Quality assurance	15 h
Code polishing and documentation	10 h

Table 2: A list of assignments and their estimated completion time.

6 Quality Assurance

The emulator will undergo a multifaceted quality assurance (QA) process to ensure a qualitative user experience. The testing methods will target one of three specific areas determining the quality of an emulator: accuracy, compatibility, and performance.

The goal of the emulator is to replicate the behavior of the original hardware. This will be achieved through the use of test ROMs. These are software programs designed to stress individual hardware components and verify adherence to critical specifications. This may include cycle-accurate timing constraints.

Additionally, game compatibility will be tested by manually playing a wide variety of games, relying on different hardware behaviors. This is to verify compatibility for games using the supported mappers.

Unit tests will be developed to verify the functional integrity of subsystems.

As a secondary objective, performance testing will be conducted. This may involve memory profiling. However, due to NES's low performance requirements relative to modern hardware, this is not a priority.

References

[1] *Nintendo Entertainment System*, Nintendo Wiki [Online]. Available: `https://nintendo.fandom.com/wiki/Nintendo_Entertainment_System#`

[2] *Mapper*, NESdev Wiki [Online]. Available: `https://www.nesdev.org/wiki/Mapper`

[3] *CPU*, NESdev Wiki [Online]. Available: `https://www.nesdev.org/wiki/CPU`

[4] *PPU*, NESdev Wiki [Online]. Available: `https://www.nesdev.org/wiki/PPU`

[5] *APU*, NESdev Wiki [Online]. Available: `https://www.nesdev.org/wiki/APU`