-10em

Requirements Document for KnavesNES

Niko Savas	Joe Crozier	Sam Nalwa
1300247	1311502	1332792

December 7, 2015

Software Engineering 3XA3, Lab #3

Contents

1 General Introduction

KnavesNES is a fully functioning MOS Technology 6502 CPU [1] emulator, the very same CPU included in the Nintendo Entertainment System [2], first sold in 1983. An emulator is a piece of software which at its core, aims to imitate a functioning piece of hardware. An emulated CPU is capable of processing instructions in the form of OP codes which direct the CPU to perform operations on values in its memory. In the case of KnavesNES, the instructions will be processed through ROM files which can be created by the user using an external tool, or alternatively downloaded from the internet.

The motivation behind developing a CPU emulator is that of academic pursuit. The architecture behind a modern CPU is incredibly complex and can be formidable for amateur software developers to consider developing for it on a system level. KnavesNES aims to provide a platform for software developers interested in learning about assembly to test and debug their 6502 code. In addition to this, educators will be able to use the software to aid in teaching their students about CPU architecture. Many have compared the pursuit of learning a language like 6502 to be akin to learning a written language like Latin. While both are antiquated and in general no longer used, both can foster a better understanding and mastery of modern languages like Java or English. For these reasons, educators and their self-teaching computer enthusiasts are the main stakeholders and by extension users of the project.

The main scope of this project is to simulate the execution of compatible OP code instructions on an emulated CPU [3]. The emulated CPU should be able to process the same operation codes, represented as hexadecimal numbers, as the physical CPU. The CPU will manage its own RAM, in this case 2kB [2]. The CPU must be able to process instructions from a ROM file, as well as store the contents of this ROM file in memory. Tests are available as ROM files with sample instructions meant to test the fidelity of the CPU.

2 Definitions

NES - Nintendo Entertainment System. Game console released in 1983. The architecture of the system has been examined thoroughly and replicated in emulators such as this one. **CPU** - Central Processing Unit: The part of a computer which executes instructions by performing basic arithmetic, logical, control, and input/output operations. **Op Code** - Operation code. This is an instruction in machine language that is sent and executed by the CPU. **ROM file** - A file that contains the read-only information that would be present in a game cartridge, in this case an NES cartridge. This file can be read and executed by an emulator as if it was a real cartridge.

3 General System Description

3.1 Module Description

MainNES Main executable which is launched through the command prompt. This module will handle the launching of the actual ROM file and the management of the CPU. It is also responsible for ending the process when necessary, and for instructing MemoryLogger when to log the CPU memory.

CPU The CPU is passed the set of instructions from the ROM file from MainNES, stores them in its memory, and begins their execution immediately. It is responsible for informing ErrorHandler when an error occurs in execution, and also for informing MainNES when execution is complete.

ErrorHandler The ErrorHandler is responsible for deciding the outcome of various errors which can occur while executing or preparing to execute. Examples include invalid ROM formats, unknown instructions, instructions with undefined results, and out of bounds memory accesses. Typically, these errors will result in ErrorHandler informing MainNES to halt execution and echo into the command prompt that said error has occurred.

Memory - The Memory module is responsible for managing the state

of its memory, and logging said memory as ASCII formatted .log files which are placed in the user's working directory.

3.2 User Characteristics and Conditions

Users of KnavesNES are expected to be comfortable using the command line to execute software. They are also expected to have a basic understanding of a CPU and the concept of memory in order to fully grasp the project. They are expected to be running on a Windows, Mac OS X, or Linux operating system on a 32 or 64-bit processor.

3.3 Assumptions and Dependencies

It is assumed that the ROM files to be executed are of a reasonable size and length (¡ 512kb). This is due to memory limitations of KnavesNES, as found in the NES console itself. It is also assumed that any ROM files ran by KnavesNES consist only of valid OP code instructions found in [3]. KnavesNES is exceptionally portable in the sense that once compiled, it does not rely on or reference any external libraries.

4 Functional Requirements

- KnavesNES must be able to run standard NES ROMs (.nes files)
- KnavesNES CPU must be able to execute opcodes from the 6502 MOS CPU architecture
- KnavesNES CPU must be able to manage 2kB of RAM
- KnavesNES will log its memory state into a .log file in the user's working directory at a specified interval if requested.

5 Non-Functional Requirements

5.1 Quality Requirements

- KnavesNES should be built to exactly mimic the 6502 MOS CPU architecture, including any abnormalities with the running of opcodes

5.2 Look and Feel Requirements

- KnavesNES should be run as an .exe through the command line.
- KnavesNES
- KnavesNES must allow the user to select a .nes file from their computer to run

5.3 Usability Requirements

- KnavesNES must be able to be used by anyone over the age of 10
- KnavesNES must be easy to learn for a 10 year old

5.4 Performance Requirements

- KnavesNES should mimic the performance of a 6502 MOS CPU
- KnavesNES must not crash during any CPU tests
- KnavesNES CPU cycles must be precise down to the nanosecond (based off computer system time)
- KnavesNES must be able to run ROMs up to 1MB

5.5 Operational Requirements

- KnavesNES must be able to run on a Windows 10 Machine as an .exe

5.6 Maintainability and Portability Requirements

- KnavesNES should be easy to maintain for anyone through the GitHub repo
- KnavesNES should be written so that it is easy to jump into for a new collaborator
- KnavesNES should be able to fit on a 1GB USB to ensure portability

5.7 Security & Safety Requirements

- KnavesNES will not access any files other than the supplied ROM
- KnavesNES will not have any internet connectivity
- KnavesNES should be incapable of hurting a person or machine

5.8 Cultural and Political Requirements

- KnavesNES will not include any offensive code or words for any culture or political affiliation

5.9 Legal Requirements

- KnavesNES will be released under the GNU General Public License v2
- The full source of KnavesNES will be hosted on GitHub

6 Use Cases (Scenarios)

1. **Product Use Case Name**: Execution of ROM
   Trigger:User executes a local ROM file via a command through the users command prompt. The name of the file is included as an argument. KnavesNES begins executing the instructions found in the ROM

file.
Preconditions:KnavesNES is not already running.
Interested Stakeholders:Users and educators.
Actors:MainNES, CPU, ErrorHandler
Outcome:KnavesNES will execute the ROM file to completion, or until the user stops the process. State of memory will be logged at regular intervals (as explained in the memory logging use case). Upon completion (program counter reaches the end of the instructions), KnavesNES will echo to the command prompt that execution was successful and will gracefully close its process.

2. **Product Use Case Name**: Halting of Execution
Trigger:User presses the ESC key during execution of a ROM file.
Preconditions:KnavesNES is currently executing a ROM file.
Interested Stakeholders:Users and educators.
Actors:MainNES, CPU, ErrorHandler
Outcome:KnavesNES will halt executing the ROM file, log the final state of memory, and gracefully close its process.

3. **Product Use Case Name**: Memory Logging
Trigger:A specified amount of instructions have been processed since the last successful memory log.
Preconditions: KnavesNES is currently executing a ROM file.
Interested Stakeholders: Users and educators
Actors: CPU, MainNES, ErrorHandler, MemoryLogger
Outcome: KnavesNES will log the full state of memory with a timestamp into a MEMORY-DD/MM/YY-HH:MM:SS.log file in the users working directory.

4. **Product Use Case Name**: Error Handling
Trigger: KnavesNES encounters an unknown instruction, an instruction with an undefined outcome (division by 0, etc.), or any other type of catastrophic error.
Preconditions: KnavesNES is currently executing a ROM file.
Interested Stakeholders: Users and educators.
Actors: CPU, EventListener, ErrorHandler, MemoryLogger

Outcome: KnavesNES will halt executing the ROM file and will echo into the command prompt that an error has occurred, along with a trace for the associated error. Memory will be logged a final time before KnavesNES gracefully ends its process.

7 Inputs

There are two main input sets to KnavesNES. The first is the ROM file which is set to be executed. The ROM file is a compiled list of 6502 OP code instructions compliant to the specifications mentioned above. The second kind of input is commands dealt as arguments through the command prompt, of which include the name of the ROM file to be executed, and the ESC key to halt execution at any time.

8 Outputs

The main output for KnavesNES comes in the form of ASCII formatted .log files which are placed in the users working directory. These .log files are timestamped and contain the memory state of the system, represented as an indexed list of hexadecimal values. The secondary output for KnavesNES is echoes to the command line stating that execution is beginning, ending, and any errors which may occur as well as their traces.

9 Specific Details

KnavesNES requires that it is executed on a stable computer with no previous operation issues. In order to run at the desired clock rate, KnavesNES should be run on a modern computer with at least a 2.0GHz processor. KnavesNES is explicitly secure and accesses no user data apart

from the passed ROM files. Little to no risk exists of data corruption, as KnavesNES writes only to its own directory in the form of memory log files.

KnavesNES can be expected to be maintained and will receive periodic updates to fix bugs and potentially add new features. The life cycle of this product depends on its continued maintenance, however it can be safely assumed it will be maintained at least one year after its initial release.

10 Proof of Concept Demonstration

During the proof of concept demonstration, KnavesNES will demonstrate the execution of a simple ROM file of which the source code will be explained to the audience before hand. Following its completion, the CPUs memory will be investigated in the log file and it will be confirmed that execution was successful. Inherent technical risks during the demonstration include typos in the source code of the ROM file, unidentified bugs in the CPU which could cause an error during execution, and invalid rights on the enclosed folder which could prevent KnavesNES from writing memory log files.

References

[1] *MOS Technology 6502 CPU* - https://en.wikipedia.org/wiki/MOS_Technology_6502

[2] *Nintendo Entertainment System* - https://en.wikipedia.org/wiki/Nintendo_Entertainment_System

[3] *6502 CPU OP Codes* - http://www.6502.org/tutorials/6502opcodes.html

1.0	Oct 9, 2015	Niko Savas created