NIOS II Emulator

Requirements Analysis Document

Team Name
Jake Ediger
Alex Michael
Alex Czarnick
Avinash Nooka
Raksharth Choudhary

Introduction

Purpose

Current environment software for the NIOS II processor has become outdated and received little maintenance throughout its life. Unreliable software accompanied by high costs of boards and chips requires a new solution to be created.

Creating an emulation of the chip and creating an environment for development brings life back to an aging system. Our system is designed to be a one and done solution for testing assembly code for the NIOS II processor without needing a separate board or micro controller. The associated GUI will provide a more efficient and user-friendly interaction.

Scope

The system will emulate the NIOS II processor. It will simulate a 32-bit little endian environment in three simple states; “NOT READY”, “PAUSED”, and “RUNNING.” Memory will be represented as 64 kilobytes block with restriction of only 16 bits for memory addresses. The environment for which the emulator will run will display all 32 register along with several special register. The system will also offer a debugging feature that allows the assembly code to be stepped through and see directly into the registers.

Objectives and Success Criteria

Project success depends on achieving the following set of goals.

- Emulation of a 32 bit processor capable of running all instructions the actual NIOS II can.
- Design of a 64 kilobyte memory block that is addressable on 16 bits.
- Displaying 32 registers along with special registers.
- Design of a processor that exists in three states; running, paused, and not ready.

- Design of a GUI that provides an intuitive user interaction.

Definitions, Acronyms, and Abbreviations

Assembly Code	low-level programming language designed for specific processor implementation
Bit	a unit of information expressed as either a 0 or 1 in binary notation
Debugging	using troubleshooting steps to find errors in programs
Environment	conditions in which the system operates
Processor	a computer chip capable of doing operations of 1s and 0s very quickly
Register	an information holder inside a processor
G.U.I.	Graphical User Interface

References

1) https://www.intel.com/content/www/us/en/programmable/documentation/iga1420498949526.html
2) https://www.draw.io/

Current System

Platform: Web

Language: JavaScript/HTML5/CSS

Proposed System

Overview

The system will be able to emulate the NIOS II processor by running inputed assembly code and showing register values, and current processor state. It will be based on the functional and nonfunctional requirements.

Functional Requirements

F1. Read in text document of assembly instructions and labels by parsing each line and executing them correctly by identifying which instructions, registers, and labels are being used.

F2. Simulate a 32 bit, little endian processor environment with 16 bit memory addresses and 32 general purpose registers.

~~F3. Display current processor state as; "RUNNING," "PAUSED," and "NOT READY."~~

F4. Represent a 64 kilobyte byte-addressable block of memory for specific data accessibility.

F5. Display Program Counter value and contents of the 32 general purpose registers and special registers.

~~F6. Run, Pause, and Reset buttons for controlling the program with the ability to step through and debug assembly code for fixing bugs or examining segments of code.~~

~~F7. User can add, update or delete in memory through the GUI.~~

F8. The user will have the ability to specify up to 16 memory locations to the monitor. For each user specified memory location.~~, the program should display the four contiguous bytes starting at that memory location in hexadecimal.~~

Nonfunctional Requirements

1. Performance
    a. The system should provide an execution time that is indistinguishable from the actual NIOS II chip
    b. The system should show data to the GUI in real-time
2. Usability
    a. The input file format will be clearly described to avoid pre-runtime errors
    b. A functional GUI will display current data from the system
    c. The GUI should be intuitive to use for anybody with experience in NIOS II
    d. Errors that occur will be clearly described to the user to aid debugging
3. Compliance
    a. TBD
4. Development Environment
    a. TBD
5. Stability and Reliability
    a. The system should be able to handle basic errors while providing information on bugs
6. Platform Compatibility
    a. The system will be able to run on multiple operating systems, provided to correct software is installed beforehand
    b. The system will provide an almost identical look across all compatible operating systems

System Models

Scenarios

Runtime Steps	User's Experience
1. Text file with assembly code of length n is fed into the system. 2. The text file in parsed into assembly code. 3. The assembly code is ran through the system. 4. The system sends real time values to the GUI. 5. The system finishes running the assembly code and displays the end values of registers with a completion message	1. User will start program with their desired file. 2. System will display "Parsing" message until parsing is done. 3. While system runs through assembly code, user will be able to see current values. 4. After code is finished,, system will display completion message and the last known values

Use Case Model

Name	ValidateFile
Actor	SYSTEM
Entry Condition:	SYSTEM is not running.
Flow:	1. User inputs text file 2. Text file is parsed and checked for code errors
Exit Condition:	File has not code errors

Name	RunProgram
Actor	USER
Entry Condition:	File is successfully loaded and passes validation
Flow:	1. User presses run 2. File is executed to completion
Exit Condition:	File instructions are run until completion or program is paused/restarted

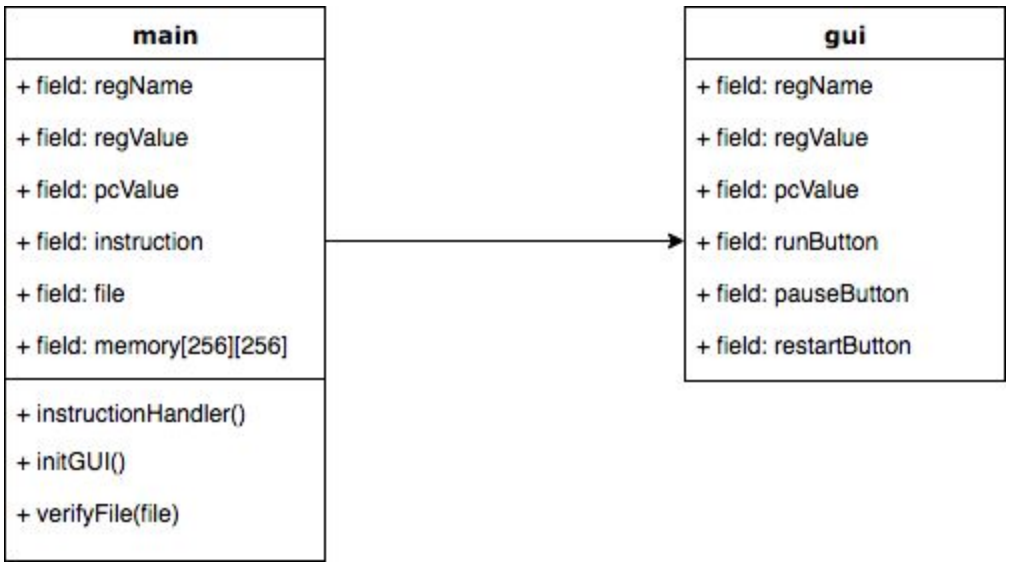
Name	RestartProgram
Actor	USER
Entry Condition:	Program is paused
Flow:	1. Reset program counter to 1 2. Reset Registers 3. Display Registers
Exit Condition:	Program counter is 1 and program is paused waiting to be executed

Name	PauseProgram
Actor	USER/SYSTEM
Entry Condition:	Program is running.
Flow:	1. Pause program 2. Display Registers
Exit Condition:	Program is paused with the registers being displayed.

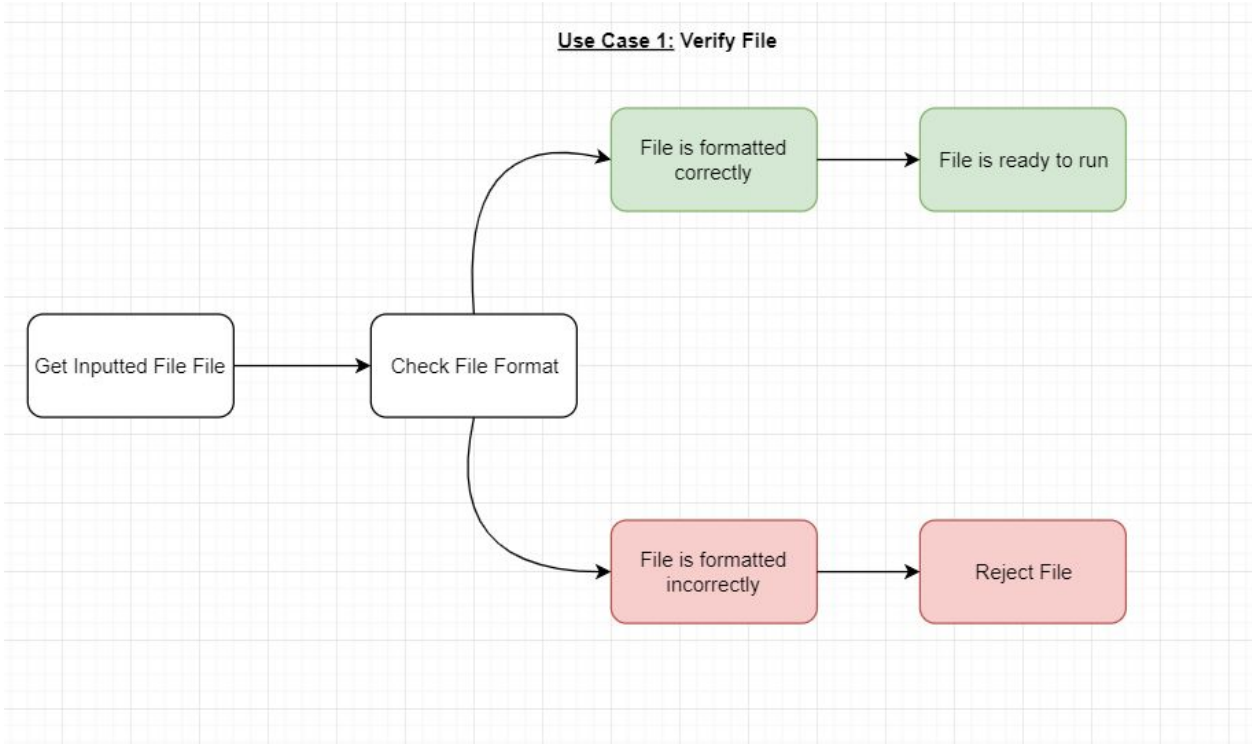
Name	ProgramFinished
Actor	SYSTEM
Entry Condition	Program has executed all lines
Flow	1. Display Registers
Exit Condition	Program is paused with the registers being displayed

Name	ReUploadFile
Actor	SYSTEM
Entry Condition	File has been ran and USER wants to upload new file
Flow	1. Reset Registers 2. ValidateFile
Exit Condition	New file is valid. Program is Paused

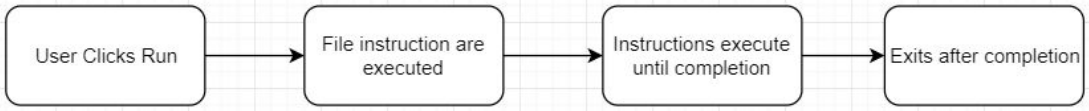
Oject Model



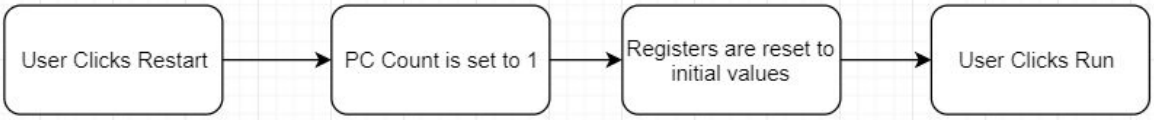
Dynamic Model



Use Case 2: Run File

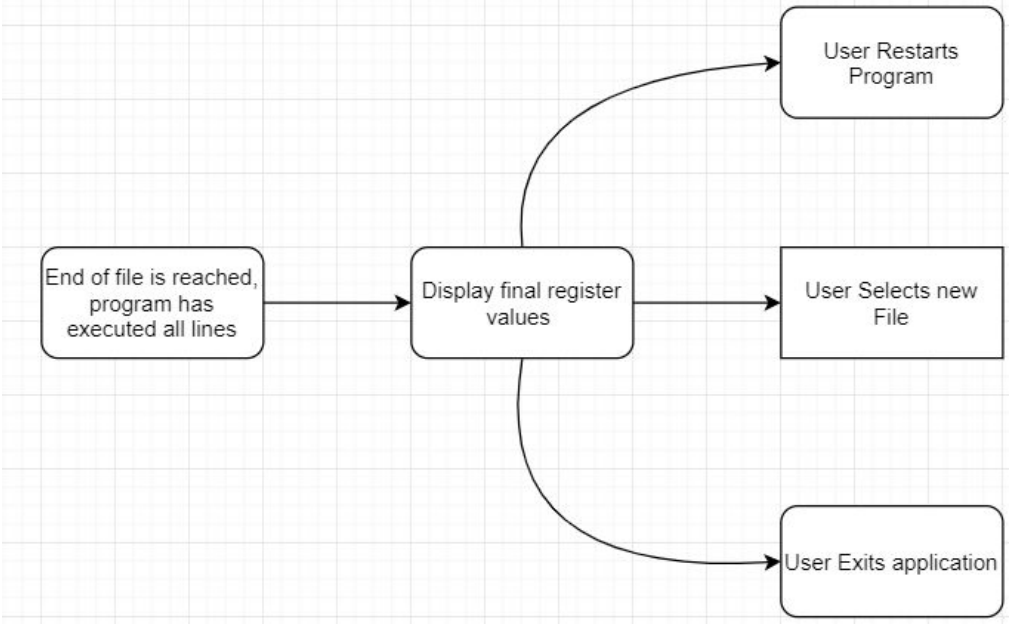


Use Case 3: Restart Program

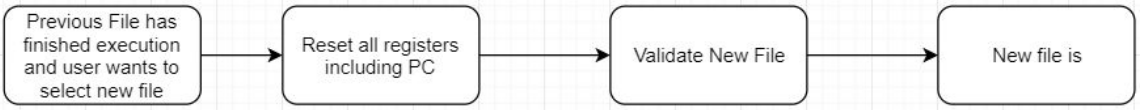


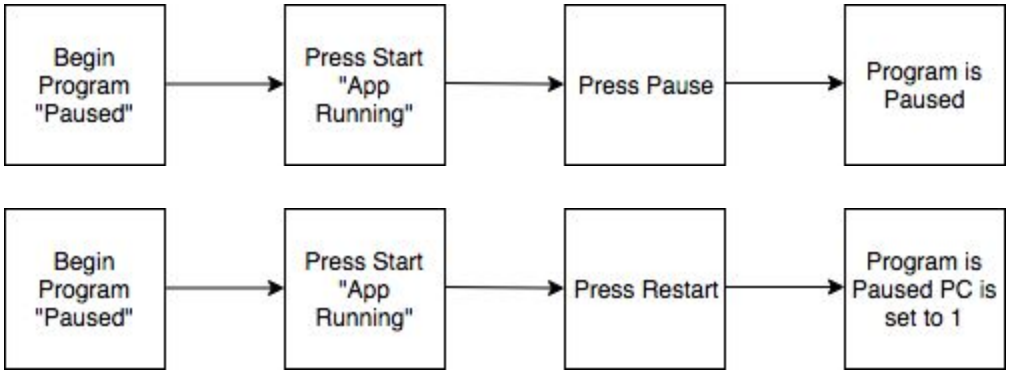
Use Case 4: Pause Program

Use Case 5: End Of Program



Use Case 6: Reupload File





User Interface: Navigational Paths and Screen Mockups

Glossary

- **Little Endian:** A method of organizing bytes with words where the least significant byte is placed at the lowest address in memory
- **NIOS II:** The NIOS II is a processor architecture with software currently provided by Intel. It is primarily for use in FPGAs, and has been in use for quite a few years.
- **Hexadecimal:** Hexadecimal is a number system with a base of 16, rather than decimal base 10. Hexadecimal notation aids in making binary information more human readable. Hexadecimal numbers are customarily denoted with the prefix “0x”
- **Byte:** Bytes are made up of 8 binary “bits,” and are used to manipulate and store information.
- **Emulation:** An emulation is a mimicking of a system to produce the same functions and results on a different system.