

# NIOS II EMULATOR

## REQUIREMENTS ANALYSIS DOCUMENT

Team Name  
Jake Ediger  
Alex Michael  
Alex Czarnick  
Avinash Nooka

### 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Current environment software for the NIOS II processor has become outdated and received little maintenance throughout its life.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

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

## REFERENCES

- 1) <https://www.intel.com/content/www/us/en/programmable/documentation/iga1420498949526.html>

## CURRENT SYSTEM

TBD

## PROPOSED SYSTEM

### OVERVIEW

The system will be able to emulate the NIOS II processor by running inputted 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.

F9. A functional GUI will be available to the user to see current data, and execute commands specified in F7.

## 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. The GUI should be intuitive to use for anybody with experience in NIOS II
  - c. 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
<ol style="list-style-type: none"><li>1. Text file with assembly code of length n is fed into the system.</li><li>2. The text file is parsed into assembly code.</li><li>3. The assembly code is ran through the system.</li><li>4. The system sends real time values to the GUI.</li><li>5. The system finishes running the assembly code and displays the end values of registers with a completion message</li></ol>	<ol style="list-style-type: none"><li>1. User will start program with their desired file.</li><li>2. System will display "Parsing" message until parsing is done.</li><li>3. While system runs through assembly code, user will be able to see current values.</li><li>4. After code is finished,, system will display completion message and the last known values</li></ol>

---

## USE CASE MODEL

Entry Condition:	Processor is NOT READY
Flow:	<ol style="list-style-type: none"><li>1. User inputs text file<ol style="list-style-type: none"><li>a. Text file is parsed and checked for code errors</li></ol></li></ol>
Exit Condition:	Processor is PAUSED

Entry Condition:	Processor is PAUSED
Flow:	<ol style="list-style-type: none"><li>1. User presses run</li></ol>
Exit Condition:	Processor is RUNNING

Entry Condition:	Processor is RUNNING
Flow:	<ol style="list-style-type: none"><li>1. System executes BREAK</li><li>2. DISPLAY registers</li></ol> OR <ol style="list-style-type: none"><li>1. User presses pause</li><li>2. DISPLAY registers</li></ol>
Exit Condition:	Processor is PAUSED

Entry Condition:	Processor is RUNNING
Flow:	<ol style="list-style-type: none"><li>1. System executes code</li><li>2. DISPLAY registers</li></ol>
Exit Condition:	Processor is RUNNING

---

## OBJECT MODEL

TBD

---

## DYNAMIC MODEL

TBD

TBD

## 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.