CISC360 - Project 1

James Feister - jfeister@udel.edu

Introduction

Project 1 required an implementation of Prof Morecrafts' Gemini computer architecture. The project requirements were as follows.

- GUI: Showing registers and instruction executing
- Loading and Parsing of program for syntax errors: Alert user of failure
- Running of program instructions, non bytecode translation
- Detection of memory access errors, Alert user of failure
- Generate the baseline of the project.
- Translate a c code file into gemini asm syntax

Background

For this project I took the opportunity to revisit and dust off my c++ skill as it has been close to 5 years in doing so. We were asked to do this in C# for real world business experience in a enterprise level language. Granted the permission to deviate and understanding the teachers concerns I was originally going to do this with the GTK toolkit but decided to do an evaluation of the windowing toolkit landscape. I came across the Qt toolkit and compared to GTK It is a more "Professional" level cross platform windowing toolkit that will work on Windows, OS X, X11, android and other various platforms. Given this type of support and backing by bigger enterprise entities Qt seems the better choice and is what I decided on.

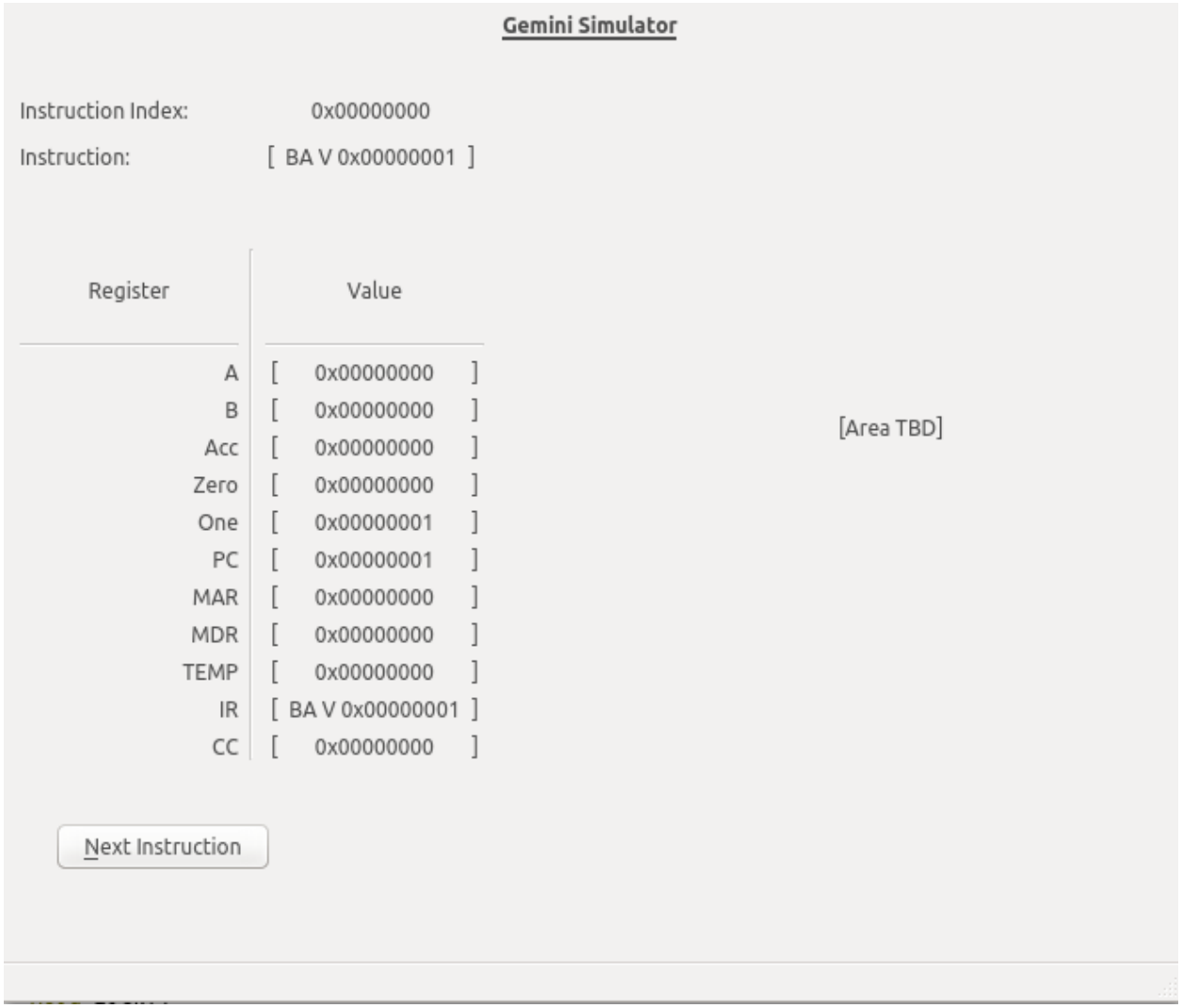
GUI

This shows the gui in its entirety.

Complete GUI

Loading Files

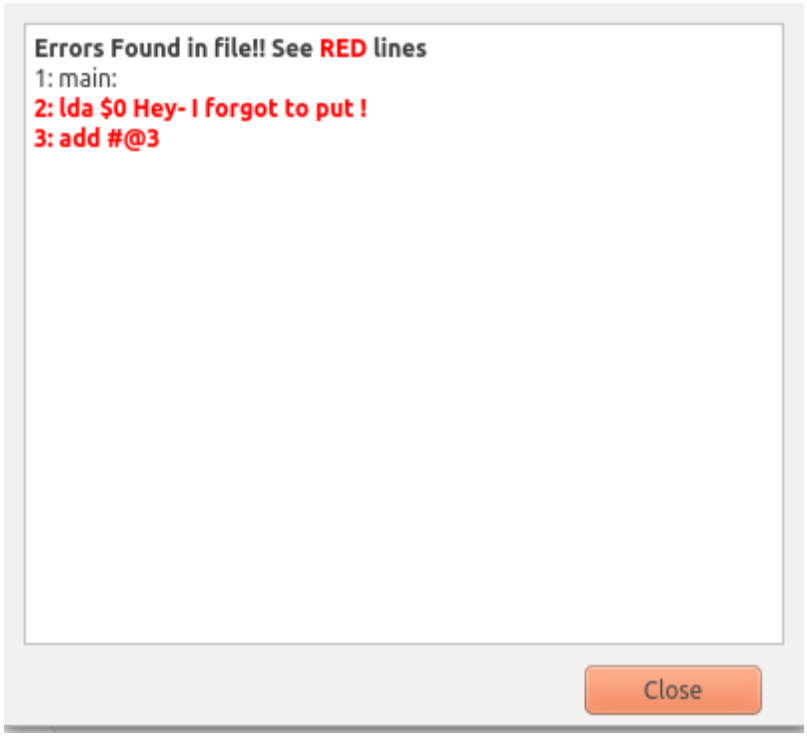
This is the gui showing a file loaded. Notice it is ready to jump to the main label and the “Next Instruction” button is enabled.



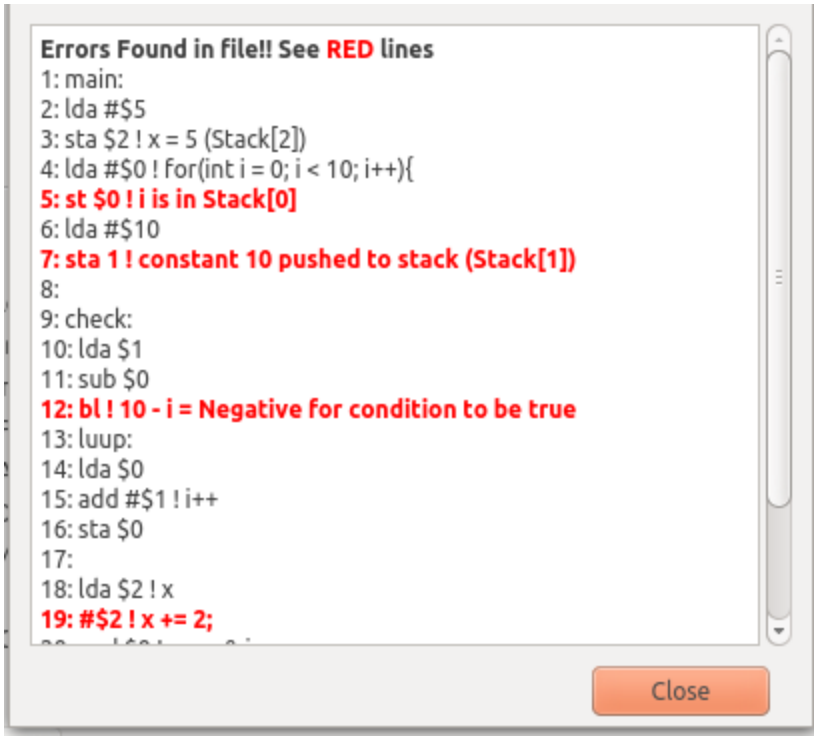
Loaded file ready to execute, will start with a branch to main.

Verifying gemini asm file

This is a shot of the gui showing errors in the source files provided, the original test2.s and a modified test3.s to emphasis the error highlighting. This is only during parsing and not runtime.



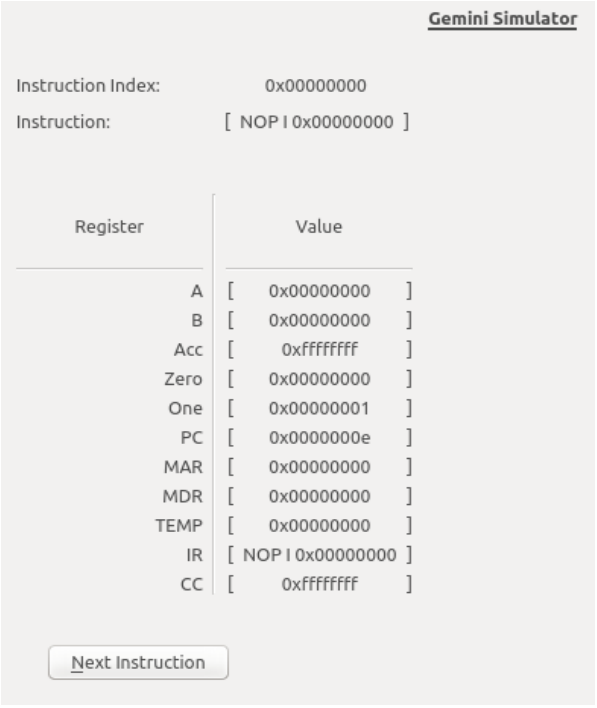
Example 1. parsing



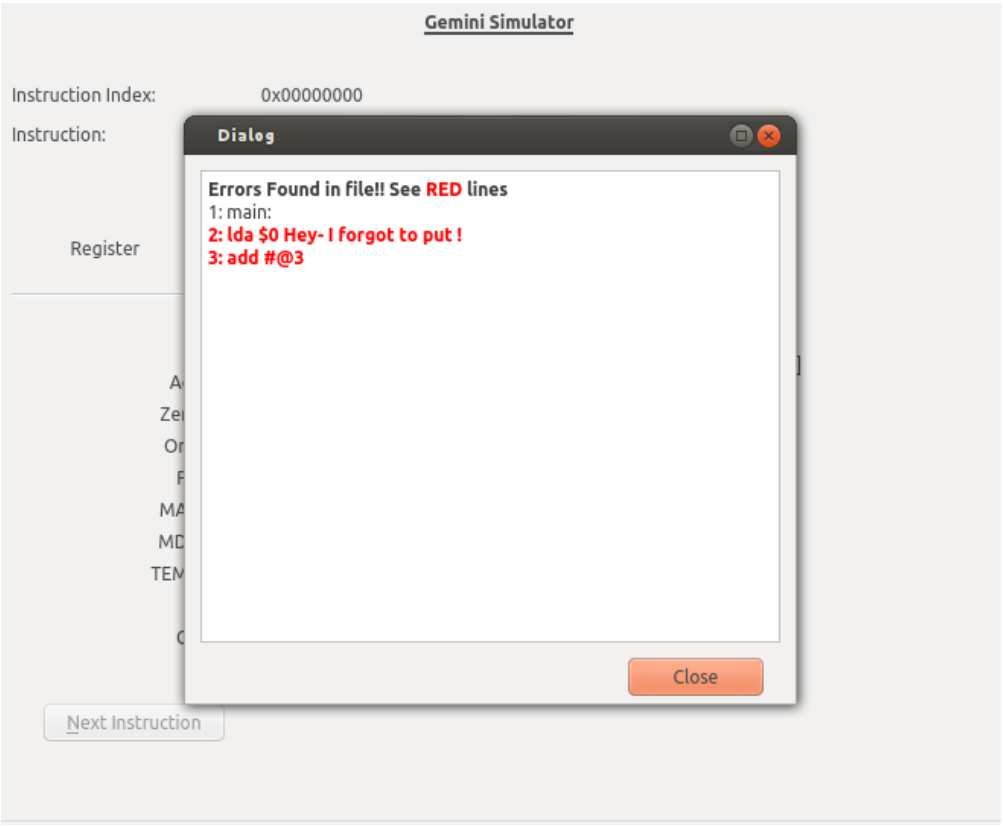
Example 2. parsing

Running applications

This show the results of the 5 running applications test1.s test2.s test3.s test4.s test5.s

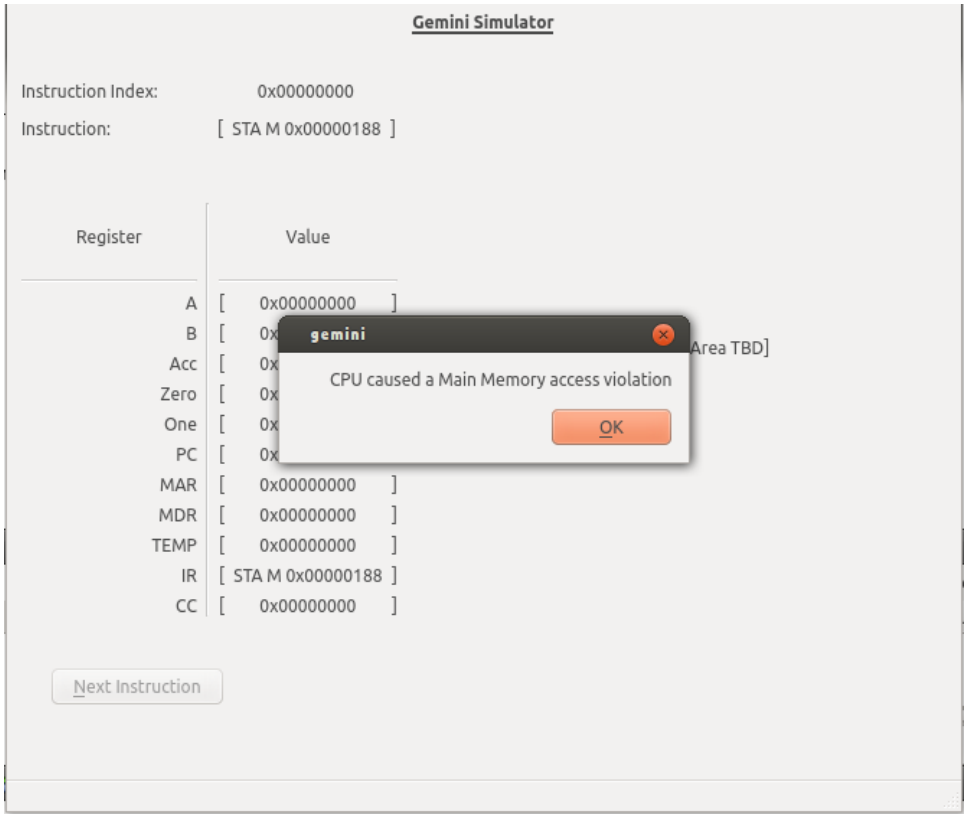


test1.s

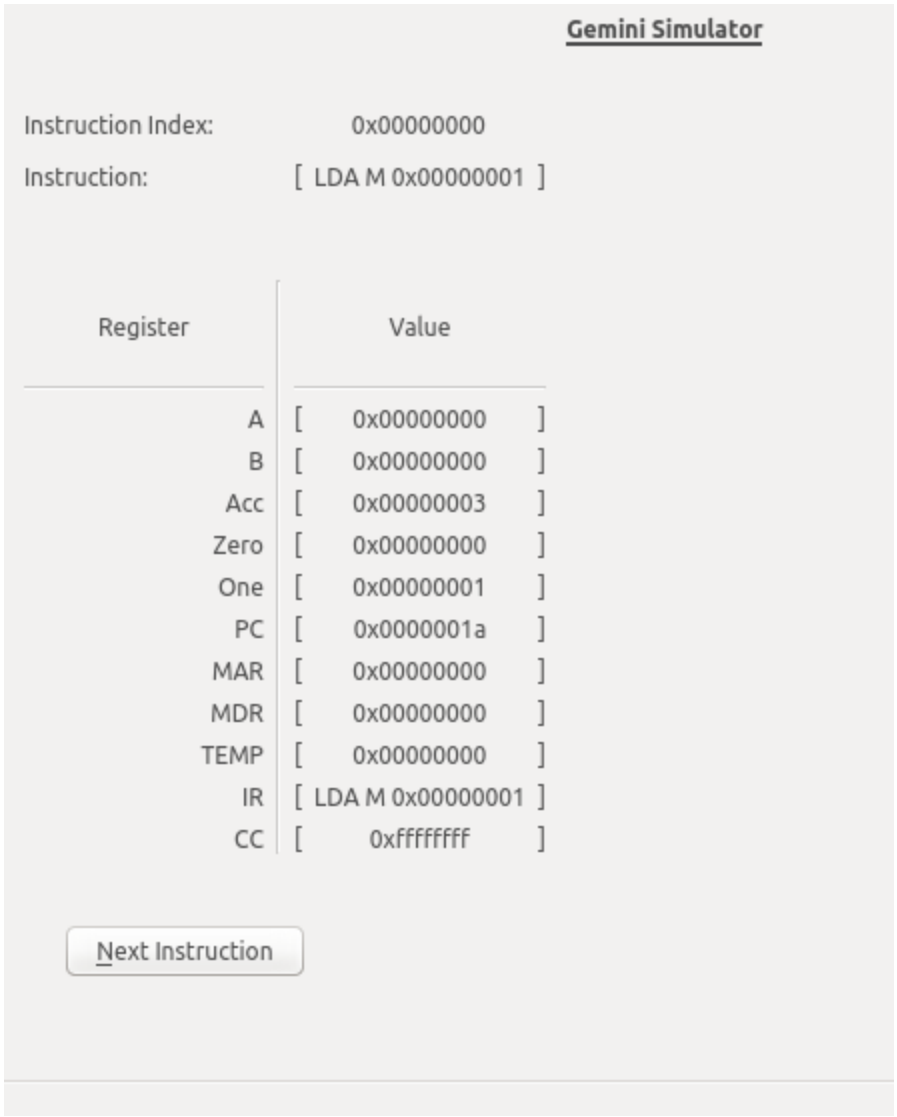


test2.s

test3.s



test4.s


Gemini Simulator
Instruction Index: 0x00000000
Instruction: [LDA M 0x00000001]
Register
Value
A [0x00000000]
B [0x00000000]
Acc [0x00000003]
Zero [0x00000000]
One [0x00000001]
PC [0x0000001a]
MAR [0x00000000]
MDR [0x00000000]
TEMP [0x00000000]
IR [LDA M 0x00000001]
CC [0xffffffff]
Next Instruction

test5.s

Conclusion

Was able to perform the parsing of the gemini asm instruction and return a pseudo byte code to pass to the cpu for execution. Generate an error popup for multiple errors in the parsed byte code. The cpu is executing the pseudo bytecode on a per user click/clock cycle. Should a memory access out of bounds occur an error dialog box is thrown leaving the cpu in a stalled state. And I believe that all programs execute correctly.

Overall this was a very challenging exercise. To get back into programming and developing a gui application. The hardest part was dealing with the editor, the constant mouse grab to keyboard edits make development very slow.