Component Based MMIX Simulator using Multiple Programming Paradigms

Stephen Edmans

Contents

1 Introduction 4

2 Existing Implementations 4

3 Outline Design 4
- 3.1 Assembler . . . 6
- 3.2 Virtual Machine . . . 6
- 3.3 Graphical User Interface . . . 6
  - 3.3.1 Programming Paradigm . . . 6
  - 3.3.2 Initial Design . . . 6
  - 3.3.3 Memory . . . 6
  - 3.3.4 Registers . . . 6
  - 3.3.5 Main Program State . . . 6
  - 3.3.6 Standard Console . . . 6
- 3.4 Simulation Controls . . . 7

4 Development Methodology 7

5 Development Plan 7

6 Summary 7

7 References 7

List of Figures

1 Component Interactions 5
2 Graphical User Interface 7

1 Introduction

As software systems get larger and more complex there is a need to handle this complexity. There is a prevailing design paradigm, which addresses these issues, that is to break these systems up into smaller components These components will interact with each other to make the complete system. When you have control over the development of more than one of these components it is traditional to use a single programming paradigm for your components. There is, however, no reason that you cannot use different languages and paradigms for these components. The goal of this project is to create a relatively complex system that is made up of multiple components where each component uses the most appropriate programming paradigm for each component.

The system that I plan to create in this project is inspired by Jeliot, which is a tool that is used as an aid in the teaching of Java. The Jeliot system allows a user to give it a piece of Java source code and it will show the user what the underlying virtual machine is doing when it runs the code.

In his seminal work The Art of Computer Programming Dr. Donald Knuth designed an artificial machine language that he called MIX. In a later volume of his work Dr. Knuth updated this machine language, which he calls MMIX. This project will create a system that take MMIX assembly code and shows the user, graphically, what the simulated machine is doing.

2 Existing Implementations

There is already one existing main implementation created by Dr. Knuth and his team at Stanford University. This implementation runs as a few command line utilities. These utilities include an assembler which converts a text file containing the relevant assembly language source code into a proprietary binary format. There is a utility which reads files in this proprietary binary format and displays the content back to the console. There is also the main utility which will execute the application stored in the binary files using an MMIX virtual machine. It is possible to get the virtual machine to output some tracing information but it does not have a graphical user interface and it does not allow you to step through the processing of the application.

These utilities are all written in CWEB

3 Outline Design

The MMIX simulator application will consists of three separate components. The task of converting the MMIX assembly language source text into the corresponding binary representation will be performed by a component I am calling the Assembler. The task of actually simulating an MMIX computer will be performed by a Virtual Machine component. The final component will be responsible for representing the current state of the MMIX computer to the user, along with orchestrating all of the interactions with

the other components. I am calling this component the User Interface. The interactions between the components can be illustrated by figure 1.

One question that needs to be answered is what is the target platform for this application. The application will initially be written for Mac OS X but if there is time available at the end of the development process I would like to make sure this application will run on other platforms.

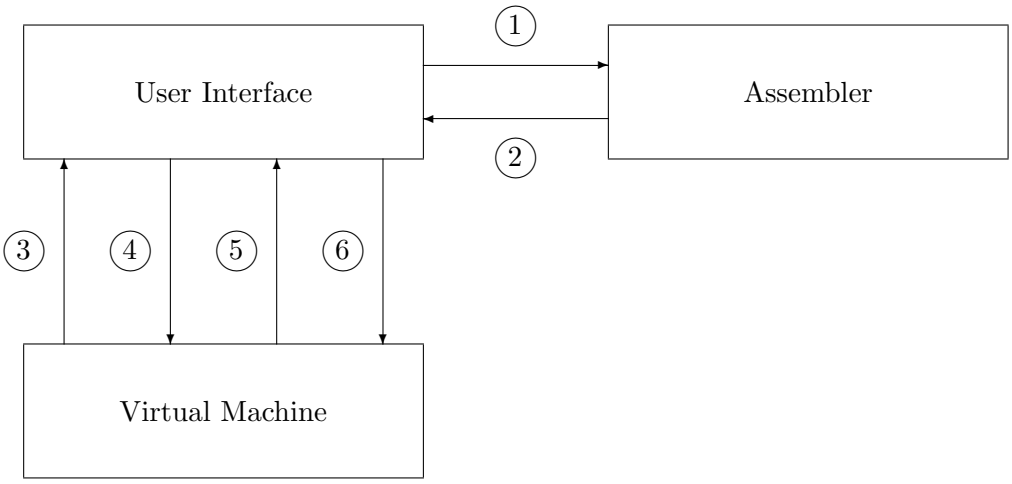

Figure 1: Component Interactions

1. Source Text
2. Binary Representation
3. Binary Representation
4. Current State
5. Process Next Step
6. Change of State

3.1 Assembler

The purpose of the assembler component is that it will take in MMIX assembly language source code contained in a source file and convert it into a binary representation of the application. The first thing that I need to decide upon, for each component, is what is the most appropriate programming paradigm. The production of an assembler is a fairly well defined process. There should be little, or preferably, no ambiguity in the translations.

3.2 Virtual Machine

Reset Simulator Load Program Reset Program Process Next Statement

3.3 Graphical User Interface

The main way that users will interact with the simulator is through a graphical user interface (GUI). The GUI will be responsible for all of the interactions with the other components. There are three main forms of user interface commonly used at the present time. The difference between the three are how the user access the application and where the processing occurs. The traditional method is an application that is installed on the client machine and runs directly on the machine. The newer approach

3.3.1 Programming Paradigm

3.3.2 Initial Design

The GUI can be broken up into 4 separate sections as illustrated in figure 2

3.3.3 Memory

The memory section of the GUI will contain a representation of the simulators current memory.

00000000: 8fff 0100 0000 0701 f4ff 0003 0000 0702 ..... 00000010: 0000 0000 2c20 776f 726c 640a 00 ...., world..

3.3.4 Registers

The registers section of the GUI will display a list of all of the available registers in the simulated MMIX machine. It will show not only the names of the registers but there current values.

3.3.5 Main Program State

3.3.6 Standard Console

There are a number of embedded computers that do not interact directly with a user however it is quite rare for a general purpose computer to have no interactions. The

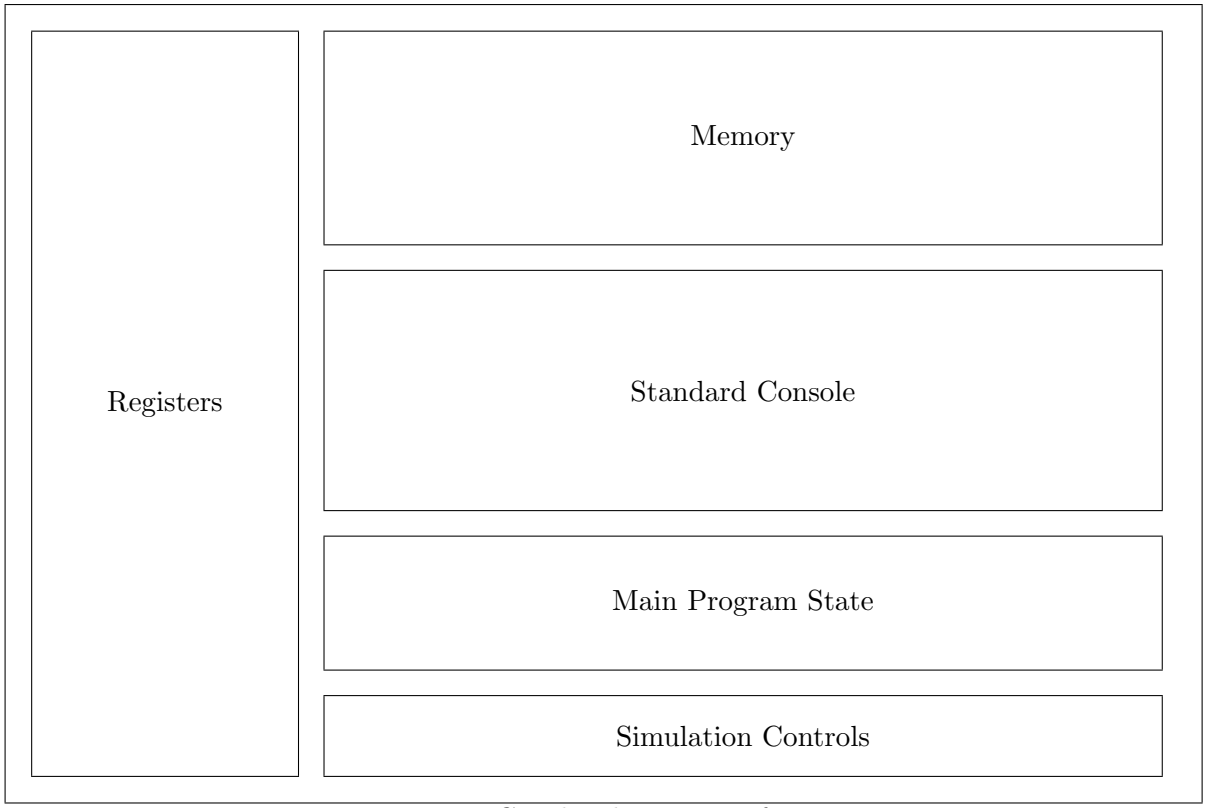

Figure 2: Graphical User Interface

MMIX simulator have got standard input and output channels. These are accessed through the standard console area on the GUI.

3.4 Simulation Controls

4 Development Methodology

5 Development Plan

6 Summary

7 References