Cosmic
A Software Simulated 8-Bit Computer

Clay Buxton
Computer Engineering, Computer Science
Elizabethtown College
Elizabethtown, PA
buxtonc@etown.edu

Kevin Carman
Computer Engineering, Computer Science
Elizabethtown College
Elizabethtown, PA
carmank@etown.edu

***Abstract*—Cosmic is a simulated 8-bit microcomputer architecture , designed to resemble retro computers. Cosmic's goal is to simulate, completely in software but with hardware design in mind, a microprocessor from the early 1980s.**

***Index Terms*—retro-computing, simulation, emulation, microprocessor, architecture.**

I. Introduction

Cosmic can be broken down into four main parts:

- **Cosmic Processor** - The simulated CPU used to drive the rest of the system. The Cosmic CPU design is comparable to the Zilog Z80 and the MOS 6502.
- **Cosmic System** - Everything that is found on the main board of a similar physical machine such as memory, input & output, graphics, and audio. This also will fit in with appropriate machines of the time like the Apple II. Our primary goals are things like RAM, ROM, keyboard, and graphics while our stretch goals are I/O, audio, and expansion peripherals.
- **Cosmic Software** - Software that will run on the cosmic system. Base goals are a simple kernel, with appropriate drivers, some form of command interpreter, basic utilities, and a simple game or two. Our stretch goals are a BASIC interpreter, a library that reflects coreutils, and a port of a popular game of the time.
- **Auxiliary Software** - Software written to run on a modern computer. This will consist of the simulation environment along with an assembler and user-friendly panel.

The project will be deemed "acceptable" once the processor and auxiliary software are complete, and the base goals of the system and software portions have been met. Once finished, the simulated system will operate similarly to a computer of the early 1980s, primarily the Apple II.

The design philosophy of the computer is to create a hardware-like machine. The software will be engineered in a way that will reflect the hardware it is based around, without having to deal with the nuances of a physical machine. Features like chip pin-out, buses, and a physical board will be taken into account while things like timing will be disregarded.

Neither team member is taking the direct lead on any specific part of the project at the moment, though we will work on things independently as the project progresses and there are more tasks available. As a group of two with a project that has many facets, there should be very little overlap.

II. Background

The largest source of background information is in the machines that Cosmic is designed to resemble. As previously mentioned, the two primary reference devices are the Apple II and the TRS-80 containing the MOS 6502 and the Zilog Z80, respectively.

There are a few other projects out there that are similar to what we are doing. Emulators of similar devices will be helpful for design inspiration. Unlike emulators, we will not have the same design constraints which adds a sense of freedom when creating the system. Outside of emulators, there are a few projects where developers have created self-defined simulated platforms. One such is project is an 8-bit Assembler and Simulator written by Marco "Schweigi" Schweighauser [**?**]. This project is similar to our first step of creating a processor and an assembler for the Cosmic Processor. Another project that influenced the creation of Cosmic is David Murray's Commander X16 project [**?**]. Unlike the Assembler-Simulator, this is a "home-brew" physical computer made using off-the-shelf parts. This project uses the 6502 and other components to create a retro feeling machine with more modern parts.

Luckily, devices of the time have been very well documented in the past 40 years. This documentation makes reverse-engineering the systems and processors simple and allows us to figure out how engineers of yesterday solved problems that we may encounter while designing our system.

References

[1] Schweighauser, M. (2019). Schweigi/assembler-simulator. [online] GitHub. Available at: https://github.com/Schweigi/assembler-simulator [Accessed 7 Sep. 2019].

[2] GitHub. (2019). Commander X16. [online] Available at: https://github.com/commanderx16 [Accessed 13 Sep. 2019].