Cosmic
Preliminary Design Report

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

I. Design Constraints \Problem Definition

Cosmic is a fully simulated 8-bit microcomputer architecture. We wanted to create a system where everything was built from the ground up. It is a true "homebrew" computer in software. While many similar devices and programs exist, they all either use off the shelf components or are very simple and underdeveloped. These types of projects hold a lot of interest in retro-computing groups because they are simple devices with easy to understand code which teaches low-level computing concepts.

Our design is rather simple. Using C++, Cosmic is broken down into the same way a hardware design would be. Each piece of hardware has its own separate class. The CPU, RAM, Graphics, and I/O that would all be different chips on a physical machine are treated the same way in the software. This setup allows for the same modularity that the hardware would provide. Using callbacks and memory pointers, we can have a valid data and address bus that connects to all of the appropriate chips with a mother class that acts similarly to the motherboard of a physical machine, routing data to where it needs to go.

To use the Cosmic system, a GUI is being implemented using ImGUI, an easy to use, and flexible GUI framework. Using this, everything going on inside of Cosmic is viewable to the user. Memory, registers, ROM, and processor states are all viewable at any given time for analysis, which allows the user to change data during run time and watch the results.

Our primary design constraint focuses on limiting the number of instructions we include in our RISC-like instruction set. Only a few of our instructions can be considered complicated giving it a RISC-like set. Our goal is to have no more than 50 assembler mnemonics to keep things simple, yet usable. The machine language is comparable to the Zilog Z80 and MOS 6502 with small inspirations from both of them. Generally, emulators have to deal with the timing of the chip to appropriately interact with other devices and keep a cycle count. Since Cosmic is done entirely in software and has no regards to a physical machine, timing is not a constraint and each instruction is assumed to be one cycle.

II. Timeline \Schedule

A. Fall Semester - Designing the Cosmic System

By the end of the fall semester, we plan to have the processor and underlying system (memory, graphics, etc.) completed. The simulation environment will be at a point where it can adequately support the system and run programs written in machine code. Also, we want to start writing an Assembler to make the targets of the spring semester much more comfortable to finish.

B. Spring Semester - Writing Software for Cosmic

With the system itself finished in the fall semester, the spring semester will be mostly dedicated to writing software to run on the system. The first goal is to write an assembler, so all subsequent work is much easier. Once the assembler is in a usable state, work will begin on the kernel and essential software to go along with it.

Along the way, we will also be doing vigorous testing to make sure that all of the preceding components work flawlessly.

III. Budget

Our current budget is $0. Since our entire project is in software, nothing needs to be bought. Later on, if time allows, we aspire to add interfacing with the GPIO pins of a Raspberry Pi, but all the required hardware is readily available in the Computer Engineering Lab already.

IV. Social, Ethical, and Environmental Impacts

A. Social Impacts

Our project doesn't have any direct social impacts. It's a project that many people will find interesting and that could be useful in an educational environment since it shows everything that is going on.

B. Ethical Impacts

The biggest ethical factor of our project is about the code itself. We decided to keep the project 100% open source. We think this is important for a project like this to allow others to learn from our design. All of the libraries included are also all open source.

C. Environmental Impacts

Our project will have no impact on the environment, as it will be written entirely in software. If time allows for us to interface with the Raspberry Pis, all of the hardware used will be completely reusable afterwards.