IBM 650 Emulator in Scheme Design
Topic Paper #10

Alex Laird
CS-3210-01
2/18/09
Survey of Programming Languages
Spring 2009
Computer Science, (319) 360-8771
alexdlaird@cedarville.edu

Grading Rubric		Max	Earn
	On Time/Format	1	
	Correct	5	
	Clear	2	
	Concise	2	
	Total	10 pts	

Peer Reviewer

ABSTRACT
This paper talks about my intended implementation design for an IBM 650 emulator written in the Scheme programming language.

Keywords
IBM 650, Emulator, Scheme, Design

1. INTRODUCTION
The Scheme emulator promises to be much more difficult to wrap your mind around and to implement; Scheme is not an object-oriented language as Python is; its premise is functional programming with minimalistic tendencies. Not only does this make it difficult to read, it makes it difficult to write. The emulator will be far more difficult to implement using this programming language.

2. READING AND PARSING THE FILE
Due to its functional nature, the commands may need to be read executed as they are read from the file; this may be the simplest approach. The file will be parsed in the same manner it was in the Python emulator.

3. EXECUTION OF STATEMENTS
The execution of statements will be identical to that of the Python emulator, simply implemented in Scheme's syntax instead.

4. INPUT AND OUTPUT
The input and output from the program, as in the Python emulator, will be either read and written from and to a file or from and to the shell.

5. POTENTIAL PROBLEMS AND TESTING
In the Python emulator, we simply defined a message object and used that to store our information; variables were easily accessible in all situations. The form of the Scheme programming language presents this as a very different problem and could make it very difficult.

For testing, the results from the output file will be compared to the results from the Python emulator and those results which are known to be accurate (namely, those calculated manually by myself).

6. CONCLUSIONS
This IBM 650 will be much more time consuming and difficult to implement, but it should be possible to do. It will require a lot of testing and designing before implementation, however.