**Meeting Minutes**

**Date:** January 21, 2014

**Start Time:** 5:00pm

**End Time:** 6:00pm

**Members Present:** Drew Aaron, Michael Beaver, Clay Boren,

Chad Farley, Andrew Hamilton, and Travis Hunt

**Members Absent:** N/A

**Topics** **Discussed**

* High-level questions for the client
* Backend modeling

**Decisions and Actions Taken**

The team formulated high-level questions to ask the client during the first meeting. Chad modeled the backend system. He also described (at a high level) the function of the backend parser and definition modules. See the attached diagram and description. The team brainstormed names for the system.

**Supplementary Information**

**Questions to ask the client:**

1. How much of the instruction set is to be included? Is it to be the entire set or a subset?
2. Which specific features of MARIE are required or desired?
3. What are your feelings concerning the use of open source software in the implementation?
4. What file extensions do you want to use?
5. What file extensions need to be allowed for I/O operations (e.g., for opening source files)?
6. Do you have any system name requirements (i.e., client-desired name)?
7. Do you want an “assemble” option which just assembles the source instead of just a “run” command that automatically runs the program?
8. What about the “Final Run” option in the original ASSIST/I?
9. Which ASSIST/I options are necessary?
10. Are there any special features (not native to ASSIST/I) you would like, such as large block comments?

**Potential system names:**

ASSIST/I01

COBRA (Coding On Broken, Rundown ASSIST/I)

KOBRA

ASSIST/II

**Chad’s Mockup Model of the Assembler Backend Portion**

**Backend**

**Parser**

**Definitions**

**Exception**

**Handling**

(Handshake)(

**Chad’s Algorithmic Description of the Assembler Backend Portions**

Define

1. Maintain a listing of available functions and attributes (spacing, case sensitivity) of ASSIST/I.

2. Provide appropriate feedback for given input. Go to Parsing Step 3.

Parsing

1. Read “Tokens” from input. A “Token” is a recognized “word” or “variable” in the language.

2. Check Token with recognized symbol. Go to Define Step 1.

3. Update the symbol table. Go to Parsing Step 1.