**Meeting Minutes**

**Date:** January 31, 2014

**Start Time:** 4:30pm

**End Time:** 7:15pm

**Members Present:** Drew Aaron, Michael Beaver, Chad Farley,

Andrew Hamilton, and Travis Hunt

**Members Absent:** Clay Boren

**Topics** **Discussed**

* Backend high-level organization
* Client questions #3
* Frontend high-level features
* File organizations
* Team Foundation Server and GitHub

**Decisions and Actions Taken**

Chad developed a temporary high-level sketch of the different backend components and their interactions. At this time, the sketch is not final and is purely a visual aid. The goal is to modularize the backend as much as possible to make development easier. A mockup of the sketch is attached.

The team discussed and developed questions to ask the client during the third client meeting. The questions are attached.

The team began collecting and discussing the high-level features of the user interface. The user interface will have a system of menus on a menu bar. The user interface will also have a text editor, a register display, a memory display, a debug interface, and a system for displaying output to the user. There will also be other miscellaneous features. The team will develop a prototype at the next meeting. If the prototype is ready, it will be presented for critique.

The team also discussed the file organization of user project files and source code files. The user interface will have export and import options, in addition to standard open and save options. The “export” option will save source code to a normal text file. The “import” option will load source code from a normal text file. The “save” option will save a project file and a source file. The “open” option will open a project file and its associated source file. The project file will save the state of the ASSIST/I options and the current source file’s name. The basic organizations of the project and source files are attached.

Chad was originally searching for a file-locking system for GitHub. However, it appears that the system is nowhere to be found. Visual Studio Team Foundation Server (TFS) does have this mutual exclusion feature, and the team will be using TFS to check-out, update, and check-in code. Incremental updates will be uploaded to GitHub from the code on the TFS system as a redundant backup.

The next team meeting will be Monday, February 3, 2014 at 6:00pm. The location of the meeting is tentatively set to be the Collier Library basement. If possible, however, the team will reconvene at the Christian Student Center.

**Supplementary Information**

Frontend

“Linker”

Initialize

Symbol Table

Traditional

Symbol Table

Operations

List

Parser

Library

Definitions

Generate

.PRT

**Chad’s Backend Mockup:**

Step 1

Step 2

Step 3

**Questions to ask the client:**

1. Do you want assemble-time errors to be handled as error detection or as error avoidance?

2. Should error detection be implemented as ASSIST/I implements it?

3. Suppose a user wishes to modify the maximum size of available memory. What is the upper bound on memory size you would like? ASSIST/I bounds the memory to 9999 bytes.

4. Do you want to support the use of the asterisk to refer to the location counter?

5. The IBM /360 has a few floating-point registers. Would you like us to emulate these as well?

6. ASSIST/I is a two-pass assembler. Would you like to follow this standard?

7. The ASSIST/I .PRT files display and print better in landscape mode. Would you prefer .PRT files to be displayed in landscape mode, portrait mode, or to have options for both?

8. Would you like the ASSIST/I assembler options to be project-specific or global?

9. As a follow-up to #8, would you like the ability to save project settings, options settings, and other relevant project information to a project file (e.g., myProject.una)?

10. As a follow-up to #9, what are your thoughts on the extension \*.una for project files?

11. As a follow-up to #8, if you prefer global options, would you like them to persist between sessions (i.e., options settings saved to a configuration file)?

12. What extension would you like for the source file? Perhaps the files could use the .TXT or the .UAS (UNA Assembly Source) extension?

13. Would you like the user to be notified in the event of a run-time error? If so, then how?

14. Memory dumps from crashes are to be saved to the .PRT file. How would you like memory dumps from XDUMP to be displayed to the user?

15. In ASSIST/I, the debugging (“Run”) mode saves a .PRT file after execution. Would you like the emulator’s “Assemble and Debug” run option to save a .PRT file or not?

16. The project description lists the option of “exiting from a file without saving the file.” Would you like the user to be prompted to save the file before exiting, like in modern IDEs?

17. In the project description there are two features “Delete line(s)” and “Insert lines(s).” Are these necessary, separate actions, or could these not be achieved simply by using backspace / delete and the return key? If these are necessary actions, could you elaborate on how you envision them functioning?

18. Do you want the emulated assembler to track and report warnings and errors? If so, should the emulator be able to handle *all* possible ASSIST/I warnings and errors or a relevant subset?

19. Would you like the option to print .PRT files from the IDE?

20. Could you elaborate on the “configuration of assembler (specify path and arguments)” requirement?

**High-Level User Interface Features:**

* Text Editor
* Register Display
* Memory Display
* Output to User
* Debug Interface
* Files
  + .PRT file
  + .UNA project file
  + .TXT or .UAS source file (extension to be determined)
* Menu Bar
  + File Menu
    - New
    - Open
    - Save
    - Save As
    - Import
    - Export
    - Print
    - Exit
  + Edit Menu
    - Cut
    - Copy
    - Paste
    - Search
    - Search and Replace
  + Assemble Menu
    - Assemble
    - Assemble and Debug
    - Assemble / Final Run
  + Tools Menu
    - View .PRT File
    - Options
  + Help Menu
    - Online Help
    - About
* Toolbar
  + New
  + Open
  + Save
  + Print
  + Assemble
    - Assemble and Debug
    - Assemble / Final Run
  + View .PRT

**Sample Project File (.UNA) Contents:**

myProj.una

[Options]

maxMemory=9999

//

// Lines preceded by double-slashes are comments

//

[Source]

proj=myProject.uas