Humane Assembly Language Tools
Software Specification

Matthew Bennett, Michael Erwin
School of Computing. University of Southern Mississippi. Hattiesburg, MS 39406
matthew.bennett@usm.edu, michael.erwin@usm.edu

Statement of Purpose

The Humane Assembly Language Tools (HALT) project seeks to provide a simple and efficient development and execution environment for the transition from higher level languages.

HALT is a toolkit for user-friendly development and inspection of Motorola 68000 assembly language. The philosophy behind HALT is to make assembly language as accessible as possible to a broad audience of programmers. That philosophy is realized through a simple, colorful human interface, connecting the user to advanced tools such as a powerful lexxer/parser, a bare M68000 machine language interpreter, an M68000 assembler and translator, and various debugging and execution tools.

HALT provides a simplified run-time visualization for the internal working environment of a virtual M68000 machine. The visualization updates the state of the stack as new code is typed into the working project. The simulator is also a visualization environment for program execution, and displays the contents of registers and memory as it occurs once the assembly instructions have been successfully translated and interpreted. All this is done at development time, within one simple and easy-to-use framework, so the developers time to product is minimized. HALT also functions well in a teaching environment, as it follows the KISS principle: Make everything as simple as possible, but no simpler. The bright and simple display of information make the user interface a fun and powerful way to learn and develop Motorola 68000 assembly language code.

HALT is also a tool for developers. HALT produces only machine instructions which are in a strict subset of the Motorola 68000 machine instruction set. Therefore, any program in HALT should also run on any machine that implements the basic 68000 instruction set.

This document provides a guideline for the software development process. The content should reflect those points expressed in the HALT philosophy[3]. All specifications outlined herein are mutable if they do not reflect the later choices of the developers.

Platform

To provide the widest accessability to academia, HALT will be cross-platform. Target platforms are Microsoft Windows (2000, XP, Vista, and NT), Linux (xorg), and OS X. Portable source code will be available so that HALT may be transferred to future desirable platforms with minimal effort.

Implementation Language

All code associated with HALT must be written in C++. Libraries used will be cmath, STL, OpenGL, and GLUT. Any other libraries must be dynamically compiled, and must run on Windows, Linux, and Windows. All code associated with HALT must be written to compile and run on Windows, Linux, and Apple OS X.

Development Process

Fully collaborative development will be accomplished using a non-locking version control (merging) version control system, subversion. It will also allow easy maintenance of code branches, and to "rewind" mal-appropriate code changes. The code for all versions after the initial release are browseable with change logs and blame tags.

Program Design

HALT will consist of several object modules, each with its own responsibility. Since HALT is an integrated development environment consisting of an editor, simulator, and machine inspector, functionality will be tightly coupled within objects. The User's Guide [2] provides as an Appendix the class dependency and class inheritance diagrams.

Documentation

Good documentation is extremely important, as the product will be used primarily within academia. User support, bug tracking, and feature requests are also imperative.

All documents, including this one, will be written in the LaTeXtypesetting system, and made available in both .PDF and bound hard copy. This is so it can be treated as source code in the subversion version control system, as well as providing for later publication of reference materials.

Two web sites will be provided. A General web site will provide news, release versions, documentation, and contact information, and can be found at [5]. A more extensive development site will provide bug tracking, feature requests, a discussion forum, version control with code rollback, multi-lingual Wiki, mailing lists, task assignment, and other features. It is located at [4].

Assembly Language Features

The language shall be a subset of Motorola 68000 Assembly. Following is a list of commands that must be supported before HALT 1.0.

Instructions: **STOP, MOV, ADD, SUB, MUL, DIV, AND, OR, NOT, EOR, LEA, BRA, BLE, BGT, BLE, BGE, BNE, CLR, NEG, NOP, BSR**.
Addressing modes: **Data Register Direct, Address Register Direct, Address Register Indirect, Address Register Indirect with Pre-Decrement, Address Register Indirect with Post-Increment, Symbolic Address Indirect, Literal**.

All assembly language instructions must behave in the simulator and virtual machine as they are defined in the Motorola 68000 Programmer's Reference[1]. Language features which are not listed here are candidates for inclusion in HALT's subset of Motorola 68000 assembly, but are not required. All language choices must reflect the HALT design philosophy[3].

Added Value Features

Some higher-level features such as array declaration, and indirect indexing may be provided, even though they are not part of the original Motorola 68000 instruction set[1]. These features must be implemented in such a way that the machine language produced by HALT's assembler is still a strict subset of the Motorola 68000 machine language. In this way, programs developed with HALT are assured to run on any 68k-based hardware.

Tentative Acknowledgements

- **BerliOS** for hosting the project web site and providing project management support.
- **NASA** for providing funds for IDCC'05-06.
- **USM School of Computing** for hosting IDCC'05-06.

References

[1] Motorola Corporation. **Motorola M68000 Programmer's Reference Manual**. 1992.
Available at www.freescale.com or with request from Motorola Corporation.

[2] Matthew Bennett and Michael Erwin. **HALT User's Guide**. 2006.
Available at http://halttool.berlios.de or printed by request.

[3] Michael Erwin. **HALT Design Philosophy**. 2006.
Available at http://halttool.berlios.de or printed by request.

[4] Berlios.de project management site for HALT. http://developer.berlios.de/projects/halttool

[5] HALT home page. http://halttool.berlios.de