HALT design principles

Mike Erwin
February 2006

Tightly integrate all tools necessary for assembly programming. HALT has a source code editor, incremental assembler, and a system simulator.

Make assembly language approachable, without dumbing it down. It should be easy to use *and* extremely powerful.

Highlight important things, and let unimportant things fade into the background.

Avoid indirect references. Don't make the user count or correlate things when it's possible to *show* them.

- no line numbers (*this* line, not *line 23*)
- address registers point directly to memory words (rings around dots)
- use symbolic names, not numeric addresses

Provide a minimal but complete instruction set. This includes arithmetic, logic, flow control, and data movement. Save the exotic stuff for a later version. And *never* implement the BCD instructions!

Binary compatibility with 68000. HALT's processor is a pure 16-bit subset, so the machine code should run on the real thing. Macintosh, Atari ST, Amiga, Sega Genesis, and TI-89 programmers rejoice!

The assembly language is similar to standard Motorola syntax, but provides more regularity for numbers, addressing modes, and comments. Data definitions are completely overhauled.

Assembly language allows some weird things. Most of the time these are unintentional bugs, but sometimes the programmer knows exactly what they are doing. Alert the user to weirdness, but allow it if they choose.

Emit relocateable code.

The program *is* the memory. A program can use the entire address space, but don't show any part of memory that is not being used. Memory outside of the program can not be accessed; for our purposes it does not exist.