19

A BASIC Interpreter for Microprocessors

Paul Allen and William Gates
MITS, Inc.

Design Philosophy

In January of 1975, the implementation of a BASIC interpreter for the 8080 microprocessor was begun.

BASIC was chosen for a variety of reasons. First, BASIC is a well known and easy to learn language for which a wealth of text books and tutorial information is readily available. Secondly, a small interpreter can be readily defined and constructed in a relatively straight foward manner. Finally, the interactive nature of BASIC is well known for its ability to provide rapid program generation and efficient debugging.

Compilers were considered but rejected for a variety of reasons. Compilers require large amounts of memory which is often in short supply on a microprocessor system. Also, mass storage devices such as tape units or floppy disks are almost an absolute necessity for efficient use of compilers. Lastly, the execution speed of floating point routines on an eight bit processor makes a compiler's advantage over an interpreter a factor of only two or three, and execution speed was not an overiding concern.

Implementation

Once a BASIC Interpreter was chosen as the final product, development began immediately. One of the cardinal rules of software design is to examine other implementations of a similar product so that useful design strategies are not re-invented. A number of different BASIC interpreters were examined and several useful ideas were used in the microprocessor version.

Coding started once the data structures and flow of the interpreter were well defined. At the same time, work was begun on a number of software tools that could be used to generate the interpreter.

We felt that a large time shared mainframe system with which we were very familiar would suit our needs for a development system. However, we decided to write our own development tools instead of using cross assemblers and simulators available from time sharing vendors or microprocessor manufacturers.

First, an assembler was contructed using the MACRO facility of the host assembler. This provides the user with all the features of the host assembler including conditional assembly and cross reference listings. Generation of the assembler took about a single man week.

Next, a simulator for the 8080 was written in the hosts assembly language. This makes simulations run very quickly, about an order of magnitude slower than an actual 8080 microprocessor. The inherent speed of the simulator reduced CPU charges and made dubugging of long routines less time consuming. The standard host debugging package was modified to recognize 8080 format operation codes and register references. Also, since the host debugging package allowed references to assembly symbols at debug time, all labels and other assembly time symbols were available during the debugging process.

These powerful tools, which were developed in under a man month, made the generation of the the first "cut" at BASIC a very quick process. The first version was ready after less than five man months of effort by a three man programming team. One programmer wrote the floating point arithmetic and input and print routines, another programmer worked on statement execution routines and another wrote the development tools and the interpreter's program editor.

This first version was executed successfully the first time it was loaded into an 8080 microcomputer, a MITS ALTAIR 8800.

Since that time, BASIC has evolved into a number of different versions, each with an increased number of features. It has also been run on other 8080 based systems, such as INTEL's development systems and other custom 8080 based units.

Versions of 8080 BASIC

The smallest version of these BASICs is the 4K version, which has all the features of the original Dartmouth BASIC. Direct or immediate execution of statements was allowed so

that the user could type 'PRINT 2+2" and immediately receive the response '4'.

Next is the 8K version, which uses about 6K of memory. A number of distinctive features, such as boolean arithmetic, characters strings with substring operators and string arrays, PEEK and POKE to read or write a byte from any memory location, and INP and OUT which are used to read and write a byte from an I/O port are available in this version. These last two features are extremely useful when debugging new I/O devices and interfaces, and are regularily used by MITS engineers for this purpose.

Also, these features allow the user to perform low speed I/O to external devices without having to resort to assembly language. An assembly language subroutine call is also provided so the user can take advantage of the speed and flexibility of assembly language where necessary.

Extended BASIC, which takes approximately 10K bytes of memory , adds integer and double precision floating point arithmetic, PRINT USING for formatted output, and many other improvements.

Double precision floating point gives the user sixteen digits of accuracy when needed. Integer variables are economical in their use of memory space, and speed up program execution where integer quantities are used.

Another special feature allows the user to default variable names that start with a certain character to a particular variable type (integer, string, single and double precision). This is similar to FORTRAN's IMPLICIT statement.

A powerful EDIT command also makes changes to already existing program lines an easy task.

Disk Extended BASIC, which requires 16K bytes of memory, adds a full file structure system for floppy disks. This includes both sequential and random file access methods, as well as commands for deleting, renameing, loading and saving disk files.

6800 BASIC

A 6800 version of 8K BASIC was coded from scratch using the 8080 version as a guide. The 6800 version required slightly more memory, about 6200 bytes as compared to 5900 for the 8080 version, but the 6800 version is slightly faster than the 8080 version. This demonstrates that the 8080 and 6800 are extremely close in terms of memory

effiency and speed. However, the 6800 tends to be simpler to program for a given application, while the 8080 requires more sophisticated programming techniques.

Future Software Development Plans

Plans for the future include a version of Extended BASIC on ROM, and continued refinement and additions to the already existing versions. One of the unique features of the ROM version is that a BASIC program can be put in ROM or PROM, and then be executed by the ROM BASIC interpreter. This means that complete turn-key systems can be developed in BASIC, and then placed in PROM when completely debugged.

Also in the works are APL interpreters for the 8080 and 6800.