BUILD YOUR OWN BASIC

by Dennis Allison & Others
(reprinted from People's Computer Company Vol. 3. No.4)

A DO IT YOURSELF KIT FOR BASIC??

Yes, available from PCC with this newspaper and a lot of your time. This is the beginning of a series of articles in which we will work our way through the design and implementation of a reasonable BASIC system for your brand X computer. We'll be working on computers based on the INTEL 8008 and 8080 microprocessors. But it doesn't make much difference - if your machine is the ZORT 9901 or ACME X you can still build a BASIC for it. But remember, it's a hard job and will take lots of time particularly if you haven't done it before. A good BASIC system could easily take one man six months!

We'd likeeveryone interested to participate in the design. While we could do it all ourselves, (we have done it before) your ideas may be better than ours. Maybe we can save you, or you can save us, a lot of work or problems. Write us and we'll publish your letter and comments.

WHICH BASIC?

There is not any one standard BASIC (yet). The question is which BASIC should we choose to implement. A smaller (fewer statements, fewer features) BASIC is easier to implement and (more important) takes less space in the computer. Memory is still expensive so the smaller the better. Yet maybe we can't give up some goodies like string variables, dynamic array allocation, and so on.

There is a standard version of BASIC which is to be the minimal language which can be called BASIC. It's a pretty big language with lots of goodies. Maybe too big. Is there any advantage to being compatible with, say, the EDU BASICS? We don't have to make any decision yet; but the time will come . . .

COMPILER OR INTERPRETER?

We favor using an interpreter. An interpreter is a program which will execute the BASIC program from its textual representation. The program you write is the one which gets executed.A compiler converts the BASIC program into the raw machine code for the machine it is to run on. Compiled code is a lot faster, but requires more space and some kind of mass storage device (tape or disk). Interpretative BASIC is the most common on small machines.

DIRECT MODE?

Some kind of "desk calculator" mode of operation would be nice. At least, we would like to be able to look at and set different variables in a program and restart execution at any given point. This feature makes it easier to find and gently terminate the existence of "bugs."

HOW MUCH MEMORY? AND ... WHAT KIND?

Can we make some guesses about how big the BASIC system will be? Only if you don't hold us to It. Suppose we want to be able to run a 50 line BASIC program. We need about 800 bytes to store the program, another 60 or so bytes for storing program values (all numeric) without leaving any space for the interpreter and its special data. Past experience has shown that something like 6 to 8 Kbytes are needed for a minimum BASIC interpreter and that at least 12K bytes are necessary for a comfortable system. That's a lot of memory, but not too much more than you need to run the assembler. A lot of BASIC could be put into ROM (Read Only Memory) once developed and checked out. ROM is a lot cheaper than RAM (Read and Write) memory, but you can't change it. It's lots better to make sure everything works first.

But ... if we can agree on some chunks of code and get it properly checked out, some enterprising person out there might make a few thousand ROMs and save us all some $$$. Let's see now ... how about ROMs for floating point arithmetic, integer arithmetic, Teletype I/O...

DATA STRUCTURES

Data structures are places to put things so you can find them or use them later. BASIC has at least three important ones: a symbol table which looks up a program name A or Z9 or A$, with its value. If we had a big computer where space was not a huge problem, we could simply preallocate all storage since BASIC provides for only 312 different names excluding arrays. When memory is so costly this doesn't make much sense. Somewhere, also, we've got to store the names which BASIC is going to need to know, names like LET and GO TO and IF. This table gets pretty big when there are lots of statements.

Lastly, we need some information about what is a legal BASIC statement and which error to report when it isn't. These tables are called parsing tables they control the decomposition of the program into its component parM

STRATEGY

Divide and Conquer is the programmers maxim. BASIC will consist of a lot of pieces which communicate with each other. Then pieces themselves consist of smaller pieces which themselves consist of smaller pieces, and so forth down to the actual code. A large problem is made manageable by cutting it into pieces.

What are the pieces, the building blocks of BASIC? We see a bunch of them:

* a supervisor which determines what is to be done next. It receives control when BASIC is loaded.
* a program and line editor. This program collects lines as they are entered from the keyboard and puts them into a part of computer memory for later use.
* a line executor routine which executes a single BASIC statement, whatever that is.
* a line sequence which determines which line is to be executed next.
* a floating point package to provide floating point on a machine without the hardware.
* terminal I/O handler to input and output information from the Teletype and provide simple editing (backspace and line deletion).
* a function package to provide all the BASIC functions (RND, INT, TAB, etc.)
* an error handling routine (part of the supervisor).
* a memory management program which provides dynamic allocation data objects.

Then are the major ones, As we got futher into the system we'll begin to see others and we'll begin to be able to more fully defwie the function of each of these modules.

TINY BASIC

Pretend you are 7 years old and don't care much about floating point arithmetic (what's that?), logarithms, sines, matrix inversion, nuclear reactor calculations and stuff like that.

And ... your home computer is kinda small, not too much memory. Maybe its a MARK-8 or an ALTAIR 8800 with less than 4K bytes and a TV typewriter for input and output.

You would like to use it for homework, math recreations and games like NUMBER, STARS, TRAP, HURKLE, SNARK, BAGELS,...

Consider then, TINY BASIC

* Integer arithmetic only - 8 bits? 16 bits?
* 26 variables: A, B, C, D, . . ., Z
* The `RND` function - of course!
* Seven BASIC statement types

```
  INPUT
  PRINT
  LET
  GO TO
  IF
  GOSUB
  RETURN
```

* Strings? OK in `PRINT` statements, not OK otherwise.

BUILD YOUR OWN BASIC-REVIVED

(reprinted from People's Computer Companj, Vol. 4, No. 1)

WHAT IS TINY BASIC???

TINY BASIC is a very simplified form of BASIC which can be implemented easily on a microcomputer. Some of its features are:

Integer arithmetic 16 bits only

26 variables (A, B, ..., Z)

Seven BASIC statements

```
  INPUT  PRINT  LET     GOTO
  IF     GOSUB  RETURN
```

Strings only in `PRINT` statements

Only 256 line programs (if you've got that much memory)

Only a few functions including `RND`

It's not really BASIC but it looks and acts a lot like it. I'll be good to play with on your ALTAIR or whatever; better, you can change it to match your requirements and needs.

TINY BASIC LIVES!!!

We are working on a version of TINY BASIC to run on the INTEL 8080. It will be an interpretive system designed to be as conservative of memory as possible. The interpreter will be programmed in assembly language, but we'll try to provide adequate descriptions of our intent to allow the same system to be programmed for most any other machine. The next issue of PCC will devote a number of pages to this project.

> * In the meantime, read one of these.
> Compiler Construction For Digital Computers, David Gries, Wiley, 1971 493 pages, $14.95
> Theory & Application of a Bottom-Up Syntax Directed Translator Harvey Abramson, Academic Press, 1973, 160 pages, $11.00
> Compiling Techniques, F.R.A. Hopgood, American Elsevier, 126 pages $6.50

A BASIC Language Interpreter for the Intel 8008 Microprocessor A.C. Weaver, M.H Tindall, R.L. Danielson. University of Illinois Compliter.Science Dept, Urbana IL., 61801. June 1974. Report No. UIUCDCS-R-74-658. Distributed by National Technical Information Service, U.S. Commerce Dept, Springfield VA 22151. $4.25.

A BASIC language interpreter has been designed for use in a microprocessor environment. This report discusses the development of 1) an elaborate text editor and 2) a table-driven interpreter. The entire system, including text editor, interpreter, user text buffer, and full floating point arithmetic routines fits in 16K 8-bit words.

The TINY BASIC proposal for small home computers was of great interest to me. The lack of floating point arithmetic however, tends to limit its usefulness for my objectives.

As a matter of a suggestion, consideration should be given to the optional inclusion of floating point arithmetic, logarithm and trigonmetric calculation capability via a scientific calculator chip interface.*

The inclusion of such an option would tend to extend the interpreter to users who desire these complex calculation capabilities. A number of calculator chip proposals have been made, with the Suding unit being of the most interest.

Thank you for the note of 13 June, regarding my letter on the Tiny BASIC article (PCC Vol. 3 No. 4). It was with regret that I learned that the series was not continued in the next volume. Even though few responded to the article published, conceptually the knowledge and principles which would be disseminated regarding a limited lexicon, high level programming language are of importance to the independent avocational microcomputer community.

At this time, PCC may not have a wide distribution in the avocation microcomputer community. This could be possibly the cause for the low number of respondies. Never the less, this should not detract from the dissemination and importance of concepts and principles which are of significance.

The thrust of my letter of 15 April, 1975, was to suggest a mechanism for the inclusion of F.P. in a limited lexicon and memory consumptive BASIC. I hope that the implication that F.P. must be included was not read into my letter.

It is my interest that information, concepts and the principles of compiler/interpreter construction as it related to microcomputers be available to the limited budget avocational user. The MITS BASIC, which you brought up, appears from my viewpoint to be a licensed, blackbox program which is not currently available to: (a) 8008 users, (b) IMP-16 users, (c) independent 8080 users (except at a very large expense.) or (d) MC6800 users who will shortly be on line.

Presently it appears that microcomputer compiler interpretor function languages will be coming available ftom a number of sources (MITS, NITS, Processor Technology and etc.). However, few will probably deal in the conceptualizations which are the basis of the interpreter. Information which will fill the void in the interpreter construction knowledge held by the avocation builder, should be made available. I strongly urge that the series started with Vol. 3 No. 4 article he continued. Possibly the hardware, peripheral, machine programming difficulties incurred by the microcomputer builder, is prohibiting a major contribution at this time. However, I would expect that by Autumn a number of builders should have their construction and peripheral difficulties far enough adong to start thinking about higher level languages. The previous objective for the article series sounds reasonable. It was not my purpose in submitting the letter to detract from the objective of a very limited lexicon BASIC, ie., to be attractive and usable by the young and beginner due to its simplicity. If wives, children, neighbors or anyone who is not machine language or programming oriented is expected to use a home-base unit created under a restrained budget a high level language will be a necessity. It is with this foresight that I encourage the continuance of the "Build Your Own BASIC" series. This issue aside, I would like to encourage the PCC to continue the quite creditable activities which have been its order of business with regard to avocational computing.

Michael Christoffer
4139 12th NE No. 400
Seattle, Wash. 98105

* Please see Dr Robert Suding's article on p. 18