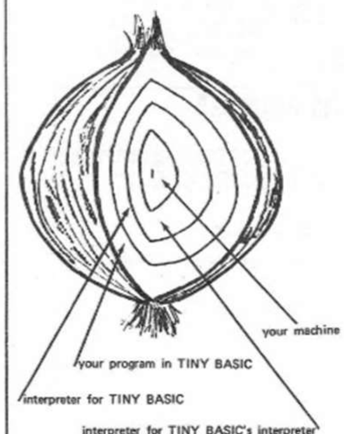
IMPLEMENTATION STRATGIES AND ONIONS

When you write a program in TINY BASIC there is an **abstract machine** which is necessary to execute it. If you had a compiler it would make in the machine language of your computer a program which emulates that abstract machine for your program. An **interpreter** implements the abstract machine for the entire language and rather than translating the program once to machine code it translates it dynamically as needed. Interpreters are programs and as such have their's as abstract machines. One can find a better instruction set than that of any general purpose computer for writing a particular interpreter. Then one can write an interpreter to interpret the instructions of the interpreter which is interpreting the TINY BASIC program. And if your machine is microprogrammed (like PACE), the machine which is interpreting the interpreter interpreting the interpreter interpreting BASIC is in fact interpreted.

This multilayered, onion-like approach gains two things: the interpreter for the interpreter is smaller and simpler to write than an interpreter for all of TINY BASIC, so the resultant system is fairly portable. Secondly, since the major part of the TINY BASIC is programmed in a highly memory efficient, tailored instruction set, the interpreted TINY BASIC will be smaller than direct coding would allow. The cost is in execution speed, but there is not such a thing as a free lunch.



LINE STORAGE

The TINY BASIC program is stored, except for line numbers, just as it is entered from the console. In some BASIC interpreters, the program is translated into an intermediate form which speeds execution and saves space. In the TINY BASIC environment, the code necessary to provide the transformation would easily exceed the space saved.

When a line is read in from the console device, it is saved in a 72-byte array called **LBUF** (Line BUFfer). At the same time, a pointer, CP, is maintained to indicate the next available space in LBUF. Indexing is, of course, from zero.

Delete the leading blanks. If the string matches the BASIC line, advance the cursor over the matched string and execute the next IL instruction. If the match fails, continue at the IL instruction labeled lbl.

QUICK REFERENCE GUIDE FOR TINY BASIC

LINE FORMAT AND EDITING

- Lines without numbers executed immediately
- Lines with numbers appended to program
- Line numbers must be 1 to 255
- Line number alone (empty line) deletes line
- Blanks are not significant, but key words must contain no unneeded blanks
- '←' deletes last character
- X^c deletes the entire line

EXECUTION CONTROL

CLEAR delete all lines and data
RUN run program
LIST list program

EXPRESSIONS

Operators

Arithmetic		Relational	
+	−	>	>=
*	/	<	<=
		=	<>,><

Variables
A.....Z (26 only)

All arithmetic is modulo 2^{15} (±32762)

INPUT / OUTPUT

```
PRINT X,Y,Z
PRINT 'A STRING'
PRINT 'THE ANSWER IS'
INPUT X
INPUT X,Y,Z
```

ASSIGNMENT STATEMENTS

```
LET X=3
LET X= -3+5*Y
```

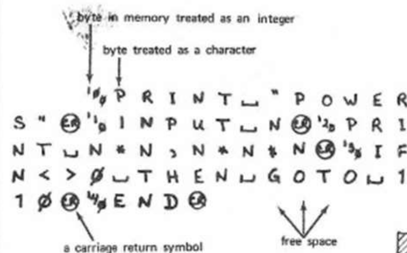
CONTROL STATEMENTS

```
GOTO X+10
GOTO 35
GOSUB X+35
GOSUB 50
RETURN
IF X>Y THEN GOTO 30
```


The TINY BASIC program is stored as an array called **PGM** in order of increasing line numbers. A pointer, **PGP**, indicates the first free place in the array. PGP=0 indicates an empty program; PGP must be less than the dimension of the array PGM. The PGM array must be reorganized when new lines are added, lines replaced, or lines are deleted.

Insertion and deletion are carried on simultaneously. When a new line is to be entered, the PGM array searches for a line with a line number greater than or equal to that of the new line. Notice that lines begin at PGM (0) and at PGM (j+1) for every j such that PGM (j)=[carriage return]. If the line numbers are equal, then the length of the existing line is computed. A space equal to the length of the new line is created by moving all lines with line numbers greater than that of the line being inserted up or down as appropriate. The empty line is handled as a special case in that no insertion is made.

TINY BASIC AS STORED IN MEMORY



ERRORS AND ERROR RECOVERY

There are two places that errors can occur. If they occur in the TINY BASIC system, they must be captured and action taken to preserve the system. If the error occurs in the TINY BASIC program entered by the user, the system should report the error and allow the user to fix his problem. An error in TINY BASIC can result from a badly formed statement, an illegal action (attempt to divide by zero, for example), or the exhaustion of some resource such as memory space. In any case, the desired response is some kind of error message. We plan to provide a message of the form:

! mmm AT nnn

where mmm is the error number and nnn is the line number at which it occurs. For direct statements, the form will be:

! mmm

since there is no line number.

Some error indications we know we will need are:

1 Syntax error
2 Missing line
3 Line number too large
4 Too many GOSUBs
5 RETURN without GOSUB
6 Expression too complex
7 Too many lines
8 Division by zero

THE BASIC LINE EXECUTOR

The execution routine is written in the interpretive language, IL. It consists of a sequence of instructions which may call subroutines written in IL, or invoke special instructions which are really subroutines written in machine language.

Two different things are going on at the same time. The routines must determine if the TINY BASIC line is a legal one and determine its form according to the grammar; secondly, it must call appropriate action routines to execute the line. Consider the TINY BASIC statement:

GOTO 100

At the start of the line, the interpreter looks for BASIC key words (LET, GO, IF, RETURN, etc.) In this case, it finds GO, and then finds TO. By this time it knows that it has found a GOTO statement. It then calls the routine EXPR to obtain the destination line number of the GOTO. The expression routine calls a whole bunch of other routines, eventually leaving the number 100 (the value of the expression) in a special place, the top of the arithmetic expression stack. Since everything is legal, the XFER operator is invoked to arrange for the execution of line 100 (if it exists) as the next line to be executed.

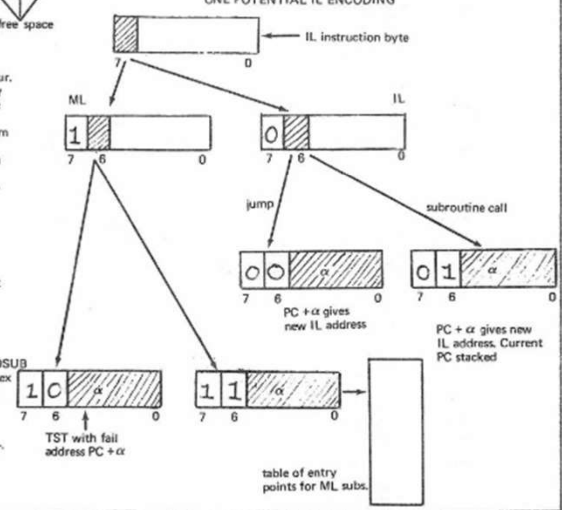
Each TINY BASIC statement is handled similarly. Some procedural section of an IL program corresponds to tests for the statement structure and acts to execute the statement.

ENCODING

There are a number of different considerations in the TINY BASIC design which fall in this general category. The problem is to make efficient use of the bits available to store information without loosing out by requiring a too complex decoding scheme.

In a number of places we have to indicate the end of a string of characters (or else we have to provide for its length somewhere). Commonly, one uses a special character (NUL = 00H for example) to indicate the end. This costs one byte per string but is easy to check. A better way depends upon the fact that ASCII code does not use the high order bit; normally it is used for parity

ONE POTENTIAL IL ENCODING



Tiny Basic - 1975

Integer only
No strings
No math functions
Easy to adapt and extend
Fits in 2K RAM

Commands:

LIST RUN NEW NEXT LET IF GOTO GOSUB RETURN REM FOR INPUT PRINT STOP RND ABS SIZE TO STEP

Imagine a microcomputer

Imagine a microcomputer with all the design savvy, ruggedness, and sophistication of the best minicomputers.

Imagine a microcomputer supported by dozens of interface, memory, and processor option boards. One that can be interfaced to an indefinite number of peripheral devices including dual floppy discs, CRT's, line printers, cassette recorders, video displays, paper tape readers, teleprinters, plotters, and custom devices.

Imagine a microcomputer supported by extensive software including Extended BASIC, Disk BASIC, DOS and a complete library of business, developmental, and industrial programs.

Imagine a microcomputer that will do everything a mini will do, only at a fraction of the cost.

You are imagining the Altair™ 8800b. The Altair 8800b is here today, and it may very well be the mainframe of the 70's.

The Altair 8800b is a second generation design of the most popular microcomputer in the field, the Altair 8800. Built around the 8800A microprocessor, the Altair 8800b is an open ended machine that is compatible with all Altair 8800 hardware and software. It can be configured to match most any system need.

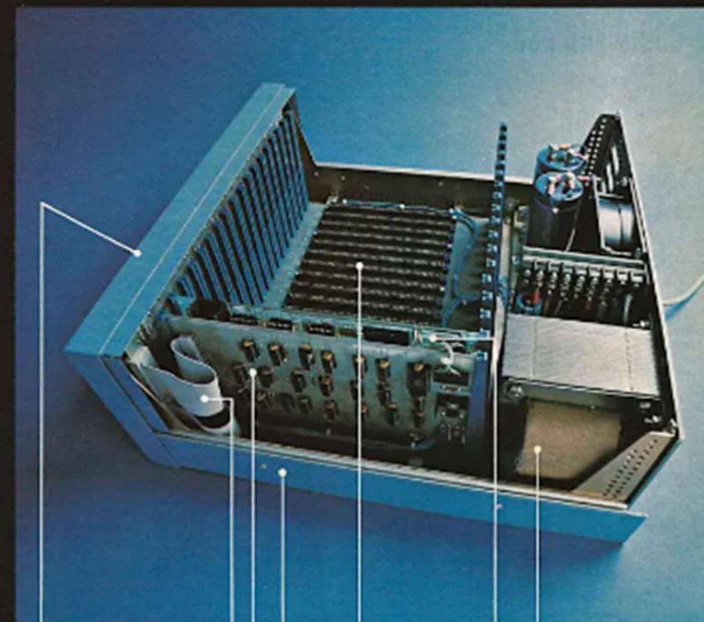
MITS' plug-in compatible boards for the Altair 8800b now include: 4K static memory, 4K dynamic memory, 16K static memory, multi-port serial interface, multi-port parallel interface, audio cassette record interface, vectored interrupt, real time clock, PROM board, multiplexer, A/D convertor, extender card, disc controller, and line printer interface.

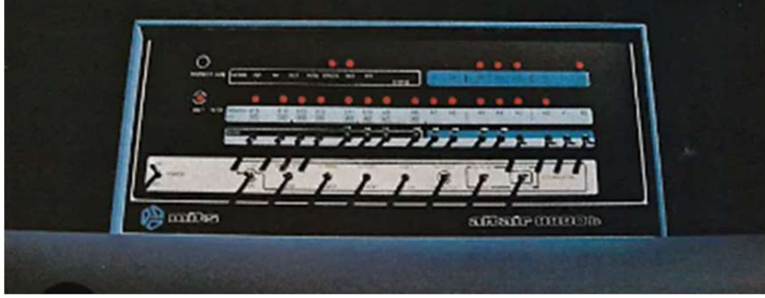
MITS' peripherals for the Altair 8800b include the Altair Floppy Disc, Altair Line Printer, teletypewriters, and the soon-to-be-announced Altair CRT terminal.

Introductory prices for the Altair 8800b are $840 for a kit with complete assembly instructions, and $1100 for an assembled unit. Complete documentation, membership into the Altair Users Club, subscription to "Computer Notes," access to the Altair Software Library, and a copy of Charles J. Sippl's Microcomputer Dictionary are included. BankAmericard or Master Charge accepted for mail order sales. Include $8 for postage and handling.

Shouldn't you know more about the Altair 8800b? Send for our free Altair Information Package, or contact one of our many retail Altair Computer Centers.

mits 2450 ALAMO S.E. ALBUQUERQUE, NEW MEXICO 87106 (505) 243-7821





Redesigned front panel. Totally synchronous logic design. Same switch and LED arrangement as original Altair 8800. New back-lit Duralith (laminated plastic and mylar bonded to aluminum) dress panel with multi-color graphics. New longer, flat toggle switches. Five new functions stored on front panel PROM including: DISPLAY ACCUMULATOR (displays contents of accumulator), LOAD ACCUMULATOR (loads contents of the 8 data switches (A7-A0) into accumulator), OUTPUT ACCUMULATOR (Outputs contents of accumulator to I/O device addressed by the upper 8 address switches), INPUT ACCUMULATOR (inputs to the accumulator from the I/O device), and SLOW (causes program execution at a rate of about 5 cycles per second—for program debugging).

Full 18 slot motherboard.

Rugged, commercial grade Optima cabinet.

New front panel interface board buffers all lines to and from 8800b bus.

Two, 34 conductor ribbon cable assemblies. Connects front panel board to front panel interface board. Eliminates need for complicated front panel/bus wiring.

New, heavy duty power supply: +8 volts at 18 amps, +18 volts at 2 amps, -18 volts at 2 amps. 110 volt or 220 volt operation (50/60 Hz). Primary tapped for either high or low line operation.

New CPU board with 8080A microprocessor and Intel 8224 clock generator and 8216 bus drivers. Clock pulse widths and phasing as well as frequency are crystal controlled. Compatible with all current Altair 8800 software and hardware.

altair 8800-b

Altair 4K/8K Basic – 1975 (Microsoft)

Floating point (6 digits)
Strings in 8K version
Math functions

4K/8K Commands:

ABS CLEAR DATA DIM END FOR GOSUB GOTO IF INPUT
INT LET LIST NEW NEXT PRINT READ REM RESTORE
RETURN RND RUN SGN SIN SQR STEP STOP TAB THEN
TO USR

8K Commands:

ASC AND ATN CHR$ CLOAD CONT COS CSAVE DEF EXP
FN FRE INP LEFT$ LEN LOG MID$ NULL ON OR NOT
OUT PEEK POKE POS RIGHT$ SPC STR$ TAN VAL WAIT

Experience the excitement of owning the finest *personal* computer – IMSAI 8080

Waiting for you — all the incredible performance and power of the IMSAI 8080. And at a price you would normally pay for a home music system: $699 in kit; $931 assembled.

WORLD OF USES

The IMSAI 8080 is a superbly engineered, quality computer. It is versatile, expandable and powerful, putting literally hundreds of applications and uses at your fingertips. Imagine sitting at your desk and enjoying interaction with your IMSAI 8080! Press the *on* switch and you're ready for game playing, research, education, business applications, or for laboratory instrument control. It has all the power you need, and more, to make your application ideas come alive.

GROWS WITH YOU

The IMSAI 8080 is designed for many years of pleasure. With its open-chassis engineering you can expand your system by adding peripherals and interfaces. The 22-slots and 28 amp power supply mean that you can plug-in today's, plus *tomorrow's* modules.

Right now you can add a module for displaying color graphics and characters on TV; a ready-to-use keyboard; small and large printers, and a single interface that lets you attach multiple devices including a cassette tape recorder. Expect the latest, exciting equipment from IMSAI. We are committed to leadership in this expanding technology.

EASY TO PROGRAM

With our BASIC language you can operate the IMSAI 8080 quickly and easily. Technically knowledgeable? Use our assembly language to develop sophisticated and unique software.

If you're thinking personal computer, treat yourself to the very best – IMSAI 8080

Send for free four-color brochure or $1 for catalog. Call or write for name of nearest dealer.

IMSAI

IMSAI Manufacturing Corporation
14860 Wicks Blvd.
San Leandro, CA 94577
(415) 483-2093
TWX 910-366-7287

Prices: USA Domestic

IMSAI 4K/8K Basic – 1976/1977

Floating point (6 digits)
Strings in 8K version
Math functions in 8K version

4K/8K Commands:

LIST NEW RUN RND ABS SQR SGN IF READ DATA FOR
NEXT GOSUB RETURN INPUT PRINT GOTO LET STOP END
REM STEP THEN

8K Commands:

DIM ON RESTORE DEF CHANGE CALL POKE OUT RANDOMIZE
INT SIN COS TAN LN LOG EXP PEEK PI LEN INSTR ASCII CHR$
STRING$ NUM$ VAL SPACE$ LEFT$ RIGHT$ MID$ POS TAB INP

Why pay $3,000^00 for a source listing of a higher level microcomputer language?

SCELBAL is here. Now. Only $49.

Introducing the new microcomputer language that's simpler than machine language . . . **SCELBAL. SC**ientific **EL**ementary **BA**sic **L**anguage for 8008/8080 systems. This complete, illustrated program book . . . a gold mine for routines, techniques, source listings, flow charts and more . . . took several years to develop. Now it's yours for many years to come. The first time that intimate details of a higher level language program has been offered to the public at such a low, low price.

Look at all you get! **5 Commands:** SCR, LIST, RUN, SAVE, LOAD. **14 Statements:** REM, LET, IF . . . THEN, GOTO, FOR with STEP, END, INPUT, PRINT, NEXT, GOSUB, RETURN and optional DIM. **7 Functions:** INT, SGN, ABS, SQR, RND, CHR, TAB. And, it runs in 8K and more.

You read right! This book is yours for only $49 ppd. Now you'll have all the data you need to custom-tailor a high level language for your system . . . at a fraction of what it might cost until now. The program alone is worth more than its price. The revealed secrets of its operation are virtually priceless. Order today! "SCELBAL — A Higher Level Language for 8008/8080 Systems." Only $49 ppd. Delivery in June.

SCELBI COMPUTER CONSULTING INC. 1322 Boston Post Rd. Rear
Milford, CT 06460/(203) 874-1573

Master Charge, Postal and Bank Money Orders only accepted. Pricing, specifications, availability subject to change without notice. Prices for U.S. and Canadian delivery at book mailing rate. Add $2.50 for each publication if Priority Air Service (U.S.) desired. Overseas include $5.00 for each publication for Airmail service.

SHOCKING!

SCELBAL Basic – 1976

Versions for 8008 and 8080
Well documented source code
Floating point (6 digits)
Optional strings and math functions

Standard Commands:

REM IF LET GOTO PRINT INPUT FOR NEXT STEP GOSUB RETURN DIM END INT SGN ABS SQR RND CHR TAB UDF LIST RUN LOAD SAVE

Extended Commands:

CHR$ LEN$ ASC$ VAL$ RIGHT$ LEFT$ MID$ SIN COS TAN ATN LOG EXP

XY BASIC

XY BASIC
XY BASIC
XY BASIC
XY BASIC T.M.

MARK WILLIAMS COMPANY

XYBasic – 1977

Modular design, easy to adapt
Supports standalone 8080, CP/M, ISIS-II, Intellec 8, MDS, SEC 80
Long variable names
Optional strings, floating point (6 digits), math functions
Unique machine interface I/O commands

Standard Commands:

LET PRINT RUN LIST NEW CLEAR GOTO CONT INPUT REM IF THEN STOP END GOSUB RETURN READ DATA RESTORE FOR NEXT STEP ON DIM ABS SGN MOD RND RANDOMIZE FRE UNS DEF TAB POS NULL ASSIGN SAVE LOAD TRACE UNTRACE BREAK UNBREAK SET RESET ROTATE RSHIFT LSHIFT BCD BIN MSBYTE LSBYTE JOIN GET DELAY OUT IN PEEK POKE SENSE WAIT ENABLE DISABLE CALL SCALL MOVE EXEC FIRST LAST TRAP UNTRAP AUTO DELETE EDIT RENUM OPEN CLOSE MARGIN LINPUT EOF DIR SCRATCH CLEAR

Extended Commands:

SQR LOG EXP SIN COS TAN ATN INT LEN LEFT$ RIGHT$ MID$
CHR$ ASC INSTR GET$ STR$ VAL CLEAR FRE$ HEX$ OCT$ BIN$


1976
BASIC Interpreters
128MB
CompactFlash
RoHS
V4
A308-V4
MADE IN TAIWAN
R.O.C