# APL/Z80:
# AN APL INTERPRETER FOR Z80 MICROCOMPUTERS

John E. Howland
Department of Computing and Information Sciences
Trinity University
715 Stadium Drive
San Antonio, Texas 78284

Philip Van Cleave
Director, Software Development
Vanguard Systems Corporation
6812 San Pedro
San Antonio, Texas 78216

## Abstract

An APL interpreter for Z80 microcomputers is described. APL/Z80 extends APL applications to the realm of low-cost personal computers. For example, APL/Z80 can be configured to run on Z80 systems with video console, 28K byte workspace, dual floppy disks for less than $6500.00. The power of such an APL system is comparable to APL on a large time sharing computer for many applications.

## 1 Introduction

### 1.1 APL/Z80, An Overview

One of the main goals of this project is to provide a low cost implementation of a large subset of APL to run on inexpensive personal computers. The Z80 microcomputer was selected because of its 16-bit arithmetic capability, BCD instructions, instructions for moving and comparing blocks of memory, and multiple index registers. The Z80 was judged to be the best available processor to implement a low-cost personal APL system.

Design objectives included an easy-to-use, highly interactive system which gives instantaneous response during program development tasks, and adequate response (at most a few seconds) to all but intensively compute bound tasks. Section 5 describes the extent to which these goals have been met. An early objective was to provide a system which consumed no more than 24K bytes of code. Present system size is 28K bytes which includes code for a video-based input editor which responds to cursor control characters. This editor allows the cursor to be moved left or right, non-destructively, over the input line as well as character insertion or deletion. The editor is active any time input can be accepted from the system console. A control character exists to recall the last line entered. When used, the last line can be edited, using the cursor control characters, then resubmitted to the system for processing by entering a carrier return. This process can be repeated several times to correct and/or develop complicated expressions without retyping the line each time.



### 1.2 Functions and Operators

The 24K-byte system-size design objective dictated certain compromises regarding which primitive functions, operators, and system capabilities would be implemented. Every effort was made to provide a sufficiently rich set of primitive functions so as to be able to easily implement the entire set of APL primitives as defined functions (see section 4 for examples). In some cases certain primitive functions, such as encode and decode, were implemented with restricted arguments, similar to their implementation in early APL systems. In other cases certain functions, defined on integer domains, such as deal and roll, were restricted to integers less than or equal to 65535 for efficiency reasons. Inner product, transpose, lamination, some circular functions, monadic format, matrix inverse and matrix divide are not implemented. A complete list of functions and operators implemented is given in tables 2, 3, and 4. A workspace of defined APL functions which implement the missing functions and operators is distributed with the system.

### 1.3 System Dependencies

System variables have abreviated names in APL/Z80. For efficiency reasons these variables have been assigned fixed locations in the workspace and hence, exist only as global variables. A list of system variables is given in table 5.

APL/Z80 has the usual workspace library which exists on a system mass-storage device

(usually a floppy disk). Performance characteristics of most personal computer mass storage units to not permit efficient search of workspaces for objects to be copied in the active workspace. APL, as implemented on most large systems, does this kind of search in main processor memory. This was not possible in a 64K byte address space, consequently, the concept of a copy object library was developed for APL/Z80. This library consists of files of individual APL objects (variables or functions). New system commands have been included to access this library. Table 6 gives a complete list of all system commands implemented.

## 2 Structure of APL/Z80

APL/Z80 consists of three modules, the supervisor, interpreter, and auxiliary processors. The supervisor handles all communication with the Z80 computing environment, while the interpreter evaluates expressions, and provides function definition facilities. Auxiliary processors are used to provide communication between APL and non APL programs such as file systems, graphics processors, etc. These modules communicate with each other using a system of restart instructions.

### 2.1 Supervisor

The supervisor program provides all Z80 computer interfaces. For example, all input/output device handling and buffering are performed by the supervisor. The input editor, described earlier, is a part of the supervisor. Console-break-condition detection is the responsibility of the supervisor. Character-set translation, overstrike recognition and generation are also performed by the supervisor. The supervisor transmits completed code strings to the interpreter and receives formatted values from the interpreter which are displayed. The supervisor also handles disk input/output to transmit workspaces and copy objects to and from the active workspace.

### 2.2 Interpreter

The interpreter is organized into two modules. The first of these interfaces with the supervisor and accepts a code string and transforms this string into a tokenized form and prepares an appropriate workspace entry. This entry is either evaluated by the second module or linked together with other lines of a defined function if the interpreter is in definition mode. Syntax analysis is done using decision table techniques which provide high interpretation speed and modular construction.

### 2.3 Data Types

Representation of four basic data types were chosen according to the arithmetic capability of the Z80 computer, ease and speed of conversion between data types as well as a desire to do precise decimal arithmetic in data processing applications. The following data representations are used in APL/Z80.

| | |
|---|---|
| CHARACTER | 1 character per byte |
| BINARY | 1 bit per byte<br>(hex 00=0<br>hex 01=1) |
| INTEGER | 1 integer per 3 bytes<br>(5 BCD packed digits) |
| REAL | 1 real per 6 bytes<br>(10 BCD digits<br>significance) |

Some operational speed, in functions using REAL values, is sacficed to gain decimal arithmetic for data-processing applications and ease of data type conversion. The latter is important in providing instantaneous response in evaluation of simple expressions and program development. The representation for BINARY data was chosen for high computational speed and ease of conversion at the expense of memory space. These representations were chosen after implementing several alternative representations and doing performance evaluation.

### 2.4 Workspace Organization

Each workspace occupies a variable amount of space depending on its contents. Objects are created and destroyed dynamically. A two-way linked list is used to link objects together. The garbage collection process has a built-in validity verification which is capable of detecting certain workspace-damage conditions or hardware malfunctions such as memory failures. After collection, a workspace contains fixed entries, a hashed symbol table, and workspace objects compactly located in low-order addresses.

Free space exists in high-order addresses. The state indicator stack grows downward in this free space. Workspaces are always compacted before being saved in the workspace library. State-indicator information is not saved with a workspace. This allows workspace libraries to be moved between systems having different active-workspace sizes (maximum workspace size is determined by the amount of real memory available on the system) without any reformatting. The only restriction is that the portion of the active workspace used does not exceed the amount of real memory available.

## 3 Implementation

Implementation of APL/Z80 was done resident on a Z80 computer system. This necessitated several changes to vendor-supplied assembler-editor software. Source code for APL/Z80 consists of several separate assemblies, one for the supervisor, two for the interpreter, and one for each

auxiliary processor. Each of these source files is segmented into multiple files each of about 12K bytes which are assembled separately, then linked together. Total time required to assemble the entire system is about 15 minutes. Currently the source code consists of about 305K bytes.

A top-down approach was taken during implementation. The input-editor and console-display functions of the supervisor were implemented first. Next, the code-string processor which builds the token strings and provides dynamic management of the workspace was implemented. Finally, the syntax analyzer and function and operator routines were implemented. Auxiliary processors were added later. During early phases of the implementation an interactive debugging package was a part of the supervisor and was used by the implementors to examine the internal operation of various parts of the system. The debugger was later removed when it was no longer needed. During all phases of implementation it was possible to access the system through the console devices and activate the parts of the system available at that time. The constant retesting of those parts of the APL system already implemented contributed greatly to the high degree of reliability achieved in the final system.

APL/Z80 relies on a host operating system to perform input/output to system mass storage devices.

## 4. Missing Primitives

Several references exist which give formal definitions of APL functions in APL. In (4) Lathwell and Mezei described APL/360 using a small set of primitive functions. Jenkins (3) has given formal descriptions of the matrix-inverse and matrix-divide functions. These functions, with minor modifications, can be used to provide matrix inverse and matrix divide in APL/Z80. Following are examples of functions and expressions which can be used to implement some of the missing primitives.

### 4.1 Lamination

Assuming proper conformability of arrays A, B and axis I, the expression

```
R←A,[I]B
```

may be replaced by the expressions

```
R←((⌊I)↑R),1,(⌊I)↓R←ρA
R←(RρA),[⌈I]RρB
```

### 4.2 Transpose

The following function implements

```
R←⍉A
```

```
      ∇R←TRANSPOSE A;I;NEW;OLD;L;O
[10] O←⎕O
[20] NEW←⌽OLD←ρA
[30] L←ρR←(ρA)ρ1↑0↑A←,A
[40] I←⎕O←0
[50] LOOP:R[NEW⊥⌽OLD⊤I]←A[I]
[60] →(L>I←I+1)/LOOP
[70] R←NEWρR
[80] ⎕O←O
      ∇
```

### 4.3 Reversal

The following function implements

```
R←⌽A
```

for a general array A assuming the existance of reversal for vectors.

```
      ∇R←REVERSAL A;C;O
[10] R←,A
[20] A←ρA
[30] R←AρR[⎕O+,(C×(-⎕O)+⍳×/¯1↓A)∘.+⌽(-⎕O)+⍳C
      ←¯1↓A]
      ∇
```

## 5. System Performance

System performance has been evaluated mostly in a subjective way. System response is instantaneous in most program-development situations. Benchmark expressions have been timed on other small APL systems. Also, floating-point operations have been timed as well as sorting functions. These times, of course, depend on the arguments used. Unless specified otherwise, uniformly distributed random arguments have been used. On all benchmarks a Z80 processor clock of 4.0 MZ has been used. In the following figures all times are in seconds.

Figure 5.1

| Expression | APL/Z80 | IBM 5110 |
|---|---|---|
| +/(⍳100)∈⍳100 | 2.1 | 10.0 |
| (⍳100)∘.*⍳3 | 2.5 | 5.5 |
| ÷⍳1000 | 16.2 | 1.5 |

Figure 5.2

| Expression | APL/Z80 |
|---|---|
| A←?100ρ1000 | .6 |
| A∈⍳100 | 3.9 |
| A⍳⍳100 | 4.2 |
| A[⍋A] | 3.0 |

Figure 5.3

Floating-Point Execution Times (micro-sec)

| Function | APL/Z80 | APL-SYSTEM/7 |
|---|---|---|
| + | 1540 | 2050 |
| - | 1740 | 1350 |
| × | 3320 | 9100 |
| ÷ | 6560 | 18000 |
| \| | 11460 | 1550 |
| * | 200000 | 72100 |

## 6. Shared Variables

Two system functions, quad S and quad R, are provided for offering and retracting shared variables. These functions return the usual degrees of coupling. Two auxiliary processors are provided with the standard version of APL/Z80. The first allows any devices attached to the Z80 using I/O ports to be controlled using simple defined APL Functions. This processor requires two variables, for control-status and for data. The control variable is used to specify which I/O port is being accessed and gives status information when accessed. The data variable is used to perform input and output. The second processor implements a file system featuring a directly indexable sequence of components each of which may be an array of arbitrary type, shape and size (up to available workspace). This processor also uses two variables for control-status and data. The control-status variable is used to specify indexing information for direct access. If no index is specified, then access is sequential. The control-status variable gives status about the previous access of the data variable when accessed. The data variable is used to pass information to and from the file. The file processor supports up to four active files at a time. When a data variable is shared with the file processor pre-and post-dispositions as well as file size are specified. This file processor allows sophisticated file structures to be created and maintained with simple defined APL functions. Memory space required for these auxiliary processors is about 3.2K bytes.

## 7 Summary

APL/Z80 is a responsive and reliable APL system. It contains a rich subset of APL primitive functions and operators which allow nearly all common APL applications to be successfully implemented. APL/Z80 has a powerful file system. Most important, APL/Z80 extends the power of the APL language to personal computing applications at a very low cost.

## 8 Bibliography

1. Alfonseca, M., M. Tavera, and R. Casajuana, "An APL Interpreter and System for a Small Computer", IBM Systems Journal, No. 1, 1977, pp. 18-40.

2. "APL Language", Form No. GC26-3847, IBM Corporation.

3. Jenkins, M. A., "The Solution of Linear Systems of Equations and Linear Least Squares Problems in APL", Tech. Report 320-2989, IBM New York Scientific Center, 1968.

4. Lathwell, R. H., and J. E. Mezei, "A Formal Description of APL", Tech. Report 320-3008, IBM Philadelphia Scientific Center, 1971.

5. Pakin, S., "APL/360 Reference Manual", SRA, Chicago, 1972.

6. Polivka, R. and S. Pakin, "APL: The Language and Its Usage", Prentice Hall, Englewood Cliffs, New Jersey, 1975.

TABLE 2

APL/Z80 SCALAR FUNCTIONS

| MONADIC FORM | | | F | DYADIC FORM | | |
|---|---|---|---|---|---|---|
| SYNTAX | NAME | NOTES | | SYNTAX | NAME | NOTES |
| + A | CONJUGATE | | + | A + B | PLUS | |
| - A | NEGATIVE | | - | A - B | MINUS | |
| × A | SIGNUM | | × | A × B | TIMES | |
| ÷ A | RECRIPROCAL | | ÷ | A ÷ B | DIVIDE | |
| \| A | ABSOLUTE VALUE | | \| | A \| B | RESIDUE | |
| ⌊ A | FLOOR | | ⌊ | A ⌊ B | MINIMUM | |
| ⌈ A | CEILING | | ⌈ | A ⌈ B | MAXIMUM | |
| ? A | ROLL | 1 | ? | A ? B | DEAL | 1 |
| * A | EXPONENTIAL | | * | A * B | POWER | |
| ⍟ A | NATURAL LOG | | ⍟ | A ⍟ B | LOGARITHM | |
| ○ A | PI TIMES | | ○ | A ○ B | CIRCULAR | 2 |
| ! A | FACTORIAL | 3 | ! | A ! B | BINOMIAL | |
| ~ A | NOT | | ~ | | | |
| | | | ∧ | A ∧ B | AND | |
| | | | ∨ | A ∨ B | OR | |
| | | | ⍲ | A ⍲ B | NAND | |
| | | | ⍱ | A ⍱ B | NOR | |
| | | | < | A < B | LESS | |
| | | | ≤ | A ≤ B | NOT GREATER | |
| | | | = | A = B | EQUAL | |
| | | | ≥ | A ≥ B | NOT LESS | |
| | | | > | A > B | GREATER | |
| | | | ≠ | A ≠ B | NOT EQUAL | |

NOTES:

1. (A,B)∈⍳65535
2. A∈0 1 2 3 4
3. GAMMA FUNCTION NOT IMPLEMENTED

TABLE 1

APL/Z80 CHARACTER SET

| | | | |
|---|---|---|---|
| A B C D E F G H I J K L M N O P Q R S T U V W X Y Z | | | |
| 0 1 2 3 4 5 6 7 8 9 | | | |
| ¨ DIERESIS | ⍺ ALPHA | ⍱ NOR | ~ ∨ |
| ¯ OVERBAR | ⌈ UPSTILE | ⍲ NAND | ~ ∧ |
| < LESS | ⌊ DOWNSTILE | ⍒ DEL STILE | ∇ \| |
| ≤ NOT GREATER | _ UNDERBAR | ⍋ DELTA STILE | Δ \| |
| = EQUAL | ∇ DEL | ⍎ CIRCLE STILE | ○ \| |
| ≥ NOT LESS | Δ DELTA | | |
| > GREATER | ∘ NULL | | |
| ≠ NOT EQUAL | ' QUOTE | ⍟ LOG | * ○ |
| ∨ OR | ⍞ QUAD | | |
| ∧ AND | ( OPEN PAREN | ⍕ DEL TILDE | ∇ ~ |
| - BAR | ) CLOSE PAREN | ⍤ BASE NULL | ⊥ ∘ |
| ÷ DIVIDE | [ OPEN BRACKET | ⍒ TOP NULL | ⊤ ∘ |
| + PLUS | ] CLOSE BRACKET | ← LEFT ARROW | |
| × TIMES | ⊂ OPEN SHOE | / SLASH | |
| ? QUERY | ⊃ CLOSE SHOE | SPACE | |
| ⍵ OMEGA | ∩ CAP | ⍝ CAP NULL | ∩ ∘ |
| ∈ EPSILON | ∪ CUP | ⍞ QUOTE QUAD | ' ⍞ |
| ⍴ RHO | ⊥ BASE | ! QUOTE DOT | ' . |
| ~ TILDE | ⊤ TOP | | |
| ↑ UP ARROW | \| STILE | ↓ DOWN ARROW | |
| ⍳ IOTA | ; SEMICOLON | : COLON | |
| ○ CIRCLE | , COMMA | * STAR | |
| → RIGHT ARROW | . DOT | \ SLOPE | |

TABLE 4

APL/Z80 OPERATORS

| OPERATOR | SYNTAX | |
|---|---|---|
| REDUCTION | S / A | |
| SCAN | S \ V | |
| OUTER PRODUCT | A ∘.S A | |
| AXIS | O [N] A | FOR REDUCTION AND SCAN, AXIS ALSO APPLIES TO CERTAIN MIXED FUNCTIONS. |

S - SCALAR DYADIC FUNCTION

A - ARRAY

O - OPERATOR

N - AXIS

V - SCALAR OR VECTOR

A FUNCTION FOR INNERPRODUCTS IS GIVEN IN THE WSFNS WORKSPACE

TABLE 3

APL/Z80 MIXED FUNCTIONS

| MONADIC FORM | | | DYADIC FORM | | | |
|---|---|---|---|---|---|---|
| SYNTAX | NAME | NOTES | F | SYNTAX | NAME | NOTES |
| ⍴ A | SHAPE | | ⍴ | V ⍴ A | RESHAPE | |
| , A | RAVEL | | , | A , A | CATENATE | 1 |
| ⌽ A | REVERSAL | 2 | ⌽ | S ⌽ V | ROTATE | 2 |
| | | | ↑ | S ↑ V | TAKE | 2 |
| | | | ↓ | S ↓ V | DROP | 2 |
| | | | / | V / A | COMPRESS | |
| | | | \ | V \ A | EXPAND | |
| | | | [] | A[V;..;V] | INDEXING | |
| ⍳ S | INDEX GENERATOR | | ⍳ | V ⍳ A | INDEX OF | |
| | | | ∈ | A ∈ A | MEMBERSHIP | |
| ⍋ V | GRADE UP | | ⍋ | | | |
| ⍒ V | GRADE DOWN | | ⍒ | | | |
| | | | ⊥ | V ⊥ V | DECODE | 3 |
| | | | ⊤ | V ⊤ S | ENCODE | 5 |
| ⍎ V | EXECUTE | 6 | ⍎ | | | |
| | | | ⍕ | V ⍕ A | FORMAT | 4 |

NOTATIONS: S-SCALAR, V-VECTOR, A-ARRAY (RANK-8 MAX)

1. LAMINATION NOT IMPLEMENTED
2. RIGHT ARGUMENT RESTRICTED TO RANK 0 OR 1
3. ARGUMENTS RESTRICTED TO RANK 0 OR 1
4. V RESTRICTED TO TWO ELEMENTS
5. RIGHT ARGUMENT RESTRICTED TO 1 ELEMENT OF RANK 0 OR 1
6. V RESTRICTED TO 120 ELEMENTS MAXIMUM

TABLE 6

APL/Z80 SYSTEM VARIABLES

| NAME | FUNCTION | VALUE IN CLEAR WS | RANGE |
|---|---|---|---|
| ⎕O | INDEX ORIGIN | 1 | 0 1 |
| ⎕P | PRINT PRECISION | 8 | ⍳10 |
| ⎕W | PRINT WIDTH | 64-VIDEO<br>120-SERIAL | 24+⍳231 |
| ⎕L | RANDOM LINK | 34952 | ⍳65535 |
| ⎕C | LINE COUNTER | | |
| ⎕A | WORKSPACE AVAILABLE | | |
| ⎕H | HARD COPY (1=ON) | | 0 1 |
| ⎕V | ATOMIC VECTOR | | |
| ⎕U | SPECIFY ACTIVE DISK UNIT | | ⍳254 |
| ⎕X | LATENT EXPRESSION | ⎕V[1] | CHARACTER |

TABLE 5 APL/Z80 SYSTEM COMMANDS

| COMMAND | FUNCTION | |
|---|---|---|
| CONSOLE CONTROL | | |
| )OFF | RETURN TO HOST OPERATING SYSTEM | |
| ACTIVE WORKSPACE | | |
| )CLEAR | FURNISH CLEAR WORKSPACE | |
| )COPY COPYOBJECT | COPY A COPYOBJECT INTO WORKSPACE | |
| )ERASE NAMES | ERASE OBJECTS | |
| )LOAD WSID | ACTIVATE A COPY OF NAMED WS | |
| )WSID [WSID] | CHANGE OR LIST ACTIVE WS NAME | |
| LIBRARY CONTROL | | |
| )DROP WSID | DROP A WORKSPACE FROM LIBRARY | |
| )SAVE [WSID] | SAVE NAMED WORKSPACE IN LIBRARY | |
| )CDROP COPYOBJECT | DROP A COPYOBJECT FROM LIBRARY | |
| )CSAVE COPYOBJECT | SAVE NAMED COPYOBJECT IN LIBRARY | |
| SYSTEM PARAMETERS | | |
| )WIDTH N | SET PRINT WIDTH | $25 \le N \le 255$ |
| )DIGITS N | SET PRINT PRECISION | $1 \le N \le 10$ |
| )SYMBOLS N | SET SYMBOL TABLE SIZE | $0 \le N \le 503$ |
| )ORIGIN N | SET INDEX ORIGIN | $N \in 0\ 1$ |
| INQUIRY | | |
| )FNS [ALPHA] | LIST FUNCTIONS IN ACTIVE WS | |
| )VARS [ALPHA] | LIST VARIABLES IN ACTIVE WS | |
| )SVARS [ALPHA] | LIST SHARED VARS IN ACTIVE WS | |
| )SI | LIST HALTED FUNCTIONS | |
| )SIV | LIST HALTED FUNCTIONS AND VARS | |
| )LIB | LIST LIBRARY ON ACTIVE DISK UNIT | |
| )CLIB | LIST COPYOBJECTS ON ACTIVE DISK | |

TABLE 8

APL/Z80 ERROR MESSAGES

| TYPE | CAUSE |
|---|---|
| DEFN | MISUSE OF ∇ |
| DOMAIN | ARGUMENTS NOT IN DOMAIN OF FUNCTION |
| INDEX | INDEX VALUE OUT OF RANGE |
| INTERRUPT | EXECUTION WAS SUSPENDED WITHIN AN EXPRESSION |
| LENGTH | SHAPES NOT CONFORMABLE |
| RANK | RANKS NOT CONFORMABLE |
| SYNTAX | INVALID SYNTAX |
| SYMBOL TABLE FULL | TOO MANY NAMES IN USE |
| SYSTEM | INTERNAL SYSTEM PROBLEMS |
| VALUE | USE OF A NAME WHOSE VALUE OR DEFINITION HAS NOT BEEN SPECIFIED |
| WS FULL | WORKSPACE IS FILLED |
| NONCE | NOT IMPLEMENTED IN THIS VERSION |
| NO SPACE | AFTER )SAVE NO DISK SPACE<br>AFTER )COPY OBJECT TOO LARGE |
| I/O ERROR | AN I/O ERROR HAS OCCURRED |
| NOT FOUND | OBJECT OR WORKSPACE NOT FOUND IN LIBRARY. |
| INCORRECT COMMAND | IMPROPERLY FORMED COMMAND |
| NOT WITH SI | SYSTEM COMMAND NOT PERMITTED WITH SI |
| NOT WITH OPEN DEFN | SYSTEM COMMAND NOT PERMITTED WITH OPEN DEFINITION |

TABLE 7

APL/Z80 SYSTEM FUNCTIONS

| SYSTEM FUNCTION | PURPOSE |
|---|---|
| P □S C | INITIATE, OR OFFER, A SHARE |
| P □R C | RETRACT A SHARE |
| □K N | CANONICAL REPRESENTATION |
| □F R | FIX |

P - PROCESSOR ID NUMBER (SCALAR INTEGER VALUE)

C - RANK-0 OR-1 CHARACTER STRING WHICH IS THE NAME OF A VARIABLE.

N - RANK-0 OR-1 CHARACTER STRING WHICH IS THE NAME OF A FUNCTION.

R - RANK-1 CHARACTER STRING WHICH IS A CANONICAL REPRESENTATION OF A FUNCTION.