Trump Card

Part 1: Hardware

Speed up your IBM PC with 16-bit coprocessing power

Steve Ciarcia
Consulting Editor

When asked what computer language I prefer, I generally reply, "Solder." This response is not an effort to be cute but rather to express a preference for dealing in the terms I know best. I don't avoid software. I just try to minimize my involvement.

When it is necessary to write simulation and test programs, I bite the bullet. Unless the function is time-critical, I most often choose BASIC because it comes closest to being a universal programming language. Virtually all personal and business computers support it, and if I confine my command choices to the more common instructions, the demonstration programs that I compose on an IBM PC should also run on your Cromemco Z2.

With few exceptions, you can compute your accounts receivable or type in and play a game equally well with an Apple or IBM PC using BASIC. The fact that one has a 6502 microprocessor and the other uses an 8088 is irrelevant. The output will be the same.

The value of high-level languages is that they isolate the user from microprocessor peculiarities and facilitate transportable software. Unfortunately, the average ROM (read-only memory)-resident BASIC interpreter was never written with perfor-

Copyright © 1984 Steven A. Ciarcia.
All rights reserved.

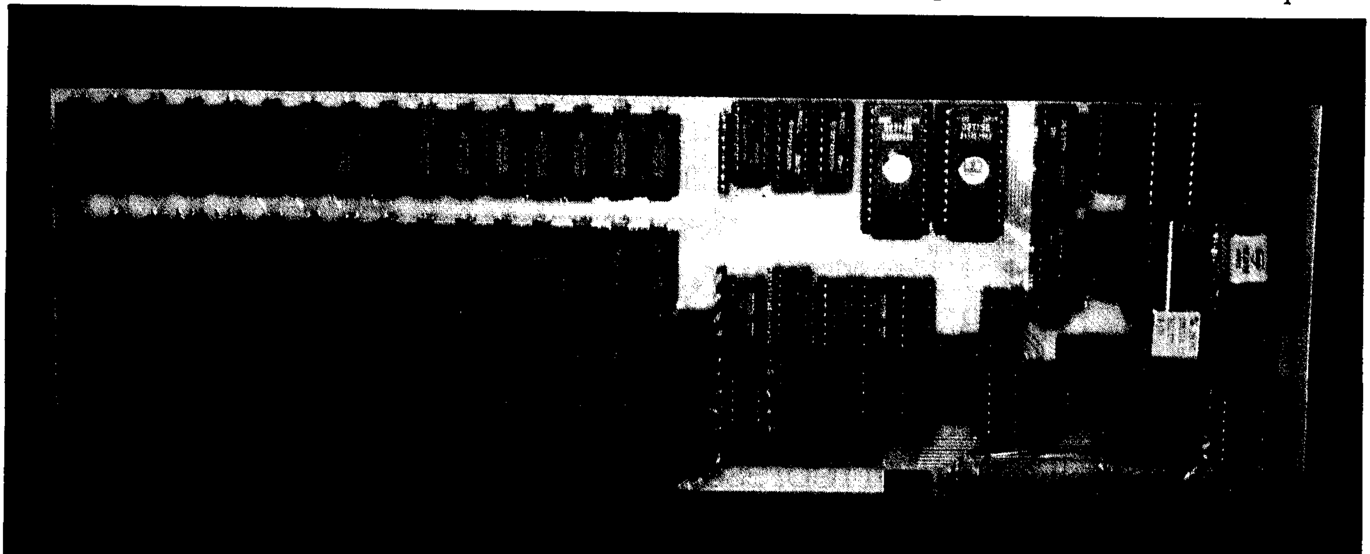


Photo 1: *The wire-wrapped prototype of the Trump Card, shown from the front. The left side of the board contains 512K bytes of type-4164 dynamic RAM; the right side contains the Zilog Z8001 and an interface to the IBM PC I/O-expansion bus.*

mance in mind. Usually taking 5 to 10 milliseconds (ms) to execute an individual instruction, it can seem like forever when running long programs.

As a writer, I have grown to appreciate the universality of BASIC, even with its shortcomings. By treating the computer as a black box with I/O (input/output) ports and BASIC, I have been able to provide projects that can be implemented on most systems directly. As an engineer/designer, however, I am aggravated by its slowness and feel no animosity toward critics who have converted to languages such as Pascal or C to gain processing speed.

Rather than make further excuses, I decided to solve the problem in classic Circuit Cellar tradition—simply build a black box that improves system throughput and runs BASIC programs faster.

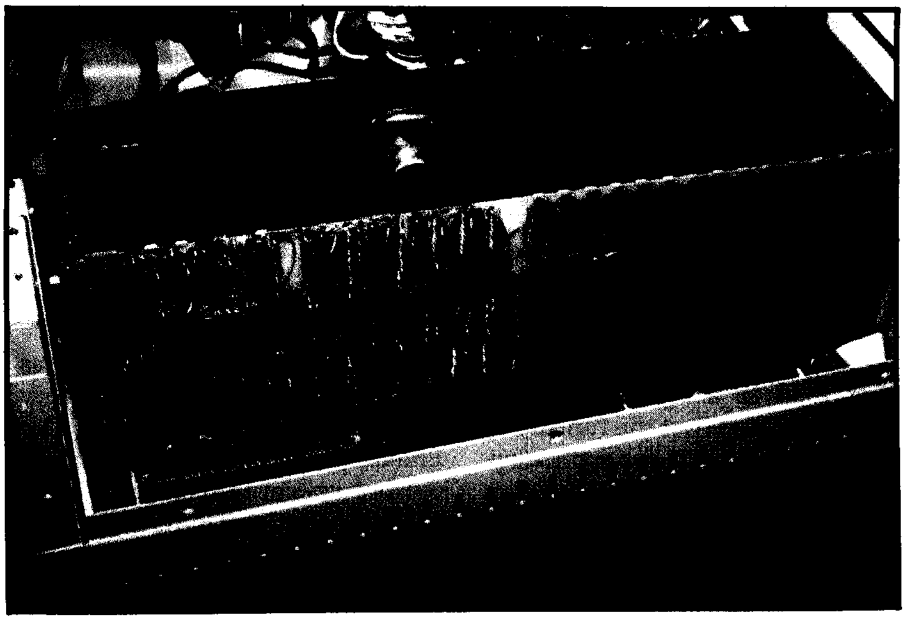


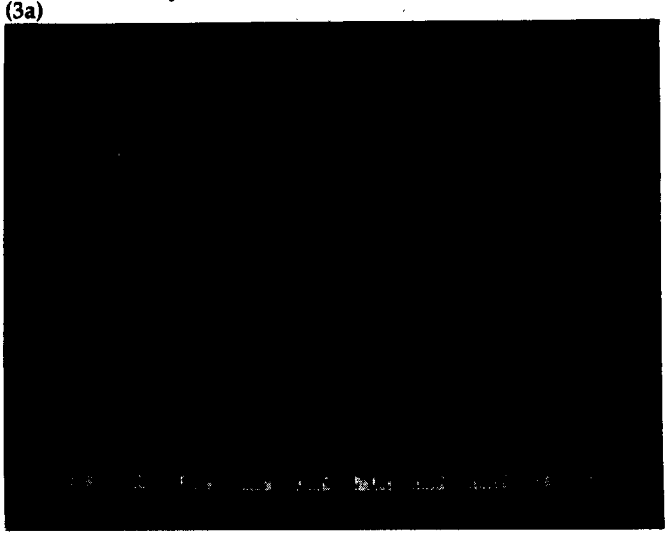
Photo 2: *The rear of the Trump Card prototype. To save time, the memory section was laid out as a printed-circuit board, with wire-wrapping saved for the processor side. As shown here, the Trump Card is installed for testing in an MPX-16 computer, which has I/O slots compatible with the IBM PC.*

Processors and Performance

Generally speaking, most people confuse microprocessor benchmarks with system throughput. The comparison of microprocessor-instruction execution speeds is not really indicative of a computer's capabilities. Performance is more often governed by the operating system and magnitude of the application program. It is a false assumption that all software written for a 16-bit microprocessor will necessarily run faster than on an 8-bit microprocessor. Machine-language fast Fourier transforms (FFT) run quickly on a 6502, but an accounting package that has to constantly interleave a program into and out of disk may be encumbered by 64K bytes of operational memory in the Apple. In all likelihood, large spreadsheets and accounting programs will run more efficiently in the larger memory space provided on an 8088 system such as the IBM PC.

Raising the performance of a high-level language such as BASIC takes more than raising a microprocessor's clock rate. Instead, it involves a combination of decisions that can ultimately affect the entire system throughput. We can expand the memory available to application programs in an effort to limit repeated disk accesses and configure a portion of memory as a RAM (random-access read/write memory) disk drive to expedite disk operations when they are required. We can optimize the effi-

(3a)



(3b)

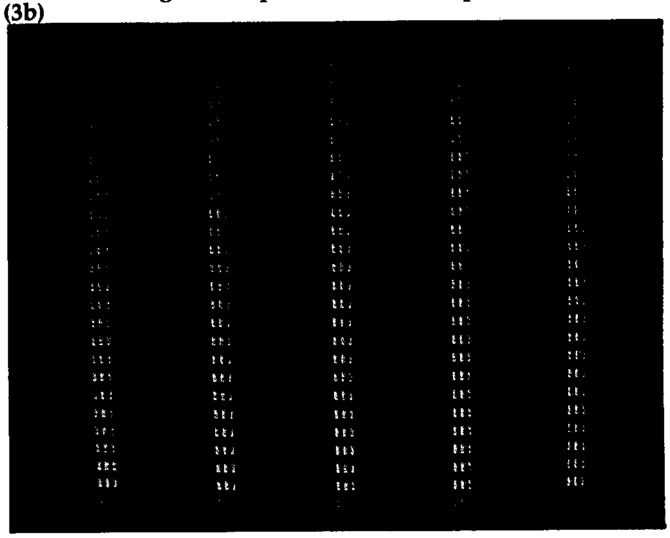


Photo 3: *Execution-time visual comparison. (3a) Without Trump Card—a two-second exposure of the display while running the BASICA program in listing 1. The program has executed the PRINT statement and still is dimensioning the arrays. (3b) With Trump Card—the same two-second exposure of the program execution (with PRINT statement added) shows the arrays have been dimensioned and the prime numbers are being overprinted so fast that they blur.*

Apple II	Apple III	TRS-80 Model II	IBM PC	IBM PC (with Trump Card)
224	222	189	190	2.4

Table 1: *A comparison of execution times (in seconds) of the benchmark program in listing 1.*

Listing 1: *Sieve of Eratosthenes prime-number-generator program.*

```
5 DEFINT A-Z
10 SIZE = 8190
20 DIM FLAGS(8191)
30 PRINT "Only 1 iteration"
50 COUNT = 0
60 FOR I = 0 TO SIZE
70 FLAGS(I) = 1
80 NEXT I
90 FOR I = 0 TO SIZE
100 IF FLAGS(I) = 0 THEN 180
110 PRIME = I+I +3
120 K = I + PRIME
130 IF K> SIZE THEN 170
140 FLAGS(K) = 0
150 K = K + PRIME
160 GOTO 130
170 COUNT = COUNT + 1
180 NEXT I
190 PRINT COUNT, " PRIMES"
```

ciency of the high-level language by operating it in a compiled mode rather than as a repeatedly interpreted task. Finally, if the functional throughput of a particular application becomes dependent upon direct microprocessor intervention, for those tasks, substitute a faster microprocessor or help it with a coprocessor.

A Black Box Called Trump Card

This article is not about building a classic speed-up board for the IBM PC. The word "speed-up" implies replacing the 8088 with an 8086 or 80186. Instead, visualize your PC as a black box with an input, output, and crank. Rather than simply turning the crank faster, think of adding another black box, in the same path between input and output, that performs selective tasks more efficiently and faster than the 8088 alone. To increase the relative throughput of the system, I have designated an alternate path for specific program functions.

I've named this separate box Trump Card. It is a functionally independent 10-MHz Zilog Z8001-based computer with its own 512K bytes of memory. Designed specifically as a compiled high-level-language computer, Trump Card is addressed as an I/O device that communicates through the expansion bus (see photos 1 and 2).

Among the specific functions that Trump Card supports are BASIC, C, CP/M-80, text editing, Z8000 assembly-language programming, and a RAM disk. It does not directly execute programs written in 8088 assembly code, such as Lotus 1-2-3. It instead executes programs written in high-level languages such as BASIC or C (a Pascal compiler and 8088-to-Z8000 translator are in the works). Alternatively, it can enhance the function of programs such as 1-2-3 by expanding available memory and speeding disk functions. The ultimate purpose of Trump Card is to improve system throughput.

This month, I will outline the basic functions of Trump Card and describe its hardware in detail. This is, of course, a Circuit Cellar construction project, and you are encouraged to build your own Trump Card. More on that later. Next month, I'll describe some of the software in detail and do a little benchmarking.

First, a little about Trump Card and the Z8001.

Trump Card

Trump Card is a peripheral board that plugs into any expansion slot on an IBM PC or PC-compatible computer. It contains a 10-MHz Z8001 and up to 512K bytes of memory. To use it, you simply load a BASIC, CP/M-80, or C program from PC-DOS and type "RUN." Its memory can also be used as a RAM disk.

Trump Card comes with software that translates existing BASIC and other high-level-language programs to run with reduced overhead. To speed the execution of BASICA, Trump Card compiles the code with a special version of BASIC called TBASIC. Unlike other compilers, this has no separate compiled-code disk files (unless you specifically want them) and no long delays. TBASIC instantly compiles the program in a few tenths of a second when you load the file into Trump Card. In appearance, it looks like any old, slow interpreted BASIC, but it runs with the speed of a compiler.

TBASIC is PC BASICA-compatible. You can use either the Trump Card screen editor or BASICA's editor to write your programs. Then run the same program using either Trump Card or BASICA. Depending upon the instructions you use, Trump Card provides a tenfold to hundredfold increase in program performance (see photo 3). Table 1 shows typical results of what Trump Card can do with the prime-number Sieve of Eratosthenes program (September 1981 BYTE, page 180) frequently used to benchmark computer systems (see listing 1).

Though I conceived of Trump Card initially as a BASICA enhancement, it didn't take me long to realize that a Z8001 with 512K bytes of memory has some real computing power and deserves proper support. For that reason, the software supplied with this project is much more extensive than usual. With the utilities and languages included, you should have little trouble using the vast software base of Z80 and Z8000 programs.

Trump Card includes the following software:

BASIC Compiler—TBASIC is PC BASICA-compatible. The differences between the BASICA interpreter and the TBASIC compiler are minimal. Most instructions are implemented without modification.

CP/M-80 Emulator—Trump Card can run your CP/M-80 Z80 assembly-language programs directly without special disk headers or translation programs. Simply download your Z80 programs and run them.

C-Compiler—Trump Card includes the industry standard version of C that is described in *The C Programming Language* by Kernighan and Ritchie.

Debugger—Intended to aid in program development. With it, you can examine and replace memory and register contents, set breakpoints, or single-step through programs.

Screen Editor—Incorporating many of the features included in word processors, the editor enables you to write or examine ASCII text files for either the PC or Trump Card's use.

Multilevel Language Compiler—This is a structured assembler that allows Pascal-like control and data types, arithmetic expressions with automatic or specified allocations of registers, and procedure calls with parameter passing.

RAM Disk—Trump Card can allocate 128K to 387K bytes of its on-board memory to function as an intelligent RAM disk (DOS 2.0 only). This memory is separate from and in addition to any already existing on the PC bus. Trump Card's other functions can run concurrently.

The Z8000 Microprocessor

A block diagram of the Z8000's internal structure appears here as figure 1. As the programmer sees it, the Z8000 contains sixteen 16-bit general-purpose registers (for addresses or data) that may also be used in groups to form as many as eight 32-bit registers or four 64-bit registers. The low-order halves of the registers may be used for byte operations, thus the Z8000 is able to manipulate data in 8-, 16-, 32-, and 64-bit pieces.

The eight addressing modes are register, indirect-register, direct-address, indexed, immediate, base-address, base-indexed, and relative-address. The instruction set utilizes data types ranging from single bits to a 32-bit-long word. The processor executes 110 distinct instruction types that, when permuted by all the addressing modes and data types, create a set of more than 400 instructions.

The Z8000 has two different modes of operation: system and normal. Which mode of operation is in effect is controlled by a bit in the flag-and-control word (FCW). The main difference between the operating modes is that some of the control/interrupt and I/O instructions work only in the system mode. To simplify the design of the Trump Card, I chose to use only the system mode.

The Z8001 (see photo 4) is the memory-segmented version of Zilog's chip; it comes in a 48-pin DIP (dual-

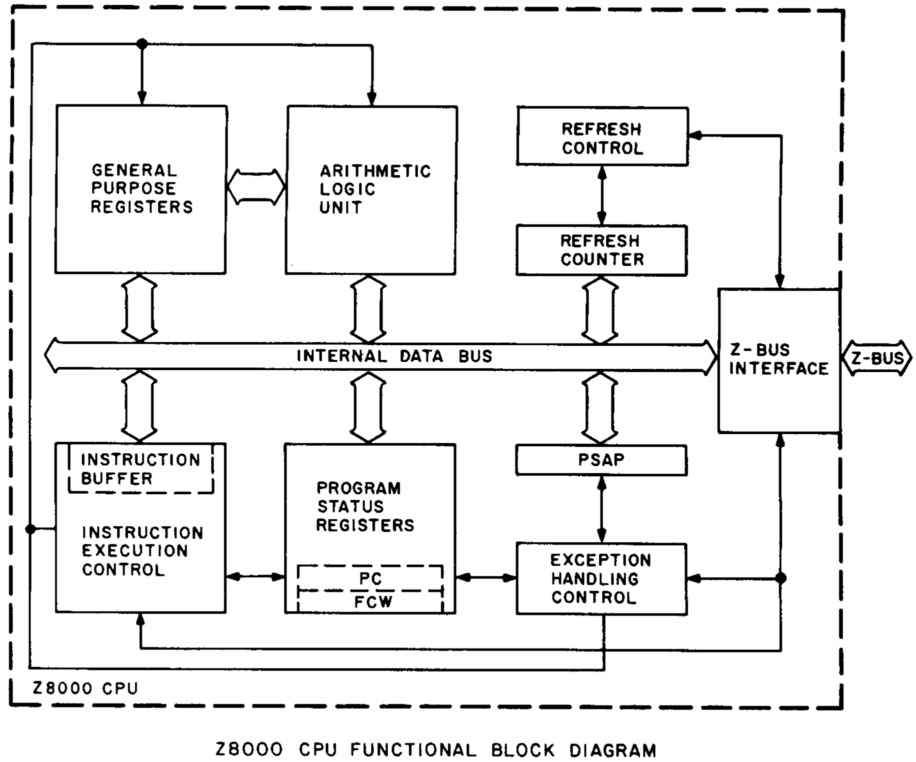


Figure 1: *Block diagram of the internal structure of the Zilog Z8000 family of microprocessors.*

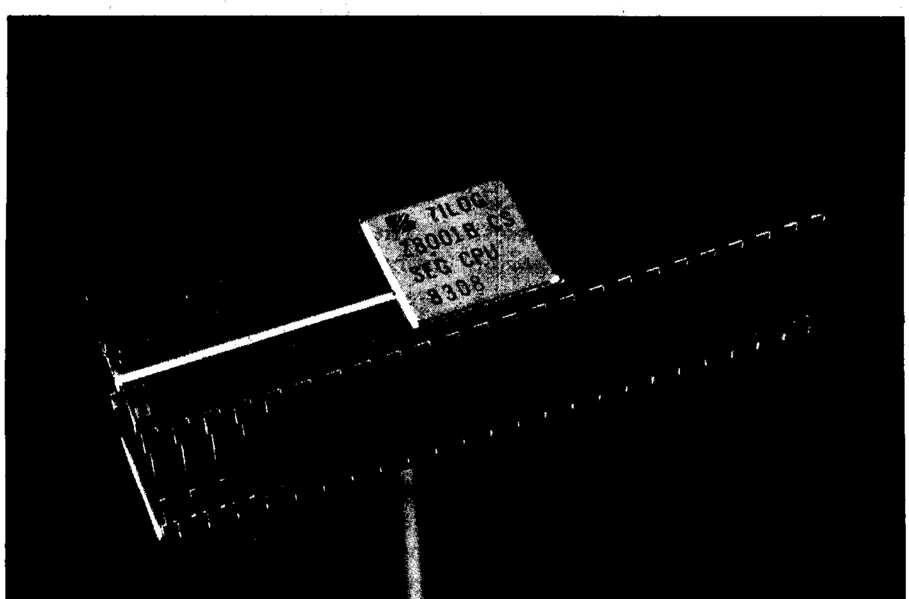


Photo 4: *The 48-pin dual-inline package that houses the Zilog Z8001 microprocessor, the heart of the Trump Card.*

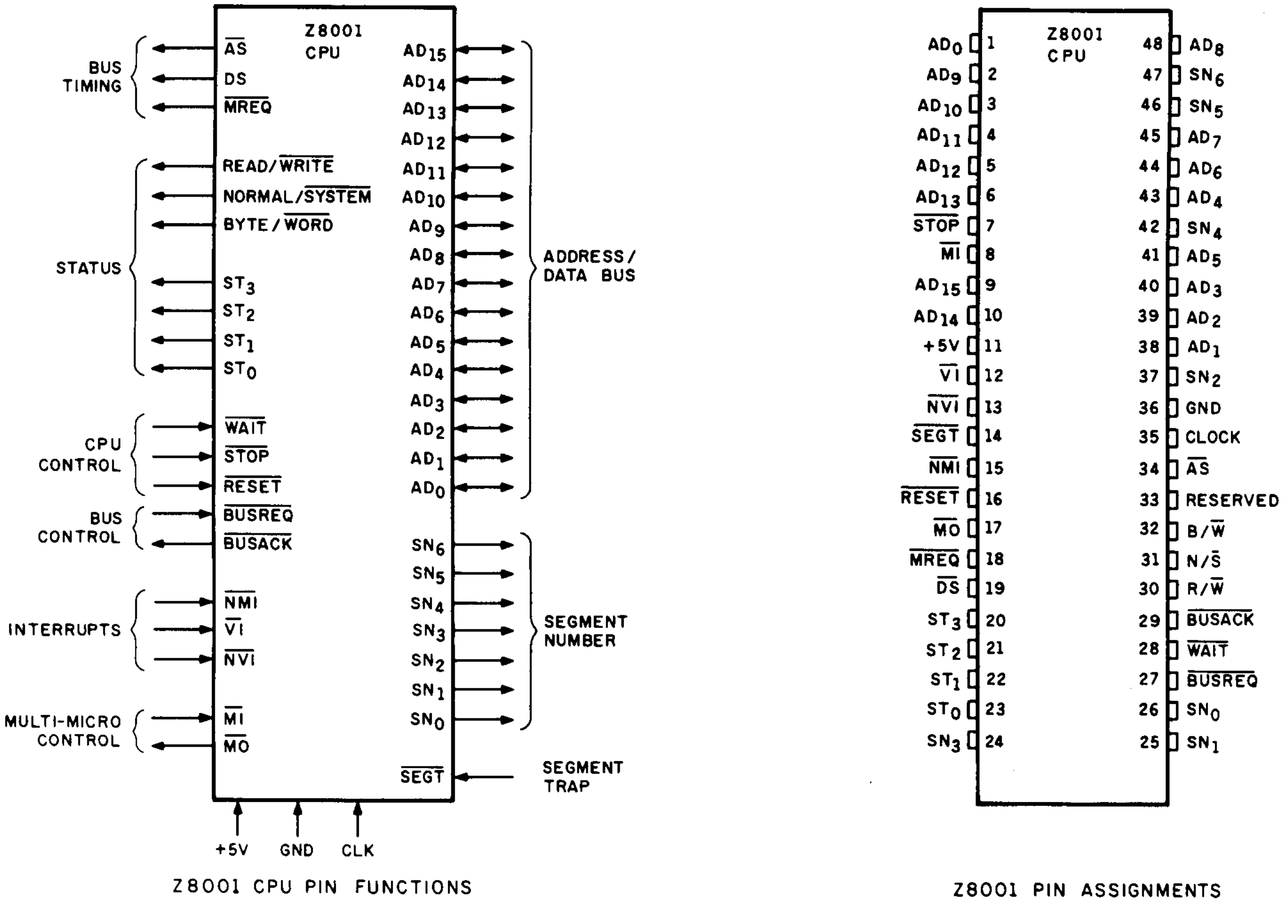


Figure 2: *Pinout arrangement of the Z8001 memory-segmented version.*

inline package), the pinouts of which are shown in figure 2. (The nonsegmented 40-pin version is called the Z8002.) By memory segmentation, the directly addressable 8-megabyte memory space is divided into as many as one hundred twenty-eight 64K-byte regions. Seven segment-selection lines coming out of the Z8001 control the high-order memory addressing. When the Z8001 is reset, the segment addressing automatically reverts to segment 0, the lowest 64K-byte block of memory. Transfer of control between segments is done by jumps, calls, and returns.

Inside the Trump Card

The schematic diagram of figure 3 shows the Trump Card's circuitry. It can be plugged into any expansion slot of an IBM PC or into any other computer with compatible I/O slots and operating system.

Five of the Z8001's seven segment-selection lines, SN0 through SN4, are used in the Trump Card to decode addresses for up to 1 megabyte of RAM (512K bytes fit on the board) and 4K bytes of ROM (read-only memory). Segment line 4 selects between the ROM, mapped into segments 0 through 15, and the RAM, residing in segments 16 through 31. The states of the segment lines are latched by IC3; segment line 4 is named RAM/$\overline{\text{PROM}}$.

Address/Data Bus

The address/data lines coming from the Z8001 (AD0 through AD15) are a time-multiplexed address and data bus, which can address a range of 65,536 (64K) bytes of memory or a like number of I/O addresses. Since the Z8001 can form addresses at either word or byte boundaries, the least significant bit AD0 is used in byte operations to determine if the upper or lower byte is to be operated upon. The address on the AD lines becomes valid when the Z8001 asserts the $\overline{\text{AS}}$ (address strobe—active low) line; it remains that way for a short hold time after $\overline{\text{AS}}$ returns to its idle high state. The address from the Z8001 is latched by two type-74LS373 transparent latches, IC5 and IC6, that are always enabled. The use of transparent latches allows for maximum address-setup time to the memories.

The latched addresses (LA0 through LA15) come out of the 74LS373s with LA0 combined with the signal B/$\overline{\text{W}}$ (byte or word address) to form the $\overline{\text{EVEN}}$ or $\overline{\text{ODD}}$ byte-bank-select signal for memory. When B/$\overline{\text{W}}$ is low, it signifies that a 16-bit memory word is being referenced; this causes the outputs of the two AND gates at IC8 pin 3 and IC8 pin 6 to be active irrespective of the state of LA0. By doing byte operations in this manner, it is possible for the Z8001 to do single-byte memory writes without first reading an entire word location.

The Trump Card contains a pair of type-2716 EPROMs (erasable programmable read-only memories),

Text continued on page 50

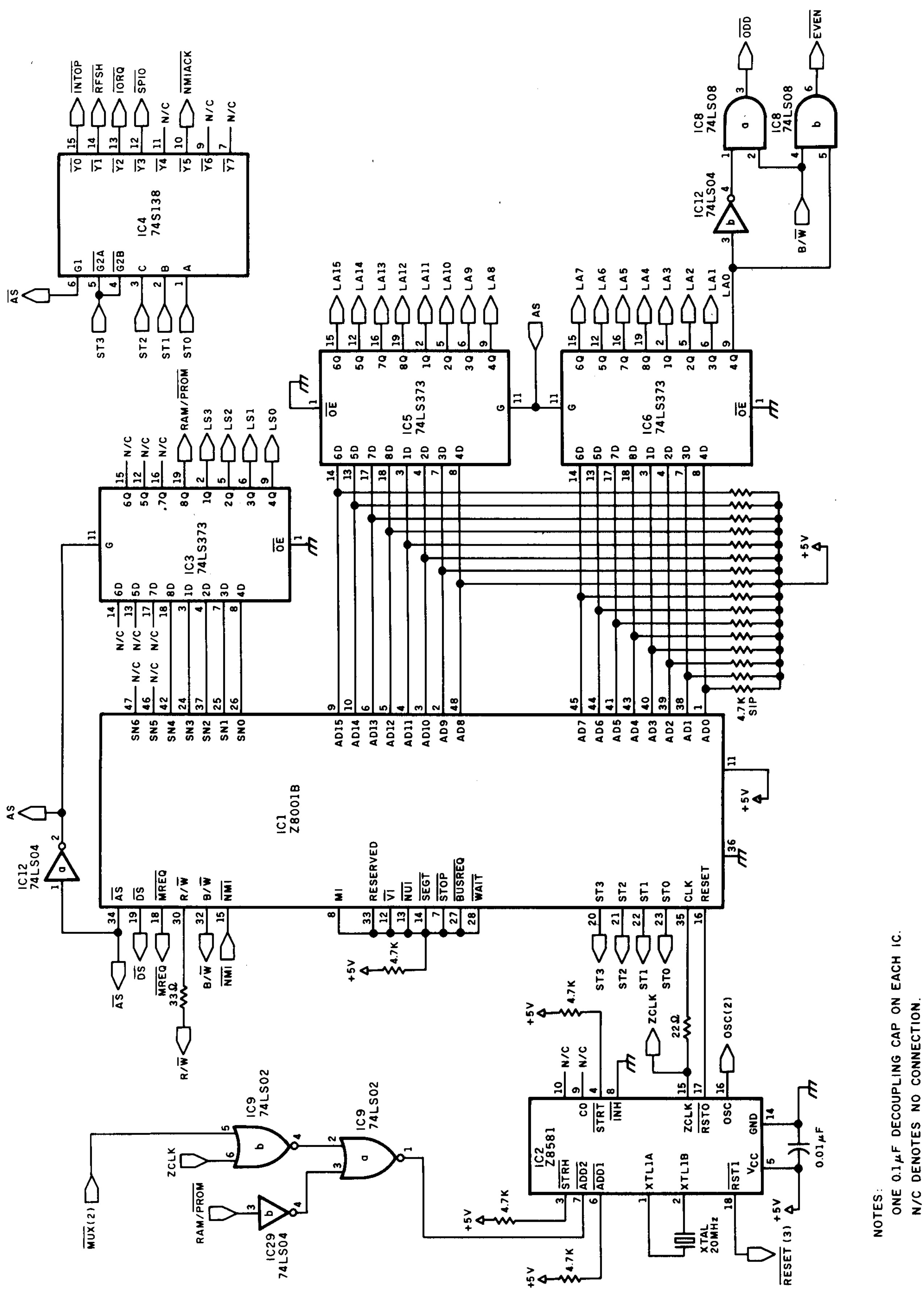


Figure 3 continued on page 46

Figure 3: *Schematic diagram of the Trump Card. Support is provided for 512K bytes of dynamic RAM in the form of type-4164 chips. If a 6-MHz Z8001A is used, the crystal frequency must be reduced to 12 MHz.*

Figure 3 continued:

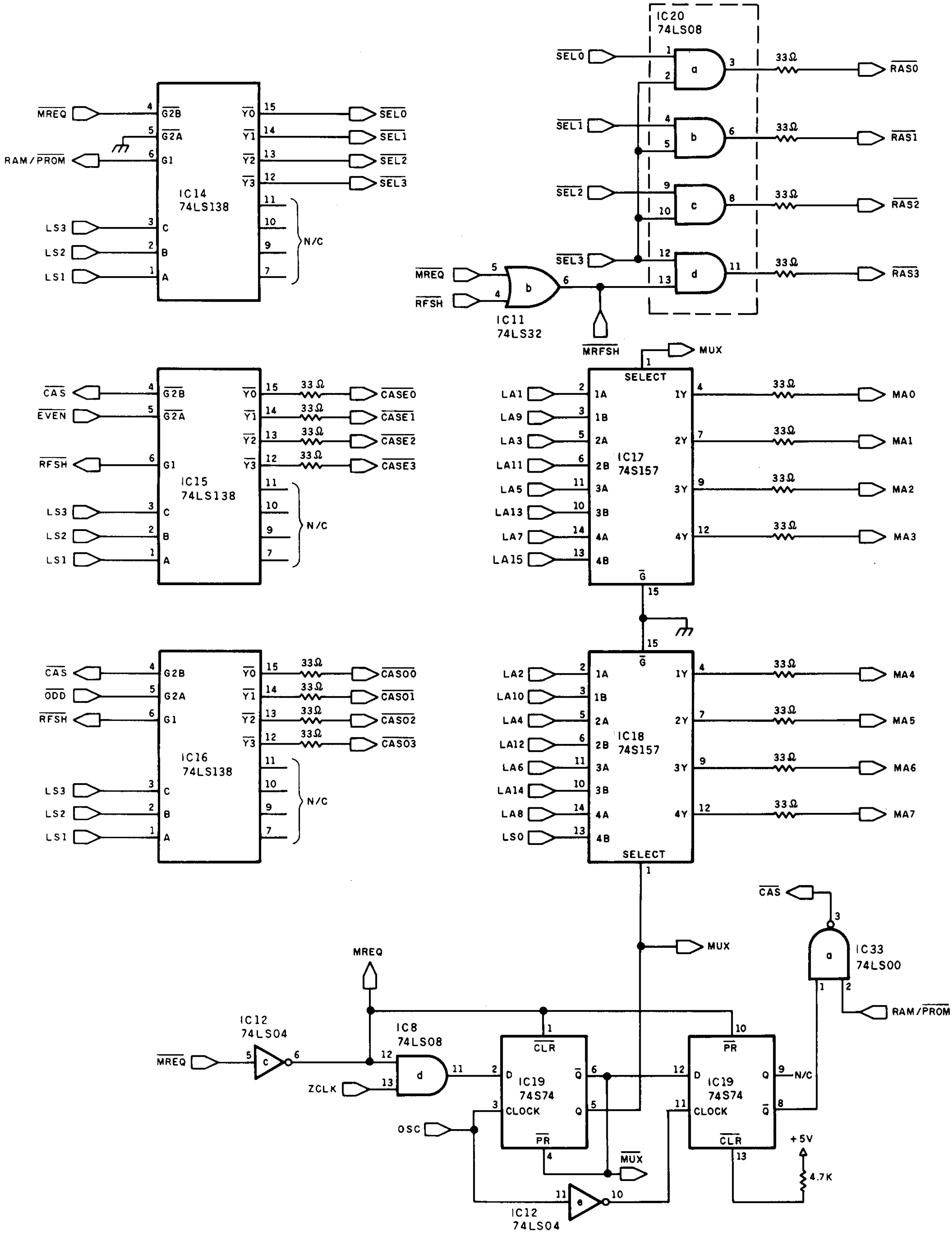


Figure 3 continued on page 47

Figure 3 continued:

NOTE: IC's IN THIS SECTION ARE CONSECUTIVELY NUMBERED FROM IC36 UPPER LEFT CORNER TO IC99 LOWER RIGHT CORNER. IC's ARE ALL 4164's.

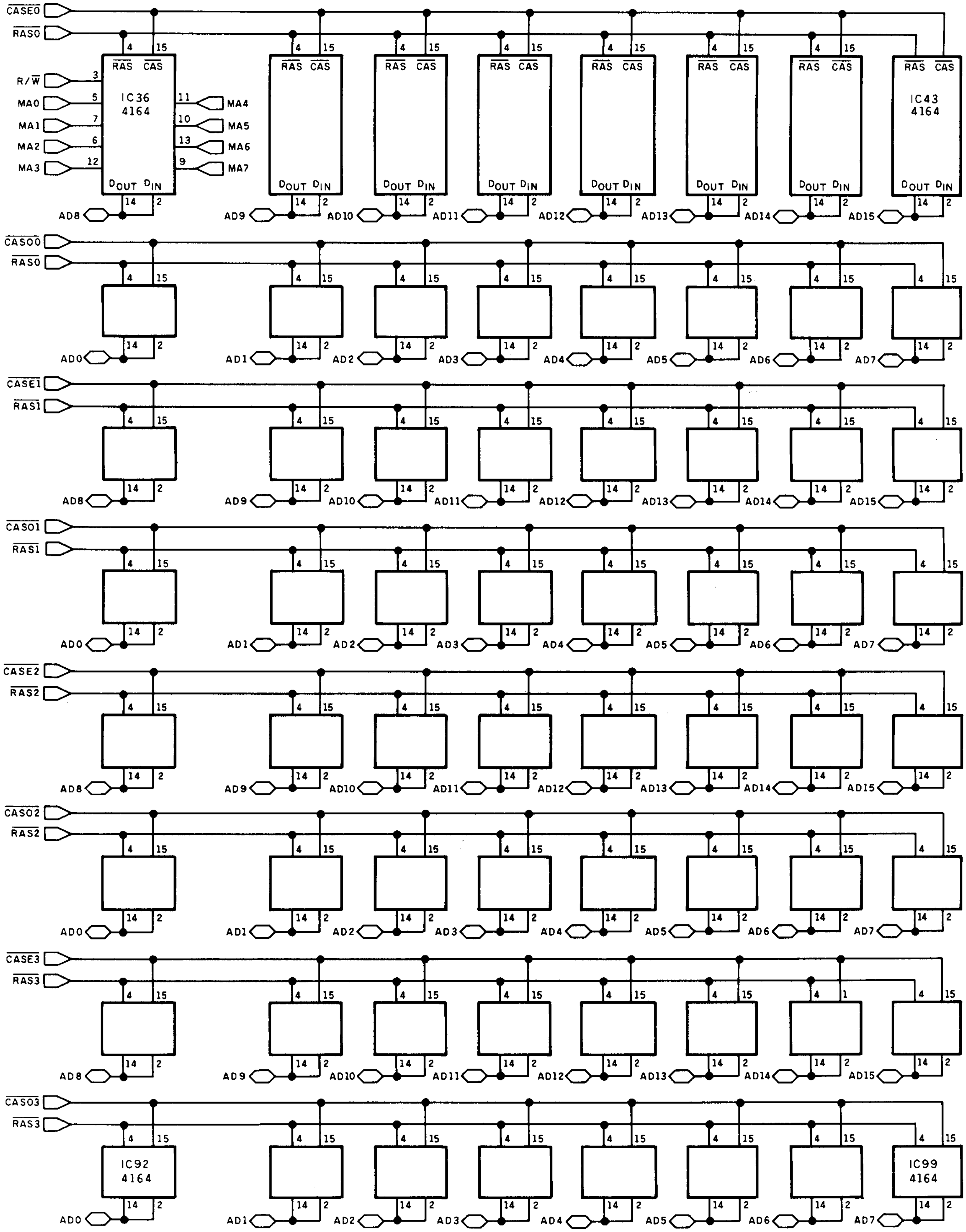


Figure 3 continued on page 48

Figure 3 continued:

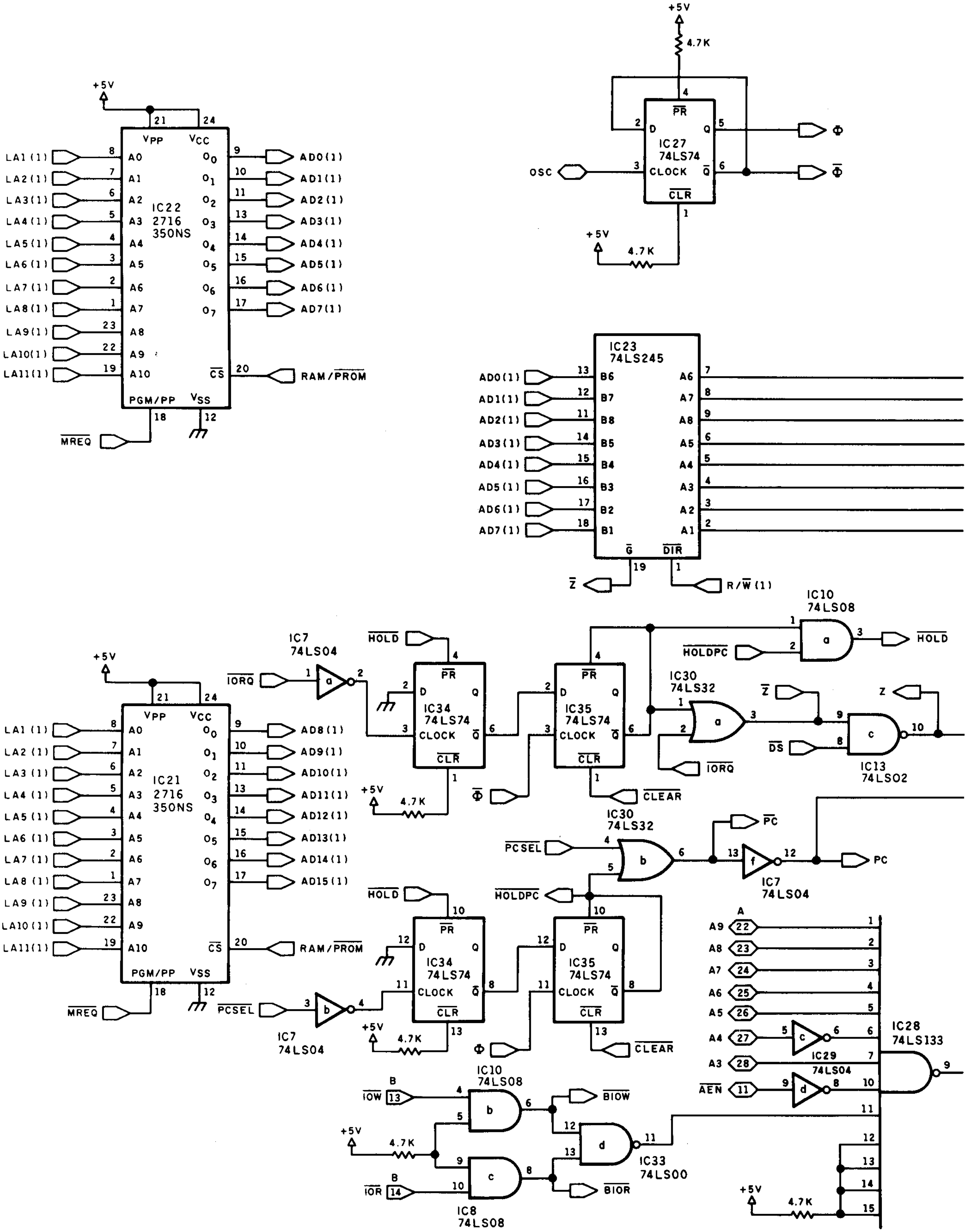
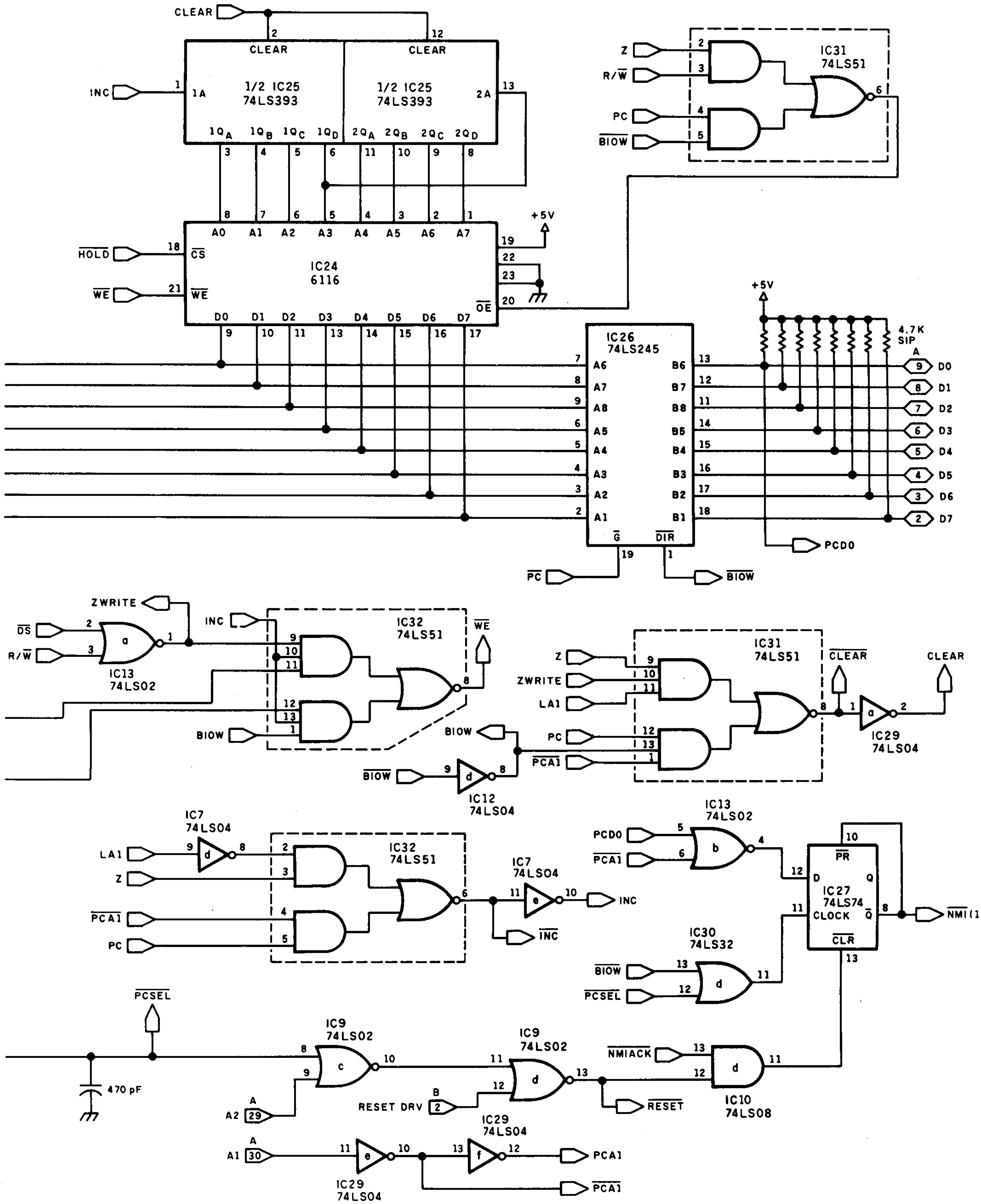


Figure 3 continued on page 49

Figure 3 continued:



Number	Type	+5 V	GND
IC1	Z8001B	11	36
IC2	Z8581	5	14
IC3	74LS373	20	10
IC4	74S138	16	8
IC5	74LS373	20	10
IC6	74LS373	20	10
IC7	74LS04	14	7
IC8	74LS08	14	7
IC9	74LS02	14	7
IC10	74LS08	14	7
IC11	74LS32	14	7
IC12	74LS04	14	7
IC13	74LS02	14	7
IC14	74LS138	16	8
IC15	74LS138	16	8
IC16	74LS138	16	8
IC17	74157	16	8
IC18	74157	16	8
IC19	74S74	14	7
IC20	74LS08	14	7
IC21	2716	24	12
IC22	2716	24	12
IC23	74LS245	20	10
IC24	6116-3	24	12
IC25	74LS393	14	7
IC26	74LS245	20	10
IC27	74LS74	14	7
IC28	74LS133	16	8
IC29	74LS04	14	7
IC30	74LS32	14	7
IC31	74LS51	14	7
IC32	74LS51	14	7
IC33	74LS00	14	7
IC34	74LS74	14	7
IC35	74LS74	14	7
IC36	4164-15 (150 ns)	16	8
.			
.			
.			
IC99	4164-15	16	8

Power wiring table for figure 3.

PC Address	Trump Card Port	Function
03EE	3	A "write" to this port by the processor that has current use of the bucket will cause the 8-bit address counter, IC25, to be reset to 0. It will also release the bucket for use by the other processor. If bit 0 of the data bus is set to a 0 when this write is performed by the 8088 processor, a nonmaskable interrupt (NMI) is also issued to the Z8001. Reading this port allows either processor to see data at the current address of the counter without incrementing the counter. If the bucket is not available, a read operation to this port will return an FF.
03EC	1	A read operation to this port by the processor that has the bucket reserved will return the data at the current address of the counter and increment the counter at the end of the read operation. A read by the processor that does not have the bucket will return a value of hexadecimal FF and will not increment the counter. A write to this port by the processor that has reserved the bucket will enter data at the current address of the counter and increment the counter at the end of the operation. A write by the processor that does not have the bucket will not enter data and the counter will not be incremented.
03E8	x	This port is not used for data transfer by the Z8001. A write to this port by the 8088 will issue a reset to the Trump Card.

Table 2: *Communication between the Z8001 and the 8088 is through the "bucket," a FIFO buffer made from a type-6116 static-memory chip and support components. Shown here are the three basic bucket functions and the addresses and codes for each.*

Text continued from page 44:
which contain a bootstrap loader for cold-start-up and system-diagnostic routines. Address lines LA1 through LA11 are connected to the EPROMs, IC22 (even byte) and IC23 (odd byte). There is no need to use the $\overline{ODD}$ or $\overline{EVEN}$ bank-select lines since no data is ever written into the EPROMs. The signal RAM/$\overline{PROM}$ is connected to the $\overline{CS}$ pin on the 2716s. The $\overline{MREQ}$ (memory request) signal from the Z8001 is also connected to pin 18 ($\overline{OE}$ or output enable) of the 2716s, to inhibit the possibility of bus contention during I/O cycles.

Status Signals

Various status signals tell the rest of the system about the processor's condition and the type of information that is appearing on the address/data bus. The status signals are as follows:

Read/$\overline{Write}$: The R/$\overline{W}$signal is used to indicate the direction of the current bus transaction. When high, the direction of data is toward the Z8001. Data is clocked into the processor at the occurrence of a positive-going pulse on $\overline{DS}$ (data strobe). When $\overline{DS}$ is low, data flows from the processor outward.

Normal/$\overline{System}$: The N/$\overline{S}$ signal indicates whether the processor is operating in the system (supervisory) mode or normal (user) mode of operation. This control line is used when there is a multitasking and/or multiuser type of environment to segregate system functions and memory. The line is unused in the Trump Card.

Byte/$\overline{Word}$: The B/$\overline{W}$ line is provided to enable the Z8001 to perform byte operations on memory. When high, it indicates that a byte operation is to take place; a low state indicates word operations. This signal is also used in the $\overline{ODD}$ or $\overline{EVEN}$ memory-select logic.

Status Lines: Lines ST0 through ST3 are utilized to define the exact type of transaction occurring on the bus. Only 4 of the 16 possible codes are required for operation of the Trump Card. The first status code, 0000 (Internal Operation), is decoded but unused. The second operation code, 0001 (Memory Refresh), is output by the internal Z8001 memory-refresh timer and is used in refreshing the on-board dynamic RAM. (This signal is ANDed with $\overline{MREQ}$ and is used as one of the two select signals in the row-address-strobe generation logic.) The third operation, 0010 (Standard I/O Reference), is used in the process of communicating with the host 8088 processor. The fourth operation code, 0011 (Special I/O), denotes I/O associated with the signal $\overline{SPIO}$ and is reserved for future expansion.

Clock Generation

The basic clock rate for the Z8001 on the Trump Card is provided by IC2, a Zilog Z8581 clock generator and controller (CGC). The Z8001's clock-input maximum voltage must come within a certain range of the power-supply potential (precisely V_{cc} − 0.4 V) and have a maximum rise and fall time of 10 nanoseconds (ns). Such requirements are difficult to meet with standard oscillators and

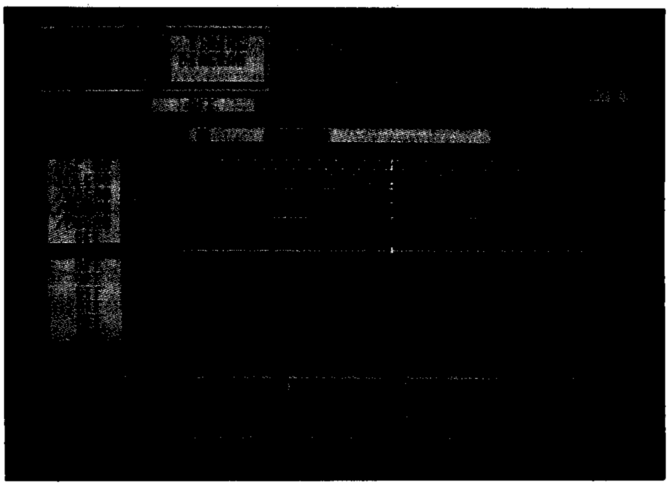


Photo 5: *A typical display on the Tektronix 1240 logic analyzer: the column-address-strobe/row-address-strobe timing of the Trump Card.*

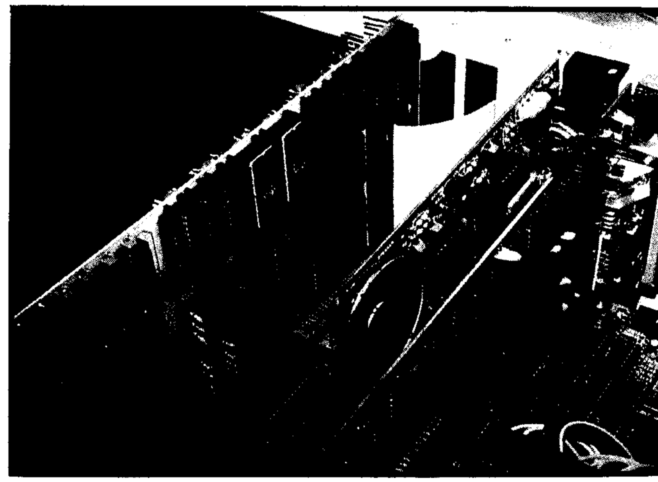


Photo 6: *To aid in my initial development, a Zscan-8000 emulator is plugged by a ribbon cable into the Z8001 socket on the Trump Card. In emulation mode, the Zscan-8000 can run diagnostic programs and exercise all functions of the Trump Card at 4 MHz. Hardware debugging is greatly simplified because all sections of the hardware need not be working to use the emulator.*

TTL (transistor-transistor logic), but they are easily met by the CGC. The Z8581 also provides an easy and effective means of adjusting the processor's bus cycles to the speed of available memory devices.

The CGC is used on the Trump Card to stretch specific bus cycles. As used on the Trump Card, the Z8001 does three different basic categories of operations: internal operations, memory access, and input/output operations. The timing of the ZCLK signal emitted by the CGC depends on which of these bus activities is taking place. The Z8581 can be configured to add wait states that enable the use of 150- and 200-ns RAM chips.

Trump Card/Host Communication

The "bucket" is the communications interface between the PC and Trump Card. This FIFO (first-in/first-out)-type dual-port memory configuration consists of a 6116 static memory (IC24), an 8-bit address counter (IC25), two data-bus buffers (ICs 23 and 26), and the necessary control logic to arbitrate access. Programs and instructions are passed between the two computers via this FIFO circuitry. As far as the PC is concerned, the bucket appears as two I/O port addresses. A system of software handshaking between the computers determines which has reserved and is using the bucket. Table 2 shows the port addresses and their functions.

It is not possible for both processors to have use of the bucket at the same time. With the processors running asynchronously, arbitration is necessary. It is provided by four D-type flip-flops: two for access requests and two for access reservations. The two access-request flip-flops are clocked by the transition of an access-request signal from either processor ($\overline{IORQ}$ for the Z8001 and $\overline{PCSEL}$ for the 8088). The preset inputs of these flip-flops are connected to the $\overline{HOLD}$ signal, which is active whenever one of the processors has succeeded in reserving the use of the bucket. When $\overline{HOLD}$ is active, it prevents the other processor from gaining access.

The Z8001 communicates through the bucket for all its normal I/O by activating the $\overline{IORQ}$ line. The 8088 selects the bucket when it performs either an IOW (I/O write) or IOR (I/O read) in the range of the IBM's regular memory-address space from hexadecimal 03E8 to 03EE. Accesses to these addresses are decoded by IC28 to generate the Trump Card's $\overline{PCSEL}$ signal.

The two access-reserve flip-flops sample the output of the request flip-flops 180 degrees out of phase with each other. This is done to prohibit simultaneous requests from being honored. These flip-flops are cleared by a reset command issued from the reserving processor.

The $\overline{Q}$ outputs from these flip-flops are combined by a logical AND function with the processor request to form the active select states used by the bucket: $\overline{Z}$ ANDed with $\overline{DS}$ for the Z8001 and $\overline{PC}$ for the 8088. Whenever either request flip-flop is active, the $\overline{HOLD}$ signal is active and is used as the chip-select input on the 6116 memory. The FIFO memory, however, is written to by the Z8001 only when a "write bucket with increment" command is used.

The $\overline{WE}$ signal, connected to the write-enable input of the 6116 memory, is active during either a Z8001 I/O request (with R/$\overline{W}$ low and $\overline{DS}$ active) or an 8088-generated write to the bucket (with $\overline{PC}$ and INC active and $\overline{BIOR}$ inactive). The INC signal is active whenever the processor that has control of the bucket sets bit 1 of the address low. The CLEAR signal is active when bit 1 of the address generated by the selected processor is high and a write operation is occurring.

A nonmaskable interrupt to the Z8001 is generated when the 8088

Listing 2: *Bootstrap initialization program for the Trump Card written in Z8000 assembly language.*

```
0000             00  DW                      - Reserved control word
D000             02  DW                      - Flag and control word
0000             04  DW                      - Segment Register
0008             06  DW                      - Segment Offset
2100 9E01        08  LD R0,#%9E01            - Set refresh freq and enable
1404 0003 0001   0C  LDL RR4,#%0003 0001     - Set port addresses
7D0B             12  LDCTL REFRESH,R0        - Load refresh value
3E40             14  OUTB @R4,RH0            - Set R4 as reset-bucket port
3C40             16  INB RH0,@R4             - Read bucket without increment
A800             18  INCB RH0,#1             - Increment input value
E6FC             1A  JR 0 EQ,%0014           - Repeat if equal to 0
3C40             1C  INB RH0,@R4             - Read bucket
8A80             1E  CPB RH0,RL0             - Compare bucket value to 01
EFF9             20  JR NC UGE,%0014         - Do again if not > 01
C803             22  LDB RL0,#%03            - Load R0 with bucket available #
3E58             24  OUTB @R5,RL0            - Load bucket with R0
8400             26  ORB RH0,RH0             - Set zero flag if RH0 is 0
E6F5             28  JR Z EQ,%0014           - Restart boot, else continue
3C52             2A  INB RH2,@R5             - Read bucket and save in register
3C53             2C  INB RH3,@R5             - Read bucket and save in register
3C5B             2E  INB RL3,@R5             - Read bucket and save in register
3C51             30  INB RH1,@R5             - Read bucket and save in register
3C59             32  INB RL1,@R5             - Read bucket and save in register
3A50 0120        34  INIRB @RR2,@R5 R1       - Read bucket into memory
F013             38  DBZNZ RH0               - Decr RH0 and at 0 goto 0014
3E40             3A  OUTB @R4,RH0            - Reset bucket
AB35             3C  DEC R3,#6               - Decrement value in R3 six times
                                               to set up first addr of code
1E28             3E  JP @RR2                 - Jump to loc defined in RR2
```

performs a write operation to the bucket with address bit 1 and data bit 0 both low. This interrupt is latched in a D-type flip-flop and is not cleared until the Z8001 issues a Nonmaskable-Interrupt Acknowledge (status decode 5) or until the host computer resets the Trump Card. (See table 2 for more detail.)

Booting Trump Card

When you plug the Trump Card into a slot in the IBM PC and turn on the computer, the Trump Card automatically executes the bootstrap-loader routine contained on board in EPROM. The loader routine is only 31 words (62 bytes) long; its assembly code is shown in listing 2.

I used two 2716 EPROMs instead of bipolar PROMs to store the bootstrap loader because they are both cost-effective and easier to program than bipolar PROMs. Two byte-wide memory devices are required because the Z8001 is a processor with a 16-bit word length. Each machine-language instruction (expressed as four hexadecimal digits) is separated into high- and low-order bytes (or "even" and "odd," if you prefer); the high and low bytes are stored in separate EPROMs. When you examine a particular 16-bit memory location, you are actually viewing the information provided from two 8-bit sources.

Using Trump Card

Trump Card is transparent to normal PC operation. To start Trump Card, you run a program stored under PC-DOS called LDZSYS. This is the Trump Card communications software that runs on the 8088. If you always want Trump Card features available, you can add this program to your regular AUTOEXEC batch file. When LDZSYS has completed initialization, it returns to the PC-DOS A> prompt to wait further instructions.

At this point, I generally configure part of Trump Card's memory as a RAM disk, using a program called SETRMDSK. This is done as follows:

```
A> SETRMDSK 4
A>
```

SETRMDSK configures the additional C drive to your existing system under DOS 2.0. The number following SETRMDSK determines how many 64K-byte blocks you wish to reserve as a RAM disk. In this case, I set up a 256K-byte drive. The RAM-disk size can be 128K to 387K bytes, depending upon the amount of memory on the Trump Card board. (While you can set a 128K-byte RAM disk in a 256K-byte board, you might have problems running large BASIC or C programs concurrently.)

The memory that I've set as a RAM disk is completely separate and in addition to the regular IBM PC memory. Even if you have a 640K-byte PC, up to 387K bytes of Trump Card's memory would be available as additional RAM-disk configured storage space.

I used the RAM disk to speed up the process of writing these articles. Many word processors, like the Volkswriter I use, make extensive calls to the disk for help files and command-execution files. After a while, the noise and delay get aggravating. To remedy this situation, I run the SETRMDSK 4 sequence just described to create the 256K-byte RAM drive C and then add the following:

```
A> COPY *.* C:
A> C:
C>VW
```

This copies the entire contents of the Volkswriter distribution disk to the RAM disk, sets it as the default drive (C), and starts the word processor. When I now press a function key the action is instant and silent. To guard against power interruptions, drive B is designated as the hard-storage location and periodically I store the article file to it.

Trump Card's other features are equally simple to use. BASIC, C, Z80, and editing files can be stored on the same disk and executed with similar ease. While I'll explain it in greater detail next month, a possible sequence of Trump Card operations is shown in table 3.

Rewarding Diligence

I've been having a lot of fun with Trump Card. I haven't done much assembly-language or C programming yet, but it has renewed my faith in BASIC.

Trump Card is not an easy project to build. Compared to other Circuit Cellar projects, however, it's manageable. I was surprised at the number of readers who hand-wired the 121-chip MPX-16 PC-compatible computer that I presented last year. Their letters suggested that the motive was neither money nor masochism. In-

Circle 376 on inquiry card. →

WAREHOUSE SOFTWARE

TECHNICAL INFORMATION (602) 842-1133

Call for programs not listed. We will try to meet or beat any legitimate price for CP/M or IBM PC Software. Most disk formats available.

DATA BASE MANAGEMENT SYSTEMS

UNBEATABLE PACKAGE PRICE!! DBASEII + Instruction Book "Using DBASEII", Extra Diskette with Dbase Accounting, Mail list and Inventory Programs. For IBM PC and CP/M. Call for our special price.

Fox and Geller Quickcode $175
DB+SORT $89
Condor III $340
NWA Statpack $350
TIM IV $269
Infostar $259
PFS File $95
RBASE 4000 .... $299
Personal Pearl $145
Fast Facts for IBM PC $135

WORD-PROCESSING

Wordstar, Mail Merge, Spellstar, Index . $359
Wordstar $245
Mail Merge or Spell Star $135
Microsoft Word W/Mouse $295
Word Perfect $295
Volkswriter for IBM PC $115
Wang Spellchecker $36
Metasoft Benchmark $265
Multimate $295
Peachtext 5000 .... $219

SPREADSHEETS

Calcstar - IBM PC Spec. $65 .. Others $95
Supercalc II $159
Supercalc III $199
Microsoft Multiplan $159

ACCOUNTING

TCS. Equivalent of Peachtree - Specially Augmented By Warehouse Software Customized For Your IBM PC Terminal and Printer - GL, AR, PA, AP, CP/M, for PC XT, DOS 1.1, 2.0
Each Module $75 For All Four $275

CYMA .... Call
Peachtree GL, AR, AP $245
Home Accountant Continental $95
Real World, GL, AR, AP, PA . each $350

TRANSFER PROGRAMS

Move-It $85
Microstuff Crosstalk $105

Best Price in U.S. for IBM PC or Clones Multifunction Board—Includes Async Adapter, Parallel Adapter, Clock with battery back-up and Software, 64K Memory Expandable to 384K.
1 year warranty $265

LANGUAGES

Lifeboat C Compiler $295
Microsoft C Compiler $335
Microsoft Pascal Compiler $245
Microsoft Basic Compiler $285
Microsoft Basic Language $250
CBASIC 86 for IBM PC $135
CBASIC CP/M-80 .... $95

16 Pounds of IBM PC DOS Compatible Portable Computer-Hyperion-2 Disc Drives - Software.
List 3690 .... Sale $2665

FOR PC DOS

Norton Utilities $55
Copy II PC $34
Prokey V3.0 .... $86
Howardsoft Tax Preparer 84 .... $215
Microsoft Flight Simulator $38

HARDWARE

Hayes 1200 Modem $495
Hayes 1200B Modem $435
Anchor Signalman 1200baud Modem . $285
Plantronics Color + Board $365
Koala pad for IBM PC $88
Quadram Color I Board $199
256 K Ram Board $299
Princeton RGB Hi-Res Monitor $495
Gemini 15X, 10x Printers .... Call
Corona Computer - Port. or Desk Top Call

TERMS: Prices include 3% cash discount. Add 3% for charge orders. Shipping on most items $5.00. AZ orders +6% sales tax. Prices subject to change.

TOLL FREE ORDER LINE 1-800-421-3135
WAREHOUSE SOFTWARE
4935 West Glendale Ave., Suite 12
Glendale, AZ 85301

A> LDZSYS	(initialize Trump Card)
A>	(return to PC-DOS or use Trump Card)
A>G	(turn over PC operation to Trump Card)
:	(Trump Card prompt)
: EE *filename*	(edit a file)
or	
: Z80EM	
filename	(emulate CP/M-80 and run Z80 programs)
or	
: C *filename*	(compile and run a C program)
or	
: Y *filename*	(compile and run Z8000 assembly language)
or	
: BASIC *filename*	(compile and run BASICA programs)
://	(return to PC-DOS)
A>	

Table 3: *A Trump Card operating sequence.*

stead, building these projects enabled them to experiment with digital circuitry yet be secure in the knowledge that their project would work. I hope this project elicits a similar response, and I'd like to reward such enthusiasm in advance.

Esoteric peripherals such as Trump Card depend a great deal on sophisticated software to fully exercise their capabilities. Unfortunately, when experimenters build rather than purchase boards, they often have to use great ingenuity to obtain software.

More than five man-years of development effort went into the present support packages for Trump Card. Some, like TBASIC and the RAM disk, were contracted by me, while others, like the C compiler and Y (a Z8000 assembler), were written by Zilog. Combined with the CP/M-80 emulator, Z8000 operating system, and telephone-book-size documentation, it is a formidable package that is difficult to independently price.

I want to encourage you to build your own Trump Card if that is your choice. If you send me a picture of the completed unit, I will send you a copy of the complete software and the documentation (provided it is for personal, noncommercial use) for the cost of duplication and shipping

The following items are available from

Sweet Micro Systems Inc.
50 Freeway Dr.
Cranston, RI 02910
(800) 341-8001 for orders
(401) 461-0530 for information

1. *Trump Card, including IC sockets, assembled and tested with 256K bytes of the 512K-byte RAM space populated. Includes TBASIC compiler, C compiler, Z8000 Y assembler, CP/M-80 emulator, RAM-disk driver, and documentation. Software supplied on a PC-DOS 2.0 disk unless otherwise specified.*
256PCB $995
2. *Trump Card, assembled and tested with 512K bytes of dynamic RAM installed. Includes support software described above and documentation. Software supplied on a PC-DOS 2.0 disk unless otherwise specified.*
512PCC $1325
3. *Partial kit for Trump Card. Includes fully socketed wave-soldered printed-circuit board, bootstrap EPROMs, 10-MHz Z8001, and Z8581. Includes software and documentation described above. Other integrated circuits not included. Software supplied on a PC-DOS 2.0 disk unless otherwise specified.*
0KPCA $525

Please add $10 for shipping and handling in the continental United States. Please add $20 for shipping to other locations. Rhode Island residents please add 6 percent sales tax.

($30). The software houses and other parties in this project have waived all royalties as a gesture of support for the Circuit Cellar.

Next Month:

In June's article, I'll describe the software in detail and do a little benchmarking. ■

Diagrams pertaining to the Z8000 are reprinted by permission of Zilog Corporation. Z8000 and Z80 are trademarks of Zilog.

Steve Ciarcia (pronounced "see-ARE-see-ah") is an electronics engineer and computer consultant with experience in process control, digital design, nuclear instrumentation, and product development. In addition to writing for BYTE, he has published several books. He can be contacted at POB 582, Glastonbury, CT 06033.

Editor's Note: *Steve often refers to previous Circuit Cellar articles. Most of these past articles are available in reprint books from BYTE Books, McGraw-Hill Book Company, POB 400, Hightstown, NJ 08250.*

Ciarcia's Circuit Cellar, Volume I *covers articles that appeared in BYTE from September 1977 through November 1978.* Ciarcia's Circuit Cellar, Volume II *contains articles from December 1978 through June 1980.* Ciarcia's Circuit Cellar, Volume III *contains articles from July 1980 through December 1981.* Ciarcia's Circuit Cellar, Volume IV, *soon to appear, will contain articles from January 1982 through June 1983.*

References

1. Ciarcia, Steve. "Build the Circuit Cellar MPX-16 Computer System." Part 1, BYTE, November 1982, page 78. Part 2, BYTE, December 1982, page 42. Part 3, BYTE, January 1983, page 54.
2. Majundar, S., K. Kumar, and K. S. Raghunathan. "Interface Unites Z8000 with Other Families of Peripheral Devices." *Electronics*, July 28, 1981, page 156.
3. Rampil, Ira. "Preview of the Z8000." BYTE, March 1979, page 80.
4. Simington, R. B. "The Intel 8087 Numerics Processor Extension." BYTE, April 1983, page 154.
5. Shima, Masatoshi. "Genealogy of the Z8000." *Electronics*, December 21, 1978, page 83.
6. Williams, Gregg. "Benchmarking the Intel 8086 and 8088." BYTE, July 1983, page 147.
7. Zingale, Tony. "Intel's 80186: A 16-Bit Computer on a Chip." BYTE, April 1983, page 132.

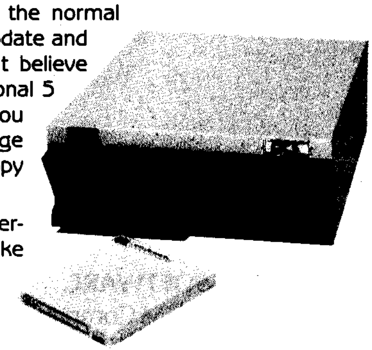
To receive a complete list of Ciarcia's Circuit Cellar project kits, circle 100 on the reader service inquiry card at the back of the magazine.

There's a better way to pack more muscle in your micro.

The HarDrive™ by QuCeS. More bytes for your buck. In a flash.

If you're looking for a way to get more performance out of your microcomputer, look no further than QuCeS.

With a QuCeS HarDrive subsystem, you can make your micro behave almost like a mainframe. 10 to 114 megabytes of mass storage lets you handle data bases that would make the normal micro blow a fuse. And you can access, update and process data so incredibly fast, you won't believe your eyes. A QuCeS HarDrive with an optional 5 megabyte backup cartridge, also means you won't have to rely on a very unreliable storage medium for your crucial data—namely floppy disks—ever again.



Another QuCeS plus is compatibility. It interfaces with most popular microcomputers like IBM, Radio Shack, Apple, DEC, Epson—you name it. Installation couldn't be easier, our software is easy to use, and each HarDrive is backed by a 1-year warranty.

The QuCeS HarDrive. It will make your micro mightier and faster than ever before.

For complete details, contact

QCS inc

Quality Computer Services

3 Quces Drive, Metuchen, N.J. 08840 (201) 548-2135
TELEX 299410 QCS

Circle 328 on Inquiry card.

C·I·A·R·C·I·A'S C·I·R·C·U·I·T C·E·L·L·A·R

Trump Card Part 2: Software

TBASIC and C compilers and an assembler

BY STEVE CIARCIA

Last month, we looked at the hardware of the Trump Card, a coprocessor board for use with the IBM Personal Computer (PC) or compatible computers. The presentation centered mainly on the Zilog Z8000's processor architecture, the support circuitry, and the interface between the Z8000 and the Intel 8088. But the power of the Trump Card can be unleashed only by the right software. This month, I'll describe the collection of software I've assembled for the Trump Card from several sources—most of it designed to support further program development. Let's first quickly review the features of the Trump Card.

WHAT IS THE TRUMP CARD?

The Trump Card (see photo 1) is a printed-circuit board that plugs into any I/O (input/output) expansion slot of an IBM PC, an IBM PC XT, or any computer compatible with them. It contains a Zilog Z8001 16-/16-bit microprocessor (the memory-segmented version of the Z8000) running at 10 MHz and up to 512K bytes of RAM (random-access read/write memory). The Trump Card communicates with the PC's built-in 8088 processor through a 256-byte FIFO (first-in/first-out) buffer.

A variety of software is available for the Trump Card. The most important, from my point of view, is the language system for its special version of BASIC. As you would expect, the Trump Card's TBASIC compiler excels at making user programs run fast, but it's also so easy to use that it makes some interpreted versions of BASIC look clumsy. The source language accepted by the TBASIC compiler is nearly identical with that of the IBM PC's Advanced BASIC interpreter (BASICA) and includes a few enhancements, such as compilation of programs larger than 64K bytes.

Other software included with the Trump Card follows:

- CP/M-80 emulator. The Trump Card can run programs designed to run under Digital Research's CP/M-80 DOS (disk operating system) by emulating the 8-bit Z80 instruction set and DOS calls. No special file headers or instruction-translation programs are required.
- C compiler. The source language accepted by this compiler follows that of Kernighan and Ritchie with a few minor differences (see reference 6).
- Screen editor. Incorporating many of the features normally found only in word-processing packages, the screen editor, called EE, enables you to write or examine ASCII (American National Standard Code for Information Interchange) text files for use either with the Trump Card or in the normal IBM PC environment.
- Y multilevel-language compiler. The unusual Y language system is essentially a structured assembler that enables Pascal-like control constructs and data types, arithmetic expressions with automatic or specified allocations of registers, and procedure calls with parameter passing.
- Debugger. With the debugger, you can examine and replace the contents of memory and registers, set breakpoints, or single-step through programs. Intended to aid in program development, the debugger is an integral part of Y.
- Semiconductor disk emulator. Under versions of PC-DOS equal to or higher than 2.0, Trump Card can allocate 128K to 387K bytes of its on-board RAM to function as a RAM disk or disk emulator. This memory is separate from the memory already existing on the PC's motherboard or other expansion boards and resides in the Z8000's separate address space. The Trump Card can run another function concurrently with the disk emulator.

(text continued on page 116)

Copyright (c) 1984 Steven A. Ciarcia. All rights reserved.

Steve Ciarcia (pronounced "see-ARE-see-ah") is an electronics engineer and computer consultant with experience in process control, digital design, nuclear instrumentation, and product development. In addition to writing for BYTE, *he has published several books. He can be contacted at* POB 582, *Glastonbury,* CT 06033.

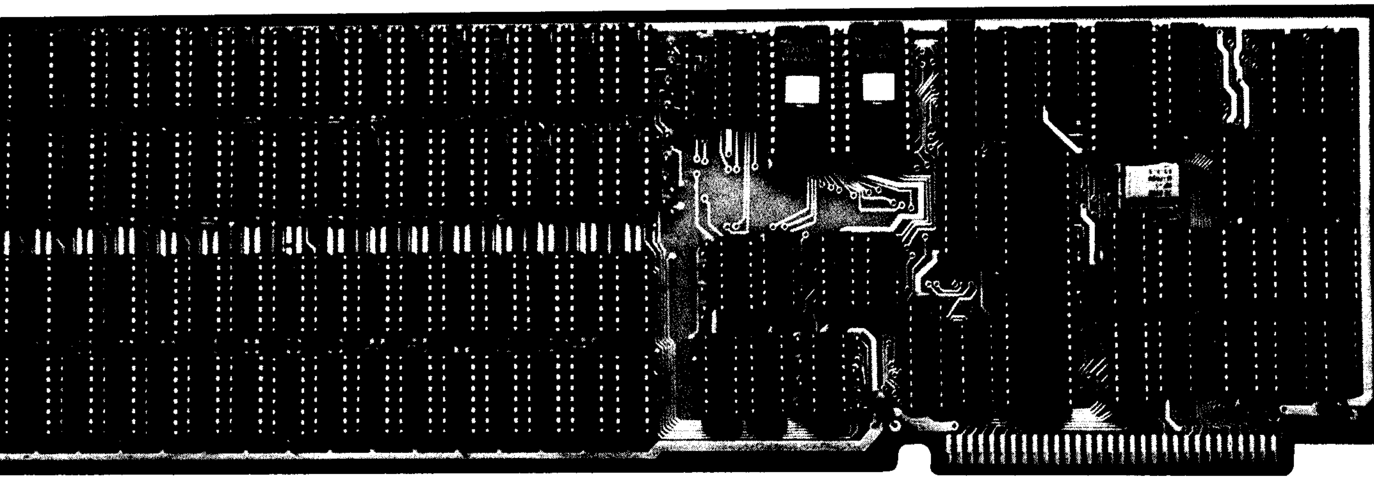


Photo 1: *The soldered prototype printed-circuit version of Trump Card.* RAM *sockets are at left.* EPROMs *are top center, and the* Z8001 *and support chips fill the remainder of the board.*

TBASIC *is a new version of the* BASIC *language that looks like an interpreter and executes like a compiler.*

(text continued from page 115)

Bringing the Trump Card Up

To initialize the Trump Card, run a program called LDZSYS.COM from PC-DOS. When it has completed setting up the Trump Card and installing the device driver needed by PC-DOS to communicate with it, LDZSYS returns control to PC-DOS and the host 8088 processor, with the Z8000 awaiting further instructions. Example 1 in the text box on page 118 contains examples of this and other typical user commands (in italics) and the system's response (in roman type). The operation of the Trump Card is transparent to programs running on the host 8088. (If you think that you will always want the Trump Card's capabilities available, you can add a line containing LDZSYS to your PC-DOS AUTOEXEC.BAT file.)

To begin using the Trump Card, execute the "go" program, G.COM (G). When the Z8000 has control of the system, it returns with a colon prompt, as the fourth line of example 1 shows, indicating that the Z8000 is ready to accept commands. The text box also shows the command format for editing and compiling files and programs, which may be stored on the same disk used to boot PC-DOS.

Interpreters versus Compilers

As I said last month, a chief cause for my building the Trump Card was a feeling of frustration with the slowness of BASIC interpreters. I had, of course, considered using an off-the-shelf BASIC compiler to speed up my programs, but I did not relish all the overhead operations required by the compilers I had seen, such as Microsoft's BASIC compiler.

The typical compiler requires three separate operations to run a BASIC program. First, the program source code must be written using an editor program. Next, the ASCII program text from the editor is compiled into object code and stored in a disk file, which often takes several minutes. Finally, the special BASIC run-time processor is loaded from the disk to supervise execution of the object program. At last, the program does its thing.

Interpreters, for all their inefficiency of execution, do have one important benefit: you quickly can add a line to your program and type RUN to see its effect. But if you want to change a line in a compiled program, it's back to the editor and all the way through the process again. So when you finally have your debugged, compiled program, it may indeed execute 100 times faster than under an interpreted one, but it may have taken you 10 times as long to get it running right. I think this is one reason BASIC compilers are not in wider use.

To counter this criticism, compiler manufacturers suggest developing code on an interpreted BASIC first and then compiling it. Such a suggestion, while valid, ignores the reason for a compiler in the first place. If a hundredfold increase in speed is necessary to achieve a program's objective, it hardly makes sense that to write and test the original program you must wait 100 times longer each time you must run it.

The answer seemed relatively trivial to me—simply write a version of BASIC that looks like an interpreter and executes like a compiler. The result is TBASIC.

The Trump Card's TBASIC language system is a BASIC compiler that offers significantly faster execution of BASIC programs than does a BASIC interpreter, while furnishing an operating environment much like that of an interpreter. TBASIC bridges the gap between traditional BASIC interpreters, which have built-in editors and are known for ease of use, and typical BASIC compilers, which produce rather efficient object code but can be difficult to work with. TBASIC's extremely fast compilation times and its capability for immediate-mode execution make working with it as easy as working with a friendly but slow interpreted BASIC, but the resulting programs run with the speed of a compiler. Unlike other compilers, the object code is not written into a disk file before execution (unless you request it). Therefore, no long delays are needed. When you load the file into the Trump Card, TBASIC compiles the program in a few tenths of a second.

Most programs that will run under the IBM PC's BASICA interpreter can be fed into TBASIC for compilation. You can use either the Trump Card's EE screen editor or the BASICA editor to write the programs. But if you then run the same program under both BASICA and TBASIC, depending upon the instructions you use, you will notice an increase in program performance by a factor of anywhere from 7 to 100. A listing of TBASIC's keywords is shown in table 1. TBASIC also supports most of BASICA's color and graphics commands (see photo 2).

Line numbers aren't required in the source code of programs written for TBASIC except where a line is to be referenced elsewhere in the program; for example, the destination of a GOTO or GOSUB statement would need a line number. Although not requiring them, TBASIC certainly allows line numbers on every line, so existing BASICA source code will run under TBASIC, to the extent that the program is compatible with TBASIC's syntax. Such programs can immediately benefit from the increase in performance provided by TBASIC.

The development of a program using a BASIC interpreter occurs in two modes: editing the program and running it. Developing a program with TBASIC involves three modes: editing, compiling, and running. Obviously, the only difference is compilation, which is invoked on the Trump Card by the DO command; once the program has been compiled, the familiar RUN command executes it.

Example 2 on page 118 shows some examples of the kind of interaction that occurs when you use TBASIC: how to enter a program using the EE editor, compile it, and run the compiled program. In the text box, input by the user is shown in italic type while the system's prompts and output are shown in roman characters.

During compilation of a program, error messages are issued each time an error is encountered. The line of the source file in which the error was detected is displayed; in some cases, an error message is also displayed. After an error is found and displayed, compilation continues and any other errors found also will be displayed. When the compilation has been completed, a list of any undefined symbols also may be output, in which case the program should not be run.

TBASIC Programs

Three methods can be used for entering program statements into the system for compilation under TBASIC. The first is to use the Trump Card's built-in EE screen editor, as mentioned previously (see photo 3). A second method is to enter the statements using TBASIC's direct-entry mode. The third choice is to enter and test the program using the computer's regular BASICA interpreter and then run it for effect using TBASIC. The three methods may be used interchangeably.

Example 3 shows an example of these functions with a minimally modified version of the Sieve of Eratosthenes program often used as a system benchmark (see references 4 and 5). A program called SIEVE.S was previously written in BASICA and stored as an ASCII file on the disk in drive B.

Suppose you want to run the program under both BASICA and TBASIC while recording how long it takes to be executed. You could use a stopwatch, but it's easier to add a few more program lines that record the starting and ending times automatically by calling the TIME$ function. It's possible to invoke the editor directly from TBASIC, as shown in example 3, to add two lines. And you can see that TBASIC took about 2 seconds to run the modified program as measured by the internal clock.

The program changes quickly were added and executed, and, when you left the editor with a QU command, the file SIEVE.S on drive B was updated to contain the TIME$-function statements. After running the slightly revised program under BASICA, you see that it takes 202 seconds, around 100 times as

(text continued on page 118)

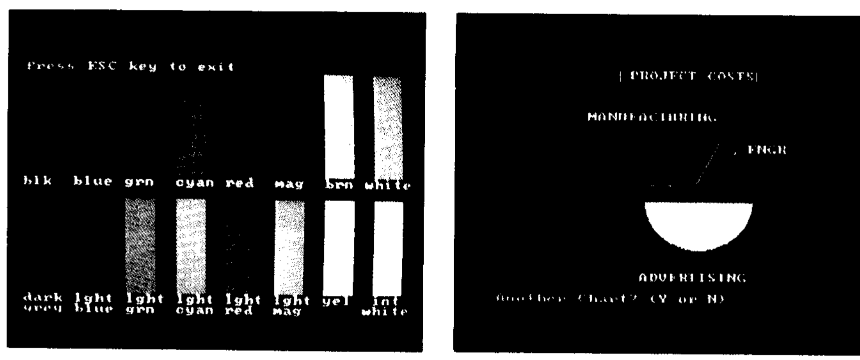


Photo 2: *Color (2a) and graphics (2b) tests demonstrate* TBASIC's *support of color/graphics commands normally associated with* BASICA.

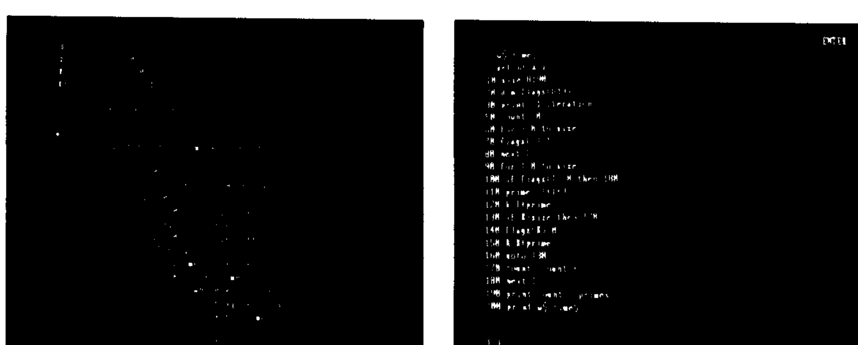


Photo 3: *Programs in* BASICA *(3a) and in* C *(3b) can be written for Trump Card or the PC by using Trump Card's built-in* EE *editor.*

(text continued from page 117)

long. Now consider the aggravation of making changes in programs that take this long to run and waiting for the results each time. Perhaps you now understand why I built the Trump Card. If you're interested in how fast some other computers and BASIC systems executed essentially the same program, see table 2. Another program that demonstrates how TBASIC speeds things up is the simple looping benchmark shown in listing 1. The results are shown in table 3.

> TBASIC *speeds up development and debugging as well as execution.*

Not all programs run a hundred times faster in TBASIC. The Sieve program purposely uses integer arithmetic and avoids difficult floating-point calculations. But we can get an idea of floating-point performance from the simple benchmark routine of listing 2. In this program, TBASIC takes 3.2 seconds while BASICA takes 24.2. This benchmark shows the wide variation in performance you can expect from a different mix of statements.

Of course, most other BASIC compilers for the IBM PC also can demonstrate dramatic speed increases over interpretive BASICA. But I believe that TBASIC is different because it speeds up development and debugging as well as execution.

(You might be wondering if the installation of an Intel 8087 Numeric Processor Extension in the IBM PC would help speed up execution of BASIC programs. Under BASICA, it would have no effect whatsoever because BASICA is not written to use it. I did a quick informal test using Morgan Professional BASIC, which uses the 8087. Morgan BASIC took 12.8 seconds to execute listing 2.)

TBASIC's Ease of Use

TBASIC has many of the same convenience features for running programs that an interpreter has. You can use the commands RUN, RUN<*line number*>, GOTO<*line number*>, and GOSUB<*line number*> just as in BASICA. To stop a program from the console, you just hit

EXAMPLE 1

Computer Interaction	Comments
A>*LDZSYS*	Initialize Trump Card from PC-DOS.
A>	Control is returned to PC-DOS.
A>*G*	Turn control over to Trump Card.
:	(Trump Card's command prompt.)
:	
: *EE<filename>*	Edit a file.
:	
: *Z80EM<filename>*	Emulate Z80 and run CP/M-80 programs.
:	
: *C<filename>*	Compile and run a C program.
:	
: *Y<filename>*	Compile and run Z8000 structured assembly language.
:	
: *BASIC<filename>*	Compile and run TBASIC programs.
:	
: *//*	Exit from Z8000 command interpreter.
A>	Control returns to PC-DOS.

EXAMPLE 2

Computer Interaction	Comments
A>*B: (Return)*	Set the PC-DOS default drive to B. TBASIC will also use this drive as its default drive.
B>*G (Return)*	Type G to "go to" the Z8000.
:	The colon (:) is the Z8000 system command prompt, equivalent to the A> or B> prompt of PC-DOS.
:BASIC (Return)	Invoke TBASIC.
-	The hyphen (-) is the command prompt used by TBASIC; you may now invoke any TBASIC command.
-EDIT TESTFILE (Return)	Edit a new file using the EE editor.
T	You are now in the EE editor
EOF	in command mode.
E	Type "E" to enter text.
FOR I=1 TO 5 (Return)	Type in your BASIC program.
PRINT "Demo program" (Return)	
NEXT I (Return)	
(Escape)	Hit the Escape key to leave the Enter mode.
QU (Return)	Quit and save program on default disk B.
-	The "-" prompt shows that you are now back in TBASIC.
-DO (Return)	Compile the program by using the DO command (takes about 0.1 second).
-	Your program is now compiled.
-RUN (Return)	Type RUN to execute the compiled program.
Demo program	Compiled program output.
Demo program	
Demo program	
Demo program	
Demo program	
-// (Return)	The // command exits TBASIC. (The SYSTEM command could be used instead.)
:DIR (Return)	Call for a disk directory from the command interpreter.
DIRECTORY OF DRIVE B:	
TESTFILE	There's the source file you created with the EE editor.
:// (Return)	The // command exits the Z8000's B> command mode and returns control to PC-DOS.

EXAMPLE 3

Computer Interaction	Comments
B> *G (Return)*	Go to the Z8000 operating system.
: *BASIC SIEVE.S (Return)*	Get SIEVE.S from disk and compile it in about 0.2 second.
- *RUN (Return)*	Execute program in TBASIC.
1 ITERATION	
1899 PRIMES	The program produces output and ends.
-	Awaiting next command.
-EDIT (Return)	Call the editor from TBASIC prompt.
T	T indicates display from top of file; the complete Sieve file is displayed, ready to edit.

```
5 DEFINT A<Z
10 SIZE = 8190
20 DIM FLAGS(8191)
30 PRINT "Only 1 iteration"
50 COUNT = 0
```

60 FOR I = 0 TO SIZE	
70 FLAGS(I) = 1	
80 NEXT I	
90 FOR I = 0 TO SIZE	
100 IF FLAGS(I) = 0 THEN 180	
110 PRIME = I+I +3	
120 K = I + PRIME	
130 IF K > SIZE THEN 170	
140 FLAGS(K) = 0	
150 K = K + PRIME	
160 GOTO 130	
170 COUNT = COUNT + 1	
180 NEXT I	
190 PRINT COUNT;" PRIMES"	
E *(RETURN)*	Enter mode, allows text entry.
2 J$ = TIME$	Two lines are added to print the time.
200 PRINT J$, TIME$	
(Escape, Return)	Type Escape key to exit Enter mode.
QU (Return)	Finished changes. Leave editor and return to TBASIC.
- DO (Return)	The file is recompiled with the DO command, taking about 0.2 second.
- RUN (Return)	The program is run again with changes.
1 ITERATION	The program produces output.
1899 PRIMES	
01:01:25 01:01:27	
-	The prompt returns after execution ends.
-// (Return)	Exit TBASIC.
:// (Return)	Exit the Trump Card system.
B>*BASICA (Return)*	Get BASICA and run SIEVE.S.
LOAD "SIEVE.S"	(SIEVE.S was stored in ASCII format.)
RUN	
1 ITERATION	The program produces output.
1899 PRIMES	
01:05:35 01:09:01	

EXAMPLE 4

Computer Interaction	Comments
B>*G (Return)*	Activate the Trump Card.
: BASIC (Return)	Enter TBASIC.
- /DIAG (Return)	Invoke subroutine-diagnostic mode.
- PRINT 2+3 (Return)	Directly add and print 2+3.
CExit:CImmxInit:Ki00000000; CPrtInit;Ki00000002;Ki00000003; b+;CPrtI;CPrtCR;R; 5	The listing shows the compiler subroutines that are executed to perform the function. CExit (call exit) jumps out of the console-input mode; CImmxInit calls for immediate execution with a flag integer-constant value of 0 set as Ki00000000. CPrtInit (call printer) directs printing to the console; the two integer values are expressed as Ki00000002 and Ki00000003, respectively; b+ calls a binary add routine; CPrtI prints the integer. CPrtCR finishes by sending a carriage return to the printer or console while R designates a return to the system. The computed value, 5, appears at the end.
- PRINT2.027+3.094 (Return)	Floating-point values produce a slightly different result.
CExit;CImmxInit;Ki00000000; CPrtInit;Kf01BA5E82;Kf46041982; CFltAdd;CPrtF;CPrtCR;R; 5.121	This time the constants are stored as floating-point numbers, and floating-point add and print routines are called instead.
-	

EXAMPLE 5

:	Back in command interpreter.
:C (Return)	Call C compiler, the "-" is
-	the C compiler prompt.
-/DO BASICIO.C (Return)	Compile I/O routines.
-/DO CDEMO.C (Return)	Compile CDEMO.C program (listing 3).
-/IMAGE CDEMO E=MAIN (Return)	Save memory image of compiled program in a disk file called CDEMO.
-// (Return)	Get out of C compiler.
:	Back in command interpreter.
:CDEMO (Return)	Run compiled program.
C language	
C language	The program produces output.
C language	
C language	
C language	
:	Back in command interpreter.
:// (Return)	Get out of interpreter.
B>	Back to IBM PC-DOS command prompt.

Control-C. If possible, TBASIC will display the statement label nearest the point in the program where the stop occurred. Programs may contain STOP statements and may be restarted by a CONT command.

TBASIC also can execute statements and commands in immediate mode. You simply type the program line without a line number. (If you precede a statement with a line number, it will be compiled into the existing program.) You can get results like

```
-PRINT SQR(2)
1.414214
-
-PRINT 2*3
6
-
```

You can print out variables or run specific program lines that contain line-identifier labels. Immediate-mode statements and commands also may be included in program files.

TBASIC also has some commands useful in debugging and problem diagnosis that you probably have not seen before. You can examine the actual compiled machine-language object code with commands like /DIAG. If you give the /DIAG command before a program is compiled, a complete list of compiler subroutine calls will be produced. This can be demonstrated in the direct-execution immediate mode, as shown in example 4 for both integer and floating-point values.

C Compiler

For more ambitious program development, the Trump Card also supports a compiler for programs in the C language, as described by Kernighan and Ritchie (see reference 6). Programs need

The Trump Card also supports a compiler for programs written in the C language.

only slight modifications for compilation. Developing and running a C program is a three-step operation similar to the process used in TBASIC: editing, compiling, and running.

(text continued on page 120)

(text continued from page 119)

C compilers expect to find input and output routines in a subroutine library separate from the compiler. Kernighan and Ritchie describe a file called "stdio.h" that contains the I/O facilities. The Trump Card's C compiler uses a file of I/O routines called "basicio.c", which includes the following routines: "getchar", "putchar", "open", "close", "read", "write", "printf", "scanf", "lseek", and "creat".

The implementation of "scanf" and "printf" in the Trump Card's version of C differs slightly from that of Kernighan and Ritchie. In their implementation, the conversion characters "d" and "x" may each be preceded by an "l" to indicate a pointer to a "long" value rather than a pointer to an "int" value appears in the argument list. In this implementation, the uppercase conversion characters "D" and "X" are used for the same purpose. The conversion character "f" is used for floating point. The "scanf" routine assumes that the input values are separated by Space or Tab characters and that a Return character ends an input sequence.

The Trump Card's C compiler was designed with a user interface similar to that of TBASIC, and it's just as easy to use. Listing 3 shows a C program that is entered into the system using the EE editor in a manner such as that used for TBASIC. Example 5 shows how the program is compiled and run. Should you care to try the Sieve program in C, it is shown in listing 4 set up for 10 iterations. It runs in 3.2 seconds on the Trump Card, which compares quite favorably with versions of C running on 8-MHz MC68000 processors and with assembly-language versions on the IBM's 4.77-MHz 8088.

Y MULTILEVEL LANGUAGE

The Y language system compiles a multilevel language that can be best described as structured assembler code. It allows you to write programs using a mixture of Z8000 assembly language (in Zilog mnemonics), Pascal-like control structures, data types, arithmetic expressions with automatic or specified allocation of registers, procedure calls with parameter passing, and a descriptive compiler language. The different levels of constructs may, for the most part, be freely mixed.

The Y compiler generates code directly into memory with one pass and supports immediate execution of statements, conditional compilation, user-defined extensions to the language, and symbolic debugging. Most of the Z8000

Table 1: *Keywords for statements and functions available in the TBASIC compiler for the Trump Card. An asterisk indicates a new feature.*

Function	Statement	Command	Variable
ABS	BEEP	ALLOCATE*	CSRLIN
ASC	CALL	BLOAD	DATE$
ATN	CLOSE	BSAVE	ERR
CALLINT$*	CIRCLE	CONT	INKEY$
CDBL	CLS	DIAG*	TIME$
CHR$	COLOR	DISP*	
CINT	DATA	DO*	
COS	DATE$	EDIT	
CSNG	DEF FN	KILL	
CVI	DEF SEG	LIST	
CVS	DEFtype	MAP*	
CVD	DIM	NAME	
EOF	END	NEW	
EXP	FIELD	REGIONS*	
FIX	FOR...NEXT	REGS*	
HEX$	GET	RESET	
INP	GOSUB	RUN	
INPUT$	GOTO	SAVE	
INSTR	IF	SYSTEM	
INT	INPUT		
LEFT$	INPUT#		
LEN	LSET		
LOC	LET		
LOF	LINE		
LOG	LINE INPUT		
LPOS	LINE INPUT#		
MID$	LOCATE		
MKI$	LPRINT		
MKS$	LPRINT USING		
MKD$	ON ERROR		
OCT$	ON GOSUB		
PEEK	ON GOTO		
POINT	OPEN		
POS	OUT		
RIGHT$	PAINT		
RND	POKE		
SCREEN	PRINT		
SGN	PRINT USING		
SIN	PRINT#		
SPACE	PRINT# USING		
SPC	PSET		
SQR	PUT		
STR$	PRESET		
STRING$	RANDOMIZE		
TAB	READ		
TAN	REM		
VAL	RESTORE		
	RESUME		
	RETURN		
	RSET		
	SCREEN		
	SEEK*		
	SOUND		
	STOP		
	TIME$		
	WAIT		
	WHILE...WEND		
	WIDTH		
	WRITE		
	WRITE#		

Table 2: *Comparison of Sieve benchmark results (one iteration) on other computers running Microsoft-derived BASIC interpreters (times measured in seconds).*

Apple II	Apple III	TRS-80 Model II	IBM PC (BASICA)	IBM PC (TBASIC with Trump Card)
224	222	189	206	2.4

Table 3: *Execution time in seconds for the looping program of listing 1 on several interpreters.*

Apple II	IBM PC (CBASIC-86)	IBM PC (BASICA)	IBM PC (TBASIC with Trump Card)
101	275	80	0.9

Table 4: *A listing of the standard CP/M-80 2.2 functions. Those marked with an asterisk are supported by the Trump Card Z80 emulator.*

Function	Supported?
0 System Reset	*
1 Console Input	*
2 Console Output	*
3 Reader Input	
4 Punch Output	
5 List Output	*
6 Dir Console I/O	*
7 Get I/O Byte	
8 Set I/O Byte	
9 Print String	*
10 Read Con Buffer	*
11 Console Status	*
12 Version Number	*
13 Reset Disk Sys	*
14 Select Disk	*
15 Open File	*
16 Close File	*
17 Search For 1st	*
18 Search For Next	*
19 Delete File	*
20 Read Sequential	*
21 Write Sequential	*
22 Make File	*
23 Rename File	*
24 Login Vector	*
25 Current Disk	*
26 Set DMA Address	*
27 Get Alloc Addr	
28 Write Protect	
29 Get R/O Vector	*
30 File Attributes	
31 Disk Params Addr	
32 User Codes	
33 Read Random	*
34 Write Random	*
35 Comp File Size	*
36 Set Random Rec	*

mnemonics are implemented; those that are not can be used via the WORD pseudo-operation, as in the following: LDCTL REFRESH,R3 = WORD 07D3B.

The TBASIC and C compilers are written in Y. Each of the compiler subroutines is a Y file that has been compiled into assembly-language code. A full explanation of Y is beyond the scope of this article, but listing 5 shows some Y code for your inspection. Y is an advanced tool for the experienced programmer.

CP/M-80 Emulator

The Trump Card supports a software emulator for CP/M-80 version 2.2, which allows the Trump Card to execute assembly-language programs for the 8-bit Z80 microprocessor.

The Z80 program must be transferred to a PC-DOS (or MS-DOS) floppy disk. (This can be done by linking a Z80-based computer and an IBM PC through a serial RS-232C connection, either through a direct cable or through a modem.) Once the Z80 program is on the IBM-format disk, its filename extension must be changed from ".COM" to ".CMD", which is consistent with the CP/M-86 convention and avoids the problem of trying to run a Z80 program under IBM PC-DOS.

The emulator normally resides on a disk in drive B and is used in a manner very much like that of the other Trump Card software we've looked at. Nearly all the normal CP/M-80 system calls are supported by the emulator, with a few exceptions as shown in table 4. The standard CP/M-80 BIOS (basic input/output system) calls dealing with the disk, punch, and reader devices are not supported by the Z80 emulator; the remaining BIOS calls are supported.

In Conclusion

The Trump Card is a board-level hardware approach to upgrading the performance of your IBM PC (or a compatible system). Aside from its function as a

(text continued on page 122)

Listing 1: *A simple FOR...NEXT loop benchmark program in BASIC.*

```
100 FOR A=1 TO 10
115 FOR J=1 TO 10
120 FOR T=0 TO 200
130 GOSUB 200
140 B=1
150 NEXT T
155 NEXT J
160 NEXT A
170 PRINT "DONE"
200 RETURN
```

Listing 2: *A simple BASIC benchmark program for floating-point division.*

```
60 A=2.71828
80 B=3.14159
100 FOR I=1 TO 5000
120 C=A/B
320 NEXT I
```

Listing 3: *A demonstration program for the C compiler.*

```
main()
{
   int count,step;
   count=1;
   step=1;
   while (count <= 5)
      {
      printf(" C language\n");
      count=count+step;
      }
}
```

(text continued from page 121)
Z8000 development system, it provides many popular system enhancements in a single package: add-on memory, execution of Z80 programs, a separate editor, and language compilers. It was designed to solve my specific personal problem—I wanted a better BASIC that wasn't slow or cumbersome—and to support the PC in other ways: as a language and RAM-disk peripheral. If you're like me, these characteristics will be the most important ones to you.

In the process of building the Trump Card, however, I've found that it has potential I never imagined. Besides the software I've described, I expect that object-code translators for Z80-to-Z8000 and 8088-to-Z8000 conversions will soon be available, along with other utilities such as a print spooler. You also eventually will see Bell Laboratories' UNIX operating system for the Trump Card.

NEXT MONTH

Whimsy is in vogue, as Steve designs a musical telephone bell. ■

Z8000 and Z80 are trademarks of Zilog Corporation, a subsidiary of Exxon. CP/M-80 is a trademark of Digital Research.

To receive a complete list of Ciarcia's Circuit Cellar project kits available, circle 100 on the reader-service inquiry card at the back of the magazine.

Listing 4: *The Sieve of Eratosthenes benchmark in C.*

```
#define true 1
#define false 0
#define size 8190
#define sizepl 8191
    char flags[sizepl];
main(){
    register int i,prime,k,count,iter;
     printf("10 iterations\n");
    for (iter = 1;iter <= 10;iter ++){
        count=0;
        for(i = 0;i <= size;i ++)
            flags[i] = true;
        for(i = 0;i <= size;i ++){
            if(flags[i]){
                prime = i + i + 3;
                k = i + prime;
                    while(k <= size){
                    flags[k] = false;
                        k += prime;
                        }
                    count = count + 1;
                }
            }
        }
    printf("\n%d primes",count);
}
```

Listing 5: TBASIC *subroutines written on the* Y *multilevel-language compiler.*

```
(5a)
if SWITCH=0 or CNT>100 then begin
SWITCH:=1; GODOIT(2, VAL&OF)
end
else begin
  R3:=^ABC; R5:=@R9(2); R1:=CNT/2
  LDIR @R3,@R5,R1
end

(5b)
COLOR: PROC ...passed flag, then other params
depending on flag
  ...if flag bit 2=1, then set border color (if text
mode)
  ...if bit 1=1, set background color (text) or
palette (graphics)
  ...if bit 0=1, set foreground color (text) or
background color (graphics)
  save R6,R7
  POPL RR6,@RR12
  if BIT R7,2 not zero then begin
    POPL RR2,@RR12
    if SCRMODE<=1 then SETBORDER(R3)
  end
  if BIT R7,1 not zero then begin
    POPL RR2,@RR12
    if R0:=SCRMODE<=1 then SETBG(R3) else
if R0=2 then
SETPALET(R3)
  end
  if BIT R7,0 not zero then begin
    POPL RR2,@RR12
    if R0:=SCRMODE<=1 then SETFG(R3) else
if R0=2 then
SETGRAPHBG(R3)
  end
  restore R6,R7
  RET
SOUND: PROC ...passed duration
(in 1/18.2 secs) and frequency
  ...make sound
  POPL RR4,@RR12  ...duration
  POPL RR2,@RR12  ...frequency
  EXB RL3,RH3; EXB RL5,RH5
  R3:->BX; R5:->CX
  AH:=4  ...sound
  EXTCALL(SPSCRINT)
  RET
```


REFERENCES

1. Brown, Peter J. *Writing Interactive Compilers and Interpreters.* New York: John Wiley & Sons, 1979.
2. Ciarcia, Steve. "Trump Card, Part 1: Hardware." BYTE, May 1984, page 40.
3. George, Donald P. "Professional BASIC." BYTE, April 1984, page 334.
4. Gilbreath, Jim. "A High-Level Language Benchmark." BYTE, September 1981, page 180.
5. Gilbreath, Jim, and Gary Gilbreath. "Eratosthenes Revisited: Once More through the Sieve." BYTE, January 1983, page 283.
6. Kernighan, Brian W., and Dennis M. Ritchie. *The C Programming Language.* New York: Prentice-Hall, 1978.
7. Lee, J. A. N. *The Anatomy of a Compiler*, 2nd ed. New York: Van Nostrand Reinhold, 1974.
8. Mello-Grand, Sergio. "The Docutel/Olivetti M20: A Sleek Import." BYTE, June 1983, page 188.

The following items are available from

Sweet Micro Systems Inc.
50 Freeway Dr.
Cranston, RI 02910
(800) 341-8001 for orders
(401) 461-0530 for information

1. Trump Card, including IC sockets, assembled and tested with 256K bytes of the 512K-byte RAM space populated. Includes TBASIC compiler, C compiler, Z8000 Y assembler, CP/M-80 emulator, RAM-disk driver, and documentation. Software supplied on a PC-DOS 2.0 disk unless otherwise specified.
256TCB $995
2. Trump Card, printed-circuit board completely socketed, assembled, and tested with 512K bytes of RAM, support software described above, and documentation. Software supplied on a PC-DOS 2.0 floppy disk unless otherwise specified.
512TCC $1325
3. Trump Card partial kit, completely socketed and wave-soldered with all passive components, less ICs but including bootstrap loader EPROMs, 10-MHz Z8001, and Z8581. Includes support software described above on PC-DOS 2.0 floppy disk (unless otherwise specified) and documentation.
OKTCA $525

Please add $10 for shipping and insurance in continental United States, $20 elsewhere. Rhode Island residents please include 6 percent sales tax.

Editor's Note: Steve often refers to previous Circuit Cellar articles. Most of these are available in reprint books from BYTE Books, McGraw-Hill Book Company, POB 400, Hightstown, NJ 08250.

Ciarcia's Circuit Cellar, Volume I covers articles that appeared in BYTE from September 1977 through November 1978. *Ciarcia's Circuit Cellar, Volume II* contains articles from December 1978 through June 1980. *Ciarcia's Circuit Cellar, Volume III* contains articles from July 1980 through December 1981. *Ciarcia's Circuit Cellar, Volume IV* contains articles from January 1982 through June 1983.

Circle 384 on inquiry card. →