by pulling and pushing it to and from the data stack. When a command is given to manipulate data, that data had better be on the stack already. To add 1 and 2, we must say 1 2 + so as to place the two numbers on the stack first. This type of notation is called Reverse Polish (or Postfix) notation and seems clumsy at first; but a little practice will make it seem quite natural.

Values in the data stack are usually limited to 16 bit signed integers. That means that numbers can be integers in the range of -32768 to +32767. You can't fit the value of pi into an integer, nor can you represent your annual salary in pennies unless you're unusually poor. And what about arrays? You might have trouble fitting marks for 300 students on the stack. And, if you made it somehow, you'd have a devil of a time getting the particular one you wanted. And we haven't even mentioned how to deal with strings...

Because of such limitations, FORTH quickly leaves its simple form and starts to demand considerable skills of the programmer. He must be able to allocate memory space and set up sets of indirect pointers that will steer him to the particular unit of data he needs. He will need to be able to handle floating point by building special commands; in many cases this feature is at least partly provided by the vendor.

The beginner is faced with a huge vocabulary of commands, most of which he will need to learn. Not everyone will have the patience to slug through this in order to develop competence in FORTH. When he finds he needs to handle indirect pointers and tables, he'll need to have an aptitude for this kind of thing. FORTH demonstrations can be misleading; the language seems to be so easy when a few simple things are shown.

For those who take the time and trouble to develop FORTH competence, the payoff can be high: fast-running code that can be written quickly. But the beginner must realize that it's not all easy sailing; FORTH won't help you along in the same way that BASIC does.

Advocates of structured programming tend to be suspicious of FORTH. Since FORTH encourages to build upward from the detailed code to the total job, it is considered a "bottom up" type of language. Many computer scientists would prefer to see you go the other way: from the top – the big picture – down into increasing levels of details, or "top down" programming.

It's a language that excites many users. For others, it may be tough sledding and too far off the mainstream of small computer activities. Those who are hooked on FORTH become fanatics: they insist that a job is well done only if it's FORTH right. ©

Part I of this three-part machine language monitor for the OSI Superboard appeared in March, 1982, issue #22.

Part II:

A Superboard II Monitor

Frank Cohen
Pacific Palisades, CA

In the March issue of **COMPUTE!** we presented the first part of a fairly complex program to add a sophisticated "monitor" program to the Superboard II. A monitor does nothing more than to peek into the machine's memory and enter, display, move, or store data in the form of hexadecimal bytes.

Stored in the ROM memory on the Superboard is OSI's monitor program. When originally designing the Superboard, a microcomputer called the KIM-1 was selling well in the microcomputer market. The OSI monitor largely resembles the monitor for the KIM-1. The KIM-1 had a six digit display and a hexadecimal (base 16) keypad plus some other keys which had specific functions devoted to each. With the six digit display, there was room to display a two byte address and the contents of that memory location.

There were two modes of operation: the address mode, and data mode. In the address mode, a key pressed was rotated into the current address being displayed. By rotating the key in, the existing address digits are all shifted left one position (the left-most digit was lost) and the new key pressed is put into the right-most digit. The same kind of scheme is used for entering data in the data mode. However, instead of changing the address digits, the contents of that location are changed.

Changing from the data to the address mode (or vice versa) is accomplished by pressing the AD key, or the DA key. The Superboard II uses the period (.) key instead to enter the address mode and the comma (,) key for the data mode. This system works well for the KIM-1 considering that it cost about $175 and did not have a video display or an advanced keyboard as the Superboard does.

Of course, the monitor program for the Superboard only occupies a small fraction of the space that Super-Monitor uses. However, if you start using your Superboard more and more, you normally will learn how to program in machine language. Possibly blocking your move into the wonderful world of machine language is the resident monitor program.

Last month we outlined what the capabilities of the Superboard II's new monitor program should include. Top on the list was the ability to look into memory, a group of locations at a time. Second, we wanted to be able to modify the Superboard's memory and, at the same time, see what we just modified. Third, we want to fill a block of memory with some value. Next, we want to be able to move a whole block of memory from one location to another. Finally, we'll need an intelligent cassette interface routine for storing and retrieving blocks of memory.

Since Super-Monitor is over 500 bytes long, it has been split into sections. Last month's issue presented the listing for a program called HEXDUMP. HEXDUMP was listed first since most of the other routines in Super-Monitor use its subroutines. When looking at the listings of the individual programs, you will find that they are each mini-programs. The start of each listing also tells what other programs (subroutines) are needed to make it work. The logic behind the listing's structure lies in the fact that loading Super-Monitor in its entirety takes about five minutes with the Superboard's slow cassette interface. By loading just the routines that you want, Super-Monitor can be customized.

HEXDUMP fills the screen of the Superboard with data from memory, eight bytes at a time. HEXDUMP, like most of the routines in Super-Monitor, uses a program called Super-Cursor V1.3 (**COMPUTE!** December, 1981, #19, pp. 124-128) to handle its video output. To use Super-Cursor V1.3, a program puts the ASCII character in the CPU's accumulator and executes a jump-to-subroutine (JSR) to the start address of Super-Cursor, 1E40 (Hex). Super-Cursor also is used to clear the screen, address 1EC2 (Hex), and to home the cursor, address 1E80 (Hex). If you don't want to use Super-Cursor, you will have to write your own video output routine. If you want Super-Cursor and Super Monitor you can send a blank cassette and $3 to the address below and I will copy it for you.

The main subroutines from HEXDUMP that the other routines use are called INADR and PLINE. INADR, starting at address 1D93 (Hex), inputs a two byte address from the keyboard and echoes it to the video screen. The resulting address is stored in address 00E7, called ADR, and 00E8. PLINE is used to print a row of eight bytes of data on the screen. The beginning address is located in ADR, 00E7 and 00E8.

INDATA

The first program in this issue is called INDATA. This program is approximately 199 bytes long and allows the user to look into, and modify, any group of memory locations. Entering machine language programs is simple using INDATA. In fact, after writing HEXDUMP, and Super-Cursor V1.3, I used INDATA to enter the other routines. It is fast and efficient.

INDATA shows the programmer a line of eight bytes of data at a time. Preceding the data is the address of the left-most byte of data. A greater-than sign (>) is placed next to the currently "open" memory location. Any hexadecimal key you hit will be rotated into that byte. When you have finished changing the contents of the current memory location, you can move the greater-than sign to the next location (one space right) by pressing the SPACE bar. Or, you can go back to the last location (one space left) by pressing the RUB-OUT key. If you think that you made a mistake just look up at the screen and compare.

If you are at the right-most byte on a row, the next time you hit the space bar the next line of eight bytes will be displayed. The opposite is true for typing a Rub-Out when you are at the left-most byte. When you are finished entering data, pressing the RETURN key will exit the program. In the listing, when you press the RETURN key, the program will go back into OSI's ROM monitor program.

Program 1 is a complete assembly listing of INDATA. As it is listed, it fits right under HEXDUMP on an 8K Superboard II. I do not suggest trying to move INDATA to another part of memory as it uses many absolute addresses which would have to be modified. However, if you don't have an assembler, it is possible to move it. (This is your encouragement to get a more complex system.) If your Superboard has only the original 4K bytes of RAM, I suggest you add some 2114's.

BMOVE

BMOVE is short for Block Move Routine. As the name implies, this routine is set up to move any size block of memory from one location to another. This is especially handy if you have entered a long program and found that you accidentally started at the wrong location. Another application is looking into the ROM's on your Superboard. By telling BMOVE to move the beginning of the BASIC-in-ROM, located at A000, to the memory mapped video area you can see the internal organs of BASIC.

To use BMOVE, you enter the program at location 1BC6 (Hex). The program first asks you for the starting location of the block to be moved by printing "S=" on the screen. Then it asks you for the ending address by printing "E=" on the screen. (No, it is not asking for Einstein's Theory of Relativity.) Finally, BMOVE prompts you to enter the beginning destination address by printing "D=."

BMOVE is very fast. You will find that it can move a block 8K long in about a second. The majority of BMOVE's program listing is devoted to inputting the three addresses. After it has those addresses loaded, BMOVE calculates the last address of the destination. It then proceeds to move the block, byte by byte, from the top down. For every byte it moves, it will decrement the ending address and check to see if it is equal to the starting address. When the two are equal, it will return to OSI's ROM monitor. Again, later, we will modify the program to return to Super-Monitor's main menu routine.

In the third and final installment, next month, the listings will be described and listed. So far we have enough to call this an advanced monitor routine. The three programs, HEXDUMP, INDATA, and BMOVE, allows you to look at, modify, and move, data in very simple steps.

*These routines make extensive calls to HEXDUMP and SUPER-CURSOR V1.3. It also changes SUPER-CURSOR "system variables," such as cursor position. If you want to use INDATA and BMOVE without HEX-DUMP and SUPER-CURSOR, you will need to refer to the listings of SUPER-CURSOR (***COMPUTE!*** *February, 1982, #21) and HEXDUMP (***COMPUTE!*** *December, 1982, #14). Zero page usage:$E7-$ED*

Program 1: INDATA

```
1C56 20 80 1E A9 41 20 40 1E
1C5E A9 3D 20 40 1E 20 96 1D
1C66 A9 00 85 EA A9 3E 8D 61
1C6E 1F 20 80 1E 20 00 1E AE
1C76 CC D0 20 80 1E 86 E2 20
1C7E FB 1E A6 EA 20 FB 1E 20
1C86 FB 1E 20 FB 1E E0 00 F0
1C8E 04 CA 4C 82 1C A5 E7 38
1C96 E9 08 85 E7 B0 02 C6 E8
1C9E A4 EA B1 E7 85 E9 20 BA
1CA6 FF C9 0D D0 08 A9 A0 8D
1CAE 61 1F 4C 43 FE C9 20 D0
1CB6 03 4C D8 1C C9 7F D0 03
1CBE 4C F8 1C 20 F3 1D 8D CF
1CC6 1C A5 E9 0A 0A 0A 0A 18
1CCE 69 00 85 E9 20 15 1D 4C
1CD6 6F 1C 20 15 1D A5 EA C9
1CDE 07 D0 12 A9 00 85 EA A5
1CE6 E7 18 69 07 85 E7 90 02
1CEE E6 E8 4C 6F 1C E6 EA 4C
1CF6 6F 1C 20 15 1D 98 D0 12
1CFE A9 07 85 EA A5 E7 38 E9
1D06 08 85 E7 B0 02 C6 E8 4C
1D0E 6F 1C C6 EA 4C 6F 1C A4
1D16 EA A5 E9 91 E7 60 AA AA
```

Common routines:

1C56	**INDATA**	**Entry point for INDATA program**
1C66	**BLOOP**	**Main loop start for INDATA**
1C6F	**BPCS**	**Print a line and fix SUPER-CURSOR bug**
1C80	**SKIP**	**Positions cursor to current open cell**
1C93	**CKSP**	**Fix HEXDUMP bug by adding $08 to ADR**
1C9E	**OPCELL**	**Load BYTE with current open cell**
1CA4	**KEY**	**Decodes key pressed and jumps to routine**
1CC1	**ROTIN**	**Rotates key pressed value into current cell**
1CD8	**GNCELL**	**Open next cell**
1D15	**CLCELL**	**Close last cell**

Program 2: BMOVE

```
1BC6 20 80 1E A9 53 20 40 1E
1BCE A9 3D 20 40 1E 20 96 1D
1BD6 A5 E7 85 EB A5 E8 85 EC
1BDE 20 95 1E 20 AB 1E A9 45
1BE6 20 40 1E A9 3D 20 40 1E
1BEE 20 96 1D A5 E7 85 E9 A5
1BF6 E8 85 EA 20 95 1E 20 AB
1BFE 1E A9 44 20 40 1E A9 3D
1C06 20 40 1E 20 96 1D A5 E9
1C0E 38 E5 EB 85 ED A5 EA E5
1C16 EC 48 A5 ED 18 65 E7 85
1C1E E7 68 65 E8 85 E8 A0 00
1C26 B1 E9 91 E7 A5 EB C5 E9
1C2E D0 09 A5 EC C5 EA D0 03
1C36 4C 43 FE A5 E9 38 E9 01
1C3E 85 E9 B0 02 C6 EA A5 E7
1C46 38 E9 01 85 E7 B0 02 C6
1C4E E8 4C 24 1C EA EA EA
```

Common routines:

1BC6	**BMOVE**	**Inputs starting location of block to be moved**
1BDE	**INELOC**	**Inputs ending location of block to be moved**
1BF9	**INDADR**	**Inputs destination address of block to be moved**
1B0C	**CALC**	**Calculates ending address of destination block**
1C24	**MOVIT**	**Moves a byte from EBAD to DBAD**
1C2A	**CKFIN**	**Checks to see if we're finished**
1C39	**NFIN**	**Decrements two byte registers EBAD and DBAD**

©

OSI TRS-80 COLOR-80 OSI

GALAXIAN - 4K - One of the fastest and finest arcade games ever written for the OSI, this one features rows of hard-hitting evasive dogfighting aliens thirsty for your blood. For those who loved (and tired of) Alien Invaders. Specify system — A bargain at $9.95 OSI

LABYRINTH - 8K - This has a display background similar to MINOS as the action takes place in a realistic maze seen from ground level. This is, however, a real time monster hunt as you track down and shoot mobile monsters on foot. Checking out and testing this one was the most fun I've had in years! — $13.95. OSI

THE AARDVARK JOURNAL

FOR OSI USERS — This is a bi-monthly tutorial journal running only articles about OSI systems. Every issue contains programs customized for OSI, tutorials on how to use and modify the system, and reviews of OSI related products. In the last two years we have run articles like these!

1) A tutorial on Machine Code for BASIC programmers.
2) Complete listings of two word processors for BASIC IN ROM machines.
3) Moving the Directory off track 12.
4) Listings for 20 game programs for the OSI.
5) How to write high speed BASIC — and lots more —

Vol. 1 (1980) 6 back issues - $9.00

Vol. 2 (1981) 4 back issues and subscription for 2 additional issues - $9.00.

ADVENTURES!!!

For OSI, TRS-80, and COLOR-80. These Adventures are written in BASIC, are full featured, fast action, full plotted adventures that take 30-50 hours to play. (Adventures are interactive fantasies. It's like reading a book except that you are the main character as you give the computer commands like "Look in the Coffin" and "Light the torch".)

Adventures require 8K on an OSI and 16K on COLOR-80 and TRS-80. They sell for $14.95 each.

ESCAPE FROM MARS (by Rodger Olsen)

This ADVENTURE takes place on the RED PLANT. You'll have to explore a Martian city and deal with possibly hostile aliens to survive this one. A good first adventure.

PYRAMID (by Rodger Olsen)

This is our most challenging ADVENTURE. It is a treasure hunt in a pyramid full of problems. Exciting and tough!

TREK ADVENTURE (by Bob Retelle)

This one takes place aboard a familiar starship. The crew has left for good reasons - but they forgot to take you, and now you are in deep trouble.

DEATH SHIP (by Rodger Olsen)

Our first and original ADVENTURE, this one takes place aboard a cruise ship - but it ain't the Love Boat.

VAMPIRE CASTLE (by Mike Bassman)

This is a contest between you and old Drac - and it's getting a little dark outside. $14.95 each.

OSI NEW-NEW-NEW OSI TINY COMPILER

The easy way to speed in your programs. The tiny compiler lets you write and debug your program in Basic and then automatically compiles a Machine Code version that runs from 50-150 times faster. The tiny compiler generates relocatable, native, transportable machine code that can be run on any 6502 system.

It does have some limitations. It is memory hungry — 8K is the minimum sized system that can run the Compiler. It also handles only a limited subset of Basic — about 20 keywords including FOR, NEXT, IF THEN, GOSUB, GOTO, RETURN, END, STOP, USR(X), PEEK, POKE, -, =, *, /, , <,> , Variable names A-Z, and Integer Numbers from 0-64K.

TINY COMPILER is written in Basic. It can be modified and augmented by the user. It comes with a 20 page manual.

TINY COMPILER — $19.95 on tape or disk OSI

SUPERDISK II

This disk contains a new BEXEC* that boots up with a numbered directory and which allows creation, deletion and renaming of files without calling other programs. It also contains a slight modification to BASIC to allow 14 character file names.

The disk contains a disk manager that contains a disk packer, a hex/dec calculator and several other utilities.

It also has a full screen editor (in machine code on C2P/C4)) that makes corrections a snap. We'll also toss in renumbering and program search programs — and sell the whole thing for —

SUPERDISK II $29.95 (5¼") OSI

BARE BOARDS FOR OSI C1P

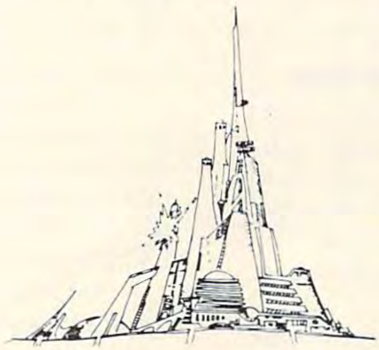
MEMORY BOARDS!!! — for the C1P — and they contain parallel ports!

Aardvarks new memory board supports 8K of 2114's and has provision for a PIA to give a parallel ports! It sells as a bare board for $29.95. When assembled, the board plugs into the expansion connector on the 600 board. Available now!

PROM BURNER FOR THE C1P — Burns single supply 2716's. Bare board — $24.95.

MOTHER BOARD — Expand your expansion connector from one to five connectors or use it to adapt our C1P boards to your C4/8P. - $14.95.

16K RAM BOARD FOR C1P — This one does not have a parallel port, but it does support 16K of 2114's. Bare Board $39.95.



WORD PROCESSING THE EASY WAY— WITH MAXI-PROS

This is a line-oriented word processor designed for the office that doesn't want to send every new girl out for training in how to type a letter.

It has automatic right and left margin justification and lets you vary the width and margins during printing. It has automatic pagination and automatic page numbering. It will print any text single, double or triple spaced and has text centering commands. It will make any number of multiple copies or chain files together to print an entire disk of data at one time.

MAXI-PROS has both global and line edit capability and the polled keyboard versions contain a corrected keyboard routine that make the OSI keyboard decode as a standard typewriter keyboard.

MAXI-PROS also has sophisticated file capabibilities. It can access a file for names and addresses, stop for inputs, and print form letters. It has file merging capabilities so that it can store and combine paragraphs and pages in any order.

Best of all, it is in BASIC (0S65D 51/4" or 8" disk) so that it can be easily adapted to any printer or printing job and so that it can be sold for a measly price.

MAXI-PROS — $39.95. Specify 5¼ or 8" disk.

SUPPORT ROMS FOR BASIC IN ROM MACHINES — C1S/C2S. This ROM adds line edit functions, software selectable scroll windows, bell support, choice of OSI or standard keyboard routines, two callable screen clears, and software support for 32-64 characters per line video. Has one character command to switch model 2 C1P from 24 to 48 character line. When installed in C2 or C4 (C2S) requires installation of additional chip. C1P requires only a jumper change. — $39.95

C1E/C2E similar to above but with extended machine code monitor. — $59.95 OSI

ARCADE GAMES FOR OSI, COLOR-80 AND TRS-80 (8K OSI, 16K TRS-80 AND COLOR-80)

TIMETREK - A REAL TIME, REAL GRAPHICS STARTRECK. See your torpedoes hit and watch your instruments work in real time. No more unrealistic scrolling displays! $14.95.

STARFIGHTER - This one man space war game pits you against spacecruisers, battlewagons, and one man fighters, you have the view from your cockpit window, a real time working instrument panel, and your wits. Another real time goody. $9.95

BATTLEFLEET - This grown up version of Battleship is the toughest thinking game available on OSI or 80 computers. There is no luck involved as you seek out the computers hidden fleet. A topographical toughie. $9.95

QUEST - A NEW IDEA IN ADVENTURE GAMES! Different from all the others, Quest is played on a computer generated mape of Alesia. Your job is to gather men and supplies by combbat, bargaining, exploration of ruins and temples and outright banditry. When your force is strong enough, you attack the Citadel of Moorlock in a life or death battle to the finish. Playable in 2 to 5 hours, this one is different every time.

16K COLOR-80 OR TRS-80 ONLY. $14.95

Please specify system on all orders

This is only a partial listing of what we have to offer. We offer over 120 games, ROMS, and data sheets for OSI systems and many games and utilities for COLOR-80 and TRS-80. Send $1.00 for our catalog.

AARDVARK TECHNICAL SERVICES, LTD.
2352 S. Commerce, Walled Lake, MI 48088
(313) 669-3110

OSI **COLOR-80**