Cromemco Trace System Simulator

PREFACE—Trace is a powerful system simulator designed to facilitate Assembly Language Program Development on Cromemco Computer Systems. Virtually all aspects of System Operation can be simulated including Prioritized Interrupts and I/O Commands. A historical record of Program Execution is maintained in a 100-Instruction Circular Queue. The advanced features of Trace enable it to be used in place of Logic analyzers or In-circuit Emulators in Program Development.

Cromemco
incorporated
Specialists in computers and peripherals
280 Bernardo Ave., Mountain View, CA 94041

Cromemco Trace System Simulator

Instruction Manual

Five Dollars

Table of Contents

Chapter 1

Introduction To Trace

Introduction

The Cromemco TRACE program enables the user to test and debug programs. TRACE emulates the behavior of a Z-80 processor as it follows the logic of the user program, providing a selection of reports of the processor state so as to minimize extraneous printing.

TRACE operates as part of DEBUG, Cromemco's assembly language debugging program. DEBUG is automatically loaded along with TRACE, and acts as a monitor for TRACE. In this manual the terms "monitor" and "DEBUG" are used interchangeably.

TRACE Options include control of register display, choice of display frequency, i.e. after every instruction, after every branch, at CDOS calls only, and with register history written to a circular queue for display after returning to the DEBUG monitor.

Features which help the user locate errors quickly include warnings if the user writes to unexpected areas, simulation of Input/Output commands on the console, warnings of attempts to execute undefined commands, undefined calls to CDOS routines, improper return from subroutine calls, and execution of branch instructions or decimal adjust if the relevant flags are in an undefined state.

TRACE also allows the user to simulate interrupts and queues them in order of priority.

If execution time is critical to the application, TRACE can display cumulative machine cycles at each step or on return to the monitor.

Instructions

TRACE includes all the commands available in DEBUG as well as some special commands to access summary information developed by a trace of a user program. A complete summary of TRACE commands follows.

C - Continue with TRACE

The format of CONTINUE is

```
C  (CR)
C number-of-lines (CR)
C# (CR)
```

The first format traces the program through one instruction. The second format traces the user program through "number-of-lines" instructions, displaying the registers in accordance with the options entered following the "T" command described below. "number of lines" is entered as a hexadecimal number. You may trace through RAM or ROM.

The third format "C#" will cause the user program to be simulated continously, or until encountering a STOP ADDRESS, a HALT command, RST 30H or severe errors in the user program.

To abort a trace during execution hit any key on the console. Displays from the simulated instruction in progress will be completed,

the user registers will be preserved, and control returns to the monitor.

CN - Continue with no printing

The "CN" command is the same as the "C" command except that register display is surpressed.

Q - display circular queue

The circular queue showing the last 100 branches or the last 100 instructions simulated is displayed.

TT - Total time

The total number of simulated machine cycles from the last 'T' command is displayed as a decimal number. Time spent in user CDOS calls or real time segments of the user program is not included. This cycle count assumes memory with no wait states.

TU (CR) - Upper contents of User stack to CALL Stack

This command sets the upper area of the TRACE "CALL" stack equal to the 10 bytes above the user's stack pointer. If tracing starts inside up to 5 nested subroutines, TRACE will report no return errors if the return addresses were on the user's stack.

INITIALIZING SELECTED PARAMETERS

TRACE allows the user to initialize selected parameters at any time.

The following commands:

TD	Initialize DATA area list
TH	Initialize HALT addresses
TI	Initialize simulated INTERRUPT list
TN	Initialize real time INPUT channel list
TO	Initialize real time OUTPUT channel list
TP	Initialize OPTIONS
TR	Initialize trace RESTART addresses
TS	Initialize trace STOP addresses

are treated in detail in the sample trace below.

T - Initialize a TRACE

To initialize the parameters controlling the tracing of a user program type:

T (CR)

The following shows the use of TRACE to examine the action of the program SAMPLE listed in chapter 5. The user response is shown in upper case, the TRACE prompts are given in lower case. Entering an 'X' at any time following a TRACE prompt will return control to the monitor. The user may enter the parameters requested by TRACE as address expressions. Within these expressions addition,subtraction, the "@" register, and "$" (the current instruction counter) may be used.

start addr? 100(CR)

Simulation starts at the address entered.

TH - (HALT)

stop addrs (5)? $+3B (CR)
E000 (CR)
L (CR)

A list of up to 5 addresses (maximum) at which tracing stops and control returns to the monitor. These addresses must correspond to the first byte of an instruction in order to take effect. 'L' terminates a list.

TD - (Data areas)

data areas (10) low addr,high addr
184,1AC;L

Enter a maximum of 10 memory areas to which the user program writes. If the user program writes to an address within these areas, no error is reported.

Commas or spaces may be used as delimiters, carriage returns or semicolons separate items entered as sets (e.g.,pairs of addresses). If the stack area is not declared here, the message: @AAAA AAAA' MMMM MEMORY WRITE NOT IN SPECIFIED RECION will be printed after every stack writing operation.

After a CDOS call, the 8

locations of the user stack used by CDOS will be printed.

TW - (Display on write)

```
display on write (10) low addr,high addr
198 198+10
L
```

Enter a maximum of 10 areas within which a display of the contents of the memory is desired whenever the user program writes to these areas.

```
extended opt (y or n)?  Y
```

Type Y to enter the additional options listed below.

TN - (real time input)

```
real time:  inp ch(10)?
FF;L
```

List up to 10 input channel port addresses for which real time execution is desired.

TO - (real time output)

```
          output ch (10)?
FF;L
```

Designate up to 10 real time output channels. In this example inputs to the sense switches (INPUT 0FFH) and outputs to the sense lights (OUTPUT 0FFH) will be executed instead of simulated.

TS - (TRACE stop)

```
trace stop (5)?
012F;L
```

Enter up to 5 addresses at which control passes from TRACE to the user program. These must be on instruction boundaries. This and the following TRACE restart option allow the user to avoid tracing through subroutine calls, timing loops, etc.

TR - (TRACE restart)

```
trace restart (5)?
132;L
```

Enter up to 5 addresses at which control is to return to TRACE from the user program. These must be on instruction boundaries and this portion of the user program must be writeable memory.

TI - (Interrupts)

```
interrupts(10),addr,comm,pass,times?
11E,7E,1,13 (CR)
L   (CR)
```

Schedule up to 10 simulated interrupts.Enter the address (11E) at which the program will be interrupted, the byte appearing on the data bus (7E),the rate at which the address will be interrupted (1 for each time we pass this address, 2 for every other time etc.), and the number of times the program will be so interrupted by this pattern (13 hex or 19 times).

```
interrupt mode ?  0
```

Initialize interrupt mode at 0,1,or 2. If other values are entered the interrupt mode is set at 0, and a warning is printed if a simulated interrupt is executed before the interrupt mode has been defined.

TP - (Set options)

```
options  (2)?   9,2
```

The first entry (OPT 1) controls what is printed according to the following table:

```
ENTRY   ITEMS PRINTED

1       (PC) Instruction address
2 - - -  "   (F-A) Flags and Accumulator
3 - - -  "   - " -   B,C Register Pair
4 - - -  "   - " -  - " -   D,E Register pair
5 - - -  "   - " -  - " -  - " -   H,L Register Pair
6 - - -  "   - " -  - " -  - " -  - " -   IX Register
7 - - -  "   - " -  - " -  - " -  - " -  - " -  IY Register
8       All of the above plus the disassembled command.
9       All of the above plus the total machine cycles.
```

If a number for OPT 1 greater than 9 is entered, OPT 1 is set equal to 9.

The second entry (OPT 2) determines when register display occurs according to the following rules:

OPT 2 =1 — Print on executing a branch instruction, (i.e. CALL,RET,JP,JR,RST). Print current address (PC) and the next instruction address, and continue as directed by OPT 1.

OPT 2 =2 — Print at every instruction as directed by OPT 1.

OPT 2 =3 — At every instruction write the Program counter, Flags,accumulator, and the additional register pair selected by OPT 1 to the circular queue as follows:

OPT 1 <=3	BC register pair
OPT 1 = 4	DE "
OPT 1 = 5	HL "
OPT 1 = 6	IX register
OPT 1 = 7	IY "
OPT 1 = 8	SP (stack pointer)
OPT 1 = 9	(SP) contents of top of stack

This data is written to the circular queue unless OPT 2=4.

OPT 2=4 — At each branch instruction, write the Program counter, next instruction address, flags and accumulator to the circular queue.

When a Q command is given in response to the prompt "-" the contents of the circular queue are displayed, last instruction first.

If a value greater than 4 is entered, OPT 2 is set equal to 4.

Control now passes to the monitor to allow the user to set registers, examine memory, or execute the user program until TRACE restart breakpoints cause control to pass to the TRACE program, or monitor breakpoints return control to the monitor.

Typing "C#" causes TRACE to simulate the user program starting at the address given in response to the "start addr?" prompt.

Data Entry Messages

```
stop addrs (5)? 13B (CR)
135;152Z7
       i
error retype line
135;1527
```

In the example above the character "Z" was erroneously entered. TRACE types an I under the character in error, prompts "error retype line" and awaits the revised line. If a data entry containing such an error was made at the same line as a TRACE prompt, this error indicator will be offset to the right by the length of the prompt.

```
data areas (10) low addr,high addr
134,1AC
456,459;123;678,689
too few,last= 0123
```

Too few entries have been made for items required in sets. i.e. data areas, simulated interrupts. "last= 0123" shows the value of the last item properly read. In the example above, 0123 was entered without a corresponding upper boundary. The data area limits 678,689 must also be reentered as they were not read. Should it not be clear to the user which data items must be reentered, return to monitor by entering 'X' and enter the entire list under that category. Otherwise, enter the missing item and continue.

Execution Time Messages

DDAM (Data DAMage)

Parameters stored by TRACE have been changed by the user program while the user program had control. TRACE may be reinitialized without reloading.

@AAAA AAAA' INT M=XX I=YY INT=CC LEV=LL

Simulated interrupt at address AAAA. Interrupt mode = YY, I Register= YY the command on the interrupt line = CC, and the priority level LL which points to the simulated interrupt list.

@AAAA AAAA' ****** INTERRUPT MODE UNDEFINED, ASSUME =0

A simulated interrupt occurs at address AAAA with the mode not previously defined. The mode is assumed to be equal to 0.

@AAAA AAAA' ****** DAA WITH H FLAG INDEFINITE

AT location AAAA a decimal adjust command (DAA) was executed with the half-carry (H) flag undefined. Trace continues.

@AAAA AAAA' ****** MMMM MEMORY WRITE NOT IN SPECIFIED REGION

The commmand at address AAAA wrote to memory location MMMM which is neither in a declared data area nor in a " display on write " area.

@AAAA AAAA' ****** XXXX to YYYY BLOCK MOVE WILL OVERWRITE ITSELF

The command at address AAAA is a block transfer or a block input instruction which would write over itself. Tracing halts and control returns to the monitor.

PDAM (Program DAMage)

The user program has changed memory occupied by TRACE instructions. Control returns to the monitor. Reload TRACE before proceeding.

@AAAA AAAA' ****** CALL STACK OVERFLOW

Following a CALL,RST, or a MODE 1 interrupt, the call stack has overflowed or underflowed, and the return address error following this warning may be in error.

@AAAA AAAA' ****** RETURN ERROR MMMM WAS EXPECTED.
ACTUAL RETURN ADDR= NNNN.

Following a return instruction, the return address from the user stack (NNNN) does not agree with the return address (MMMM) entered to both the user stack and the CALL stack at the last CALL,RST,or MODE 1 interrupt. The simulated program counter is set at address NNNN and simulation proceeds.

S/P NOT R/W

The user program stack pointer was not assigned to RAM area when a TRACE RESTART instruction returned control to TRACE, or the user program encountered a CDOS call.

If this error message follows a trace restart instruction the proper restart address was not transferred through the stack to TRACE. Simulation halts and control returns to the monitor.

If the error message follows a CDOS call, control passes to the monitor before CDOS is called, as CDOS cannot be used with the user stack in a read only mode.

@AAAA AAAA' ****** MMMM WRITE TO SYSTEM AREA

Command attempted to write to address MMMM, within the TRACE or system area.

```
@AAAA AAAA' ****** XXXX to YYYY BLOCK WRITE TO SYSTEM AREA
```

At address AAAA a block transfer or a block input instruction attempts to write to TRACE or the system area. Tracing halts and control returns to the monitor.

```
@AAAA AAAA' ****** XXXX to YYYY BLOCK MOVE WRAPS AROUND MEMORY
```

At address AAAA a block transfer or block input command destination block wraps around upper to lower memory (addresses XXXX to YYYY). Tracing halts and control returns to the monitor.

```
@AAAA AAAA' ****** TRANSFER ON INDEFINITE FLAG
```

The flag controlling the transfer at instruction AAAA has been set by a command that leaves it in an indefinite state.

```
@AAAA AAAA' ****** UNDEFINED INSTRUCTION
CCCC
```

The instruction at AAAA is a two byte unidentified command (CCCC). Tracing halts and control passes to the monitor.

```
AAAA AAAA' ****** STOP ADDR
```

Tracing has stopped at address AAAA in the stop address list, or a monitor restart has been encountered. To continue tracing type 'C' and the instruction at the stop address will be simulated and tracing will continue.

```
================================
SIMULATED INPUT/OUTPUT MESSAGES
================================
```

```
@AAAA AAAA' IN #YY
```

This message indicates a simulated input is expected from the console. AAAA is the instruction address and YY is the input channel designated in hexadecimal. The user then enters the desired input data as a two digit hexadecimal number or as a single quote followed by an ASCII character.

Examples:

```
@2000 IN #3 7A

@2300 IN #2 'Q
```

```
============================
SIMULATED BLOCK INPUT/OUTPUT
============================
```

```
@AAAA AAAA' IN #YY SSSS TO TTTT
TYPE M FOR MONITOR
```

A block input instruction at AAAA will store input data in the

region bounded by SSSS and TTTT. The user may enter the monitor by typing M on the console and then substituting data directly into the memory area SSSS - TTTT, or fill this area with the Move command. As before YY is the input channel designation.

If the user does not invoke the monitor, TRACE expects up to 24 bytes of data per line and prompts with the destination address of the start of each line, and the total number of bytes remaining to be entered. The user may either enter the data as hexadecimal or ASCII preceded by a single quote.

```
@AAAA AAAA' OUT #YY XX A.
```

This indicates a simulated output on the console, where AAAA is the instruction address, YY is the output channel designation, XX is the output data in hexadecimal, and A is the ASCII equivalent. A period follows the ASCII output to aid in detecting spaces. Line feed is shown by 'LF', carriage return by 'CR'.

```
@AAAA AAAA' OUT #YY
BBBB: N1 R. N2 S. N3 T.
```

This indicates a simulated block output at address AAAA, directed to port YY. The starting source address for each line is BBBB, followed by a colon. N1,N2,N3 are the hex outputs, and R,S,T the corresponding ASCII outputs if these are printing characters, else blanks are output in their place. Carriage return and line feed are displayed as 'CR' and 'LF' respectively.

SIMULATED INTERRUPT OPTION

The user may enter in response to the prompt:

```
interrupts (10) addr,comm,pass,times?
```

up to 10 simulated interrupt patterns of the form

```
AAAA,CC,PPPP,NNNN
```

AAAA is the address of the command to be interrupted if interrupt enable flag is set.

CC is the command appearing on the interrupt line.

PPPP is the number of times (in hexadecimal) we pass this address to execute the interrupt. Thus if PPPP=1 we execute the interrupt on every time the simulated program counter reaches AAAA.

NNNN is the number of times this interrupt pattern is repeated. After NNNN interrupts at this address, no further interrupts from this line occur.

If the interrupts are disabled at the time an interrupt becomes pending, the interrupt is executed at the second command following an interrupt enable. If two or more interrupts become pending, the pattern entered first under the initial simulated interrputs prompt is the interrupt of highest priority.

If the interrupt mode changed while the user program was in control a simulated interrupt will not execute in the mode set under user control.

CDOS Calls

On encountering a transfer to the CDOS entry point (ordinarily CALL 5) TRACE verifies that the contents of the C-register request valid CDOS functions, and that any writing to memory is done to locations outside the TRACE or CDOS area, and that the user stack is in RAM. If the CDOS call writes to a disk, TRACE checks that the disk is logged on, displays a write to disk warning message on the console, and waits for the user to type 'Y' as confirmation before performing a disk write.

TRACE then executes the CDOS command and displays memory changes within display on write regions declared previously, and shows the results of any writing to memory outside declared areas. The actual execution will not be displayed, however.

CHAINING PROGRAMS UNDER CDOS CONTROL

If TRACE encounters a CDOS chain command the following message is displayed on the console:

```
CHAIN, TYPE D TO LOAD AND RETURN TO DEBUG, C TO CONTINUE.
```

Responding with a 'D' loads the new program originating at 100H and returns to the monitor.Typing 'C' resumes tracing after the new program is loaded, and entering any other character returns control to the monitor.

After chaining a new program the register contents are not predictable and those shown by TRACE will not be the same as a CHAIN command performed under control of the user program.

When the new program is sucessfully loaded, TRACE sends a message

```
CHAIN COMPLETE
```

to the console.

CDOS CALL ERROR MESSAGES

CHAIN ERROR

The chained program is not on the disk, or did not read properly from the disk. Control passes to the monitor.

CHAIN FILE NOT A COM FILE

The extent of the program called by CDOS is not 'COM'. Control passes to the monitor.

DISK NOT LOGGED ON

An attempt was made to access a disk not logged on. Control passes to the monitor.

INVALID CDOS CALL

The contents of the C register do not indicate a valid CDOS routine. Control passes to the monitor.

CDOS CRASH MESSAGE

This is printed in lieu of the "INVALID JUMP TO LOCATION XXXX" message reported by CDOS (due to executing a RST 7 instruction caused by defective memory, perhaps). Control then passes to the monitor.

CDOS WARM BOOT ATTEMPTED

A program has attempted to reboot CDOS via a jump to location 0000. Control passes to the monitor.

Chapter 2

Trace Command Format

Trace Command Format

The CROMEMCO TRACE program makes it possible to test and analyze user programs. TRACE is loaded into memory and moved to the highest memory available below CDOS. When using a 32K CDOS and TRACE, there is 10K left for the user program.

Loading Trace

TRACE is loaded by typing one of the following commands from CDOS.

```
TRACE
TRACE filename.ext
```

where "filename" is the name of the program to be tested, and "ext" is the file extension. In both cases, TRACE is loaded into memory directly below CDOS. The CDOS jump instruction located at location 5H is changed to jump to the start of TRACE. This allows locations 6H and 7H to still point to the lowest available memory location.

The second command above is used to load the file to be tested into memory. If the extension ("ext") is ".HEX", then the file is read as an INTEL HEX file. Any other extension is read as an absolute binary file, loaded at location 100H. **** NOTE **** TRACE does not load relocatable files. If an extension is ".REL" it will be loaded in as if it were binary and will not be executable.

Control Characters

Control characters are used in TRACE to help in entering commands. These control characters are the same as CDOS uses.

Control-C (^C)	go back to CDOS
Control-H (^H)	delete character and backspace on CRT
Control-U (^U)	delete line
Control-X (^X)	delete character and echo
underscore	delete character and backspace on CRT
RUBout (DEL)	delete character and backspace on CRT

During a printing (such as from the DM command) the following characters may be used.

Control-S (^S)	stop/start printing. If printing, this character will stop the printing. If already stopped, this character will resume the printing.
break	(or any other character) will abort the printing, prompt, and wait for the next command.

Command Format

TRACE is controlled by one and two character commands from the termwithal. The format is free-form in respect to spaces. Commas may be used in place of spaces. In the following, the examples all dump memory starting at location 1000H and ending at location 10FFH.

```
DM1000 10FF (CR)
DM1000S100 (CR)
D  M  1000    10FF (CR)
D  M  1000  S 100  (CR)
DM1000,10FF (CR)
DM1000,S100 (CR)
D  M  1000 ,  10FF (CR)
```

@ Register

TRACE was designed to give flexibility in testing relocatable programs. The "@" register is used to tell TRACE where the module you wish to debug is located. This address can be found from the map generated by the linking loader "LINK". To change the "@" register, type "@ (CR)" on the console. The computer will then type "@-xxxx " (where xxxx is the current value of the register. The computer will then wait for a new address. If a CR only is typed, the register remains unchanged. If an address and a CR is typed, then the register will contain the new address. The "@" register may now be used as part of an address. The following example demonstrates it's use.

```
G/@ @A3 1000
```

This is an exaple of the go command. Break points will be set at the beginning of the current module, relative location A3H in the current module, and at location 1000H. This feature allows you to test a module without having to calculate absolute addresses.

Address Expressions

For additional ease in specifying addresses an expression can be used. Within these expressions, addition, subtraction, the "@" register, and the "$" may be used. The "$" is the current location of the program counter (P register). If many modules are being tested, addition can be used to specify relative addresses.

```
G/2321+A3
```

The preceeding example would set a break point at relative location A3H if the module is located at 2321H.

Swath Operator

There are two ways to specify the address range of many commands. The first is to simply list the beginning and end addresses (and where appropriate, the destination address). For example, the first command below programs the range 0 through 13FFH into PROMs starting at location E400H. The second command displays the contents of memory between addresses E400H and E402H.

```
P0 13FF E400
DME400  E402
```

Another way to do the same thing is to use the Swath operator, "S", to specify the width of the address range, rather than state the end address explicitly.

```
P0 S1400 E400
DM E400S3
```

Errors

Any errors made during entering of a command may be corrected by typing Control-U (^U) to abort the line or by backspacing and correcting the line. If a CR has already been entered and TRACE detects an error, the line will not be accepted and a "?" will be printed. Re-enter the line with the incorrect data corrected.

Chapter 3

Other Trace Commands

Other Trace Commands

The remaining TRACE commands are described in detail below. The operator must wait for prompt character ("-") before entering the command. These commands are identical to those in DEBUG, the Cromemco Debugging Program.

A - Assemble into memory

This command allows in-line assembly language to be assembled into memory. The command takes the following format.

```
A beginning-addr (CR)
```

The user is prompted with the absolute address, followed by the relative address. TRACE reads from the console the assembler mnemonics. and assembles the instruction into memory. The mnemonics for the various Z-80 instructions can be found in the Z-80 CPU TECHNICAL MANUAL published by Mostek and Zilog. If there was no error in the instruction, it is stored in memory and the user is prompted for the next instruction. The The rules for address expressions apply to the addresses in the assembler mnemonics. In the following example the "@" register contains 1234H.

```
A@40
1274 0040'  ADD B
1275 0041'  CALL @93
1278 0044'  JP 1032+95
127B 0047'  .
```

The A command terminates when the first blank line or a line starting with a "." is entered from the console. If there is an error in the input line, it will not be accepted, a "?" will be printed and the console will be prompted with the addresses again.

DM - DISPLAY MEMORY

The contents of memory are displayed in hexadecimal form. each line of the display is preceded by the address of the first byte and followed by the ASCII representation of the hexadecimal bytes. An example follows

```
DM100,S30
0100  40 41 42 43 44 45 46 47-48 49 4A 4B 4C 4D 4E 4F  @ABCDEFGHIJKLMNO
0110  50 51 52 53 54 55 56 57-58 59 5A 30 31 32 33 34  PQRSTUVWXYZ01234
0120  35 36 37 38 39 00 00 00-00 00 00 00 00 00 00 00  56789...........
```

The formats of this command are as follows.

```
DM (CR)
DM beginning-addr (CR)
DM beginning-addr ending-addr (CR)
DM beginning-addr S swath-width (CR)
DM,ending-addr (CR)
DM S swath-width (CR)
```

The first format displays memory from the CURRENT display address, initially 100H, and continues for 8 lines. The second format displays from the beginning address and continues for 8 lines. The third format displays from the beginning address to the ending address. The forth format displays from the beginning address for a length specified by the swath-width. The fifth format displays from the CURRENT display address to the ending address. The sixth format displays from the CURRENT display address for a length specified by the swath-width.

If an "X" is included after the "DM", the relative addresses are also printed. In the following example assume that the "@" register contains 100H.

```
DMX100,S30
0100 0000' 40 41 42 43 44 45 46 47-48 49 4A 4B 4C 4D 4E 4F  @ABCDEFGHIJKLMNO
0110 0010' 50 51 52 53 54 55 56 57-58 59 5A 30 31 32 33 34  PQRSTUVWXYZ01234
0120 0020' 35 36 37 38 39 00 00 00-00 00 00 00 00 00 00 00  56789...........
```

DR - DISPLAY REGISTERS

When TRACE is re-entered from a break point, the user registers are saved. The registers may be displayed at any time by typing the following command.

```
-DR (CR)
SZHVNCE A=00 BC=0000 DE=0000 HL=0000 S=0100 P=0100 0100' LD   E,A
SZHVNC A'=00 B'=0000 D'=0000 H'=0000 X=0000 Y=0000 I=00
```

The letters "SZHVNC" are the flags, on the 2nd row are the prime flags. If the flag is on, it is printed, if the flag is off, a space is printed. If only the carry and zero flag are set then " Z C" would be printed. The flags are described below.

- S - Sign flag, S=1 if the MSB of the result is one, ie, the result is negative.
- Z - Zero flag, Z=1 if the result of an operation is zero.
- H - Half-carry flag, H=1 if the add operation produced a carry into the 4th bit of the accumulator or a subtract operation produced a borrow from the 4th bit of the accumulator.
- V - Parity or overflow flag. This flag is affected by arithmetic and logical operations. If an overflow occurs durring an arithmetic operation, the flag is set to one. After a logical operation, the flag is set to 1 if the result of the operation has even parity.

N - Add/subtract flag, N=1 if the last operation was a subtraction.
C - Carry flag, C=1 if the operation produced a carry.

The E flag on the first line is the state of the interrupt enabled flip-flop (IFF). If interrupts are enabled, the "E" is printed, otherwise a space is printed.

The A register is printed next, followed by the BC, DE, and HL register pairs and the stack pointer. The program counter value is then printed in both absolute and relative. The opcode pointed to by the program counter is then displayed as an instruction.

On the second line, the prime registers are displayed, F' (prime flags), A', BC', DE', and HL'. The IX, IY, and I (interrupt page) registers are printed next. If the disassembled opcode includes an address, the relative value of this address is printed as the last thing on the line.

```
-DR (CR)
S H NCE A=00 BC=0000 DE=0000 HL=0000 S=0000 P=1234 0010' CALL 1334
SZ  NC A'=00 B'=0000 D'=0000 H'=0000 X=0000 Y=0000 I=00       (0110')
```

E - EXAMINE INPUT PORT

The data port is read and displayed as a hexadecimal number. The format of the command is

E data-port (CR)

In the following example the data port 3 is read and displayed on the console.

```
-E3 (CR)
23
```

EJ - EJECT DISK

The format of the command follows.

EJ d

The d is the disk number (A, B, C, D). If the designated disk is a CROMEMCO DUAL DISK SYSTEM model PFD, with the eject option, the diskette in the disk drive will eject.

F - SPECIFY FILE NAME

This command allows the operator to insert filenames in the two default FCBs (at 5CH and 6CH) and the command line into the default buffer (at 80H). The example below loads FILE1.COM into the first FCB and FILE2.COM into the second FCB. The complete line is also loaded into the default buffer.

```
-FFILE1.COM FILE2.COM OPTION1 OPTION2
```

This command can be used with the "R" command to read in disk files.

G - GO

The GO command has the following format.

```
G(starting-addr)/(breakpoint-1) (breakpoint-2)...(breakpoint-5)
```

Each of the addresses are optional. If the starting address is omitted, the the contents of the program counter is used. The registers are loaded from the user registers (these are the values displayed with the DR command). Execution begins with the starting address or the contents of the program counter. If break points were specified, a RST 30H is inserted at the break point addresses and a jump instruction is placed at location 30H. When a breakpoint is executed, control is returned to TRACE, and all of the user registers are saved (the registers may then be displayed with the DR command). ALL breakpoints are then removed from the user program. The program counter is displayed after the breakpoint. Note the following about breakpoints:

(a) Breakpoints can only be set in programs residing in RAM. This is because a RST 30H is inserted at each break point location. (The original contents of these locations are saved so that they can be restored after a break point is executed.)

(b) Up to 5 break points can be set. If an attempt is made to enter more than 5 break points, the command will not be accepted.

(c) When a break point is used, a jump instruction is stored at location 30H. Therefore locations 30H, 31H, and 32H are not available to a user program.

The GO command has an additional feature that is very helpful in debugging a program. A count is allowed for each break-point. This count is entered after the break-point and enclosed in parentheses. This count is the number of times the program reaches this address before control is returned to TRACE. A count of one says to break the next time the address is reached. In the example below execution begins at location 100H and will break when address 109H is reached for the second time or when 123H is reached for the first time.

```
-G100/109(2) 123
```

Note that 123 and 123(1) means the same thing. Also note that the

count is a hexadecimal number. Therefore 123(F) means to break after the address has been executed for the 15th time.

H - HEXADECIMAL ARITHMETIC

Hexadecimal addition and subtraction may be performed by this command. The first number to be printed is the sum of the two input numbers. The second number to be printed is the difference between the first number and the second number. In the example following, the first number is 1234 + 321, and the second number is 1234 - 321.

```
-H1234,321
1555 0F13
```

L - LIST IN ASSEMBLER MNEUMONICS

The list command is used to list the contents of memory in assembly language mnemonics. The formats for this command are.

```
L (CR)
L starting-addr (CR)
L starting-addr ending-addr (CR)
L starting-addf S swath-width (CR)
L,ending-addr (CR)
L S swath-width (CR)
```

The first format lists 16 lines of disassembled code starting from the current list address. The second format lists 16 lines from the starting address. The third format lists from the starting address to the ending address. The fourth format lists from the starting address for a length specified by the swath width. The fifth format lists from the current list address to the ending address. The sixth format lists from the current address for a length specified by the swath address.

The first address of the disassembly is the absolute address. The second address is the relative address. If the disassembled instruction contains an address, the absolute address is printed in the instruction in hexadecimal and the relative address is printed to the right of the disassembled line. In the example that follows, the "@" register contains 2800H.

```
-L@800 812
3000 0800'  ADD  B
3001 0801'  CALL 3200          (0A00')
3004 0804'  CALL 3243          (0A43')
3007 0807'  CALL 3333          (0B33')
300A 080A'  LD   A,B
300B 080B'  OR   C
300C 080C'  JR   Z,3000        (0800')
300F 080F'  INC  HL
3010 0810'  INC  DE
3011 0811'  INC  BC
3012 0812'  LD   A,H
```

M - MOVE MEMORY

The formats of this command follow.

```
M source-addr source-end destination-addr
M source-addr S swath-width destination-addr
```

The first format moves the contents of memory beginning with the source address and ending with the source-end to the destination address. The second format uses the swath width to determine the length of the move.

The move is verified to insure that all bytes were moved correctly. If an overlapping move was made, errors will be reported. The error reporting can be terminated by typing any character.

The move command can be used to fill a block of memory with a constant. In the following example, a zero has been entered into location 100H using the SM command. The following command will move zeros from location 100H through 108H.

```
-M100 S7 101
```

Care should be taken not to move memory over TRACE or CDOS.

O - OUTPUT TO DATA PORT

This command outputs data to a data port. The following is the command format.

```
O data-byte port-number (CR)
```

P - PROGRAM PROMS

This command allows programming of PROMs. The following are the command formats.

```
P source-addr source-end destination-addr
P source-addr S swath-width destination-addr
```

The first format programs PROMs starting with the source address and ending with the source-end into PROMs beginning at the destination address. The second format determines the length from the swath width.

If the length of the source is not a multiple of 400H or if the destination does not begin at a 400H boundry, TRACE will reject the command. (Multiples of 400H end in '000', '400', '800', and 'C00'.)

Any number of 2708 or 2704 PROMs can be programmed in the execution of one command as long as there are enough BYTESAVERS to contain them. Each PROM is verified with its source after all are programmed and any discrepancies are printed out. If no discrepancies are found, a prompt is printed and the next command may be entered.

Software can be loaded into a PROM in as small increments as you desire, provided it is added to previously unused areas of the PROM. This is done by first using the Move command, "M", to transfer the contents of the PROM to RAM, adding the new software to an area of RAM which corresponds to the unused portion of the PROM and finally using the Program command, "P", to reprogram the PROM with the result. Athough the entire PROM must always be programmed, it never hurts to rewrite the same data over again. In general, a 1 may be written over a 1, a Ø over either a 1 or a Ø, but the only way to change Ø's to 1's is to erase the PROM with appropriate UV light. (See the BYTESAVER manual for details.)

R - READ DISK FILE

This command allows the operator to read a disk file. The "R" command is used with the "F" command. The "F" command is used to specify the filename, and the "R" command reads in the file. If the file has an extension of ".HEX", then the file is an INTEL HEX file and will be read into memory. Any other file is considered to be a binary file and will be read directly into memory beginning at location 1ØØH. The format of the "R" commmands is

```
R
R displacement
```

The first format reads the file with no displacement. The second format reads the file with a displacement. If the input file is in HEX, then the displacement is added to the addresses in the file to determine the addresses at which to store the file. If the file is a binary file, it will be stored at the displacement + 1ØØH.

When the "R" command is executed, TRACE prints either a "?" if there is an error (file not found, checksum error, or file attempting to read above highest available memory location) or with the following message if there is no error.

```
NEXT = xxxx
```

Where xxxx is the address of the next available memory location past the end of the file.

SM - SUBSTITUTE MEMORY

This command is used to substitute memory. The format of the command follows.

```
SM starting-addr
```

TRACE prints the absolute address, followed by the relative address, followed by the contents of the memory byte. One of the following may then be entered.

(a) data-byte value. The data byte value is stored at the address of the prompt. The address is then incremented by 1 and displayed on the next line.
(b) string enclosed in quotes. The string is stored beginning at the address of the prompt. The address is then incremented past the string and displayed on the next line.
(c) Any number of (a) and (b) above can be entered on one line. The address is then incremented past the bytes that were stored and the new address is displayed on the next line.
(d) "-". A minus sign does not store a byte. The address will be decremented to the previous address. The minus sign can be used to "back up" to a previous location in case an error has been made.
(e) (CR) only. If no entry is made on the line, the memory byte remains unchanged. The address is incremented by 1 and displayed on the next line.
(f) period. A period ends the input mode and returns to the command level.

In the example that follows, assume that the "@" register contains the value 2800H.

```
-SM@100
2900 0100' 32 0
2901 0101' 17 00
2902 0102' 31 'THIS IS AN ASCII STRING'
2919 0119' 7A 'AAAA' 0 0 1 2 3 4 5 6 7 8 9
2928 0128' 22
2929 0129' 29
292A 012A' 87 -
2929 0129' 29 .
```

Sr - SUBSTITUTE REGISTER

The Sr command allows the user registers to be altered. The letter "r" stands for the register which is to be changed. The section SUMMARY OF REGISTER NAMES gives a summary of the names that can be substituted. When substituting the F and F' flags, enter the command SF or SF'. TRACE will then print the flags that are set and wait for the operator to enter the names of the registers that are to be set. If the flags are NOT entered, the flags are reset. In the following example, the "SZHC" flags are set. After the example is executed the "ZC" flags are set. The lower case letters are entered by the operator.

```
-sf
SZH  C  zc
```

When substituting a one byte register, a one byte value is accepted. When substituting a two byte register, a two byte value is accepted. If no value is entered, or if an error occurs, the value of the register

remains unchanged. In the following example, the A register is changed to contain 41H.

```
-sa
A=98 41
```

T - TRACE

The format of trace is

```
T (CR)
```

When this command is executed, TRACE will prompt the user for information regarding the program to be simulated. At this time, start and stop addresses, data areas, interrupts, I/O and other parameters will be set up for the next TRACE to be initiated. Refer to Chapter 3 for a complete explanation of the TRACE simulation commands.

V - VERIFY MEMORY

Verify that the block of memory between source address and source end contain the same values as the block beginning at destination address. The addresses and contents are printed for each discrepancy found. The following is the format of this command.

```
V source-addr source-end destination-addr
V source-addr S swath-width destination-addr
```

This command works by reading bytes from the source and destination and comparing them. If a discrepancy is found, the memory is read again for print-out. Thus, it can happen that a discrepancy is printed-out with the source and the destination contents indicated to be the same. This is caused by a defective memory element.

A discrepancy is printed in the following order: source address, source contents, destination contents, destination address. In the example that follows, memory locations 1003H and 1008H are defective.

```
-V 0 S30 1000
0003 32 12 1003
0008 7A 5A 1008
-
```

Chapter 4

Summary Of Trace Commands

Summary Of Trace Commands

The following is an alphabetical list of the TRACE commands.

Command	Description
A	Assemble into memory
C	Continue with Trace
CN	Continue with No printing
DM	Display Memory
DR	Display Register
E	Examine input port
EJ	EJect disk
F	specify disk File name
G	Go
H	Hexadecimal arithmetic
L	List in assembler mnemonics
M	Move memory
O	Output to data port
P	Program PROMs
Q	display circular Queue
R	Read disk file
SM	Substitute Memory
Sr	Substitute register

COMMAND	DESCRIPTION
T	set up Trace parameters
TD	initialize Trace Data areas
TH	initialize Trace Halt addresses
TI	initialize Trace Interrupt list
TN	initialize Trace iNput channel list
TO	initialize Trace Output channel list
TP	initialize Trace oPtions
TR	initialize Trace Restart addresses
TS	initialize Trace Stop addresses
TT	display Total elapsed Time in machine cycles
TU	put Trace stack above User stack
TW	initialize Trace display on Write areas
V	Verify memory

Summary Of Register Names

The following register names are printed by the DM command and should be used with the Sr command.

Register	Decription
F	Flags, the following flags may be changed. S - Sign flag Z - Zero flag H - Half carry flag V - parity/oVerflow flag N - subtractioN flag C - Carry flag The interrupt enable flag ("E") may also be changed.
F'	The F' flags are the same as the "F" flags. (note that the "E" flag may not be changed here.)
A	accumulator
A'	prime accumulator
B	BC register pair
B'	BC' register pair
D	DE register pair
D'	DE' register pair
H	HL register pair
H'	HL' register pair
S	Stack pointer
P	Program counter
X	IX register
Y	IY register
I	Interrupt page register

Chapter 5

Trace Sample Run

```
TYPE SAMPLE.PRN
CROMEMCO CDOS Z80 ASSEMBLER V.1.4A

                0001 ;TEST PROGRAM FOR TRACE DOCUMENTATION.
                0002 ;
0000            0003          ORG     100H
     (E000)     0004 MONIT:   EQU     0E000H  ;MONITOR ADDRESS
                0005 ;
                0006 ;INITIALIZE REGISTERS AND MACHINE STATE
                0007 ;
0100 F3         0008 START:   DI
0101 319801     0009          LD      SP,STACK+20
0104 011601     0010          LD      BC,FLSRT        ;FALSE RETURN ADDRESS
0107 219801     0011          LD      HL,SOURCE
010A 11A201     0012          LD      DE,DEST
010D 3E02       0013          LD      A,2
010F 32AC01     0014          LD      (GOODW),A       ;WRITE TO A DATA AREA
0112 CD8001     0015          CALL    SUBR            ;SUBROUTINE WITH
                0016 ;                                ;AN UNPAIRED PUSH
0115 00         0017          NOP
0116 3E01       0018 FLSRT:   LD      A,1H    ;SUBR RETURNS HERE
0118 ED47       0019          LD      I,A             ;INITIALIZE INTERRUPT VECTOR
011A ED5E       0020          IM2                     ;SET INTERRUPT MODE
011C FB         0021          EI                      ;ENABLE INTERRUPT
011D 12         0022          LD      (DE),A
011E 011000     0023          LD      BC,16           ;INITIALIZE FOR LDIR
                0024 ;**********  THE SIMULATED INTERRUPT OCCURS HERE********
                0025 ;
0121 EDB0       0026          LDIR                    ;BLOCK TRANSFER
                0027 ;                                ;END OF BLOCK WRITES TO
                0028 ;                                ;AN UNDECLARED REGION OF MEMORY
0123 01FF10     0029          LD      BC,10FFH        ;INITIALIZE FOR BLOCK INPUT
                0030 ;                                ;FROM CHANNEL 0FFH
0126 EDBA       0031          INDR
0128 060A       0032          LD      B,10
012A EDB3       0033          OTIR                    ;BLOCK OUTPUT ON SIMULATED CHANNEL
012C FC8101     0034          CALL    M,SUBR1         ;SIGN FLAG NOT DEFINED AFTER OTIR COMMAND
                0035 ;
                0036 ;********** CONTROL PASSES TO THE USER PROGRAM AT HIS POINT**********
                0037 ;
012F 3C         0038 INCZER:  INC     A
0130 20FD       0039          JR      NZ,INCZER
                0040 ;**************************RESUME TRACE WHEN WE EXIT THIS LOOP***********
                0041 ;
0132 DD219801   0042          LD      IX,SOURCE
0136 DD0A       0043          DB      0DDH,0AH        ;A 2-BYTE UNDEFINED COMMAND
0138 DDB605     0044          OR      (IX+5)
013B C300E0     0045          JP      MONIT
013E            0046 ORG      17EH                    ;FIX VECTOR FOR INTERRUPT
017E 8101       0047          DW      SUBR1
                0048 ;
0180 C5         0049 SUBR:    PUSH    BC              ;STORE FALSE RETURN ADDRESS ON STACK
0181 C606       0050 SUBR1:   ADD     6               ;ENTRY FOR NORMAL SUBROUTINE
0183 C9         0051          RET
                0052 ;****************DATA AREA*************
                0053 ;
```

```
0184 (0014)          0054 STACK:  DEFS    20
0198 (000A)          0055 SOURCE: DEFS    10
01A2 (000A)          0056 DEST:   DEFS    10
01AC (0001)          0057 GOODW:  DEFS    1
                     0058 ;
01AD                 0059 END

0000 ERRORS


C.TRACE SAMPLE.HEX              The program "SAMPLE.HEX" is to be Traced.
TRACE VERSION 2.11
NEXT = 0184
-T
V2.10

START ADDR?100

STOP ADDR(5)?$+3B
E000
L
DATA AREAS(10)LOW ADDR,HIGH ADDR?
184,1AC;L
DISPLAY ON WRITE(10)LOW ADDR,HIGH ADDR?
198 198+10
L
EXTEND OPT(Y OR N)?Y
REAL TIME: INPUT CH (10)? FF;L
   OUTPUT CH (10)? FF;L
TRACE STOP(5)? 12F;L
TRACE RESTART(5)? 132;L
INTERRUPTS(10),ADDR,COMM,PASS,TIMES?
11E,7E,1,13
L
INTERRUPT MODE? 0

OPTIONS(2)?  9,2                Prints all machine state info after every instruction.

-C#                             Trace continuously from the start addr specified above.
INSTP        FLAGS   A  B C  D E  H L  IX   IY
 @0100               00 0000 0000 0000 0000 0000 DI
     4               00 0000 0000 0000      These are the "prime" registers.
 I=00 SP= 0100       (SP)= 31F3

 @0101               00 0000 0000 0000 0000 0000 LD SP,0198
    14 SP= 0198       (SP)= 0000            Stack Pointer and contents.

 @0104               00 0116 0000 0000 0000 0000 LD BC,0116
    24
 @0107               00 0116 0000 0198 0000 0000 LD HL,0198
    34
 @010A               00 0116 01A2 0198 0000 0000 LD DE,01A2
    44
 @010D               02 0116 01A2 0198 0000 0000 LD A,02
    51
 @010F               02 0116 01A2 0198 0000 0000 LD (01AC),A
```

```
   64
 @0112              02 0116 01A2 0198 0000 0000 CALL 0180
   74 SP= 0196       (SP)= 0115

 @0180              02 0116 01A2 0198 0000 0000 PUSH BC
   85 SP= 0194       (SP)= 0116

 @0181              08 0116 01A2 0198 0000 0000 ADD 06
   92
@0183       ********* RETURN ERROR          (due to the PUSH BC above)
0115       WAS EXPECTED, ACTUAL RETURN ADDR= 0116
 @0183              08 0116 01A2 0198 0000 0000 RET
  102 SP= 0196       (SP)= 0115

INSTP        FLAGS   A  B C  D E  H L  IX   IY
 @0116              01 0116 01A2 0198 0000 0000 LD A,01
  109
 @0118              01 0116 01A2 0198 0000 0000 LD I,A
  118 I=01                                    Interrupt register initialized.
 @011A              01 0116 01A2 0198 0000 0000 IM 2
  126
 @011C              01 0116 01A2 0198 0000 0000 EI
  130
 @011D            E 01 0116 01A2 0198 0000 0000 LD (DE),A
  137 01A2      =01
@011E       INT M=02 I=01 INT=7E  LEV=01    Simulated interrupt.
 @011E              01 0116 01A2 0198 0000 0000
  156 SP= 0194       (SP)= 011E

 @0181              07 0116 01A2 0198 0000 0000 ADD 06
  163
 @0183              07 0116 01A2 0198 0000 0000 RET
  173 SP= 0196       (SP)= 0115

 @011E              07 0010 01A2 0198 0000 0000 LD BC,0010
  183
@0121       01A2        TO       01B1       Block transfer to display on write area
:01A2       00 00 00 00 00 00 00   .......  overlaps into a non-specified area.
@0121       ********* MEMORY WRITE NOT IN A SPECIFIED REGION
M01AD       00 00 00 00 00   .....
 @0121              07 0000 01B2 01A8 0000 0000 LDIR
  529
 @0123              07 10FF 01B2 01A8 0000 0000 LD BC,10FF
  539
@0126       0199        TO       01A8
:0199       FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
INSTP        FLAGS   A  B C  D E  H L  IX   IY
 @0126         ZHVNC  07 00FF 01B2 0198 0000 0000 INDR          Real time input
  859                                                           (values above)
 @0128         ZHVNC  07 0AFF 01B2 0198 0000 0000 LD B,0A
  866
 @012A         ZH NC  07 00FF 01B2 01A2 0000 0000 OTIR
 1066
@012C       ********* TRANSFER ON INDEFINITE FLAG              Sign flag not defined
 @012C         ZH NC  07 00FF 01B2 01A2 0000 0000 CALL M ,0181
```

```
  1076
 @0132        ZH  C  00 00FF 01B2 01H2 0198 0000 LD IX,0198
  1090
@0136        ********** UNDEFINED INSTRUCTION
DD 0A

@0136        ********** STOP ADDR

P=0136 0136
```

The 2-byte instruction DD 0A is undefined for the Z-80
Control returns to the monitor.

```
-Q
```

Display contents of Queue buffer.

```
INSTP        FLAGS   A (SP)
+0132        ZH  C  00 0115
+012C        ZH NC  07 0115
+012A        ZH NC  07 0115
+0128        ZHVNC  07 0115
+0126        ZHVNC  07 0115
+0123               07 0115
+0121               07 0115
+011E               07 0115
+0183               07 0115
+0181               07 011E
+011E               01 011E
+011D             E 01 0115
+011C               01 0115
+011A               01 0115
+0118               01 0115
+0116               01 0115
+0183               08 0115
+0181               08 0116
+0180               02 0116
+0112               02 0115
+010F               02 0000
+010D               02 0000
+010A               00 0000
+0107               00 0000
+0104               00 0000
+0101               00 0000
+0100               00 31F3
-^C
C.
```