TRACE

Copyright (c) 1980

by
Roger C. Wickenden

Pegasus Software
P.O. Box 10014
Honolulu, HI 96816

TRACE

A 6502 Machine Language Program for OSI computers from PEGASUS SOFTWARE.
Copyright (c) 1980 by Roger C. Wickenden.

OVERALL DESCRIPTION

TRACE is a 6502 machine language program that operates in Ohio Scientific microcomputers to provide a powerful tool for debugging and analyzing object programs. The term "object program" refers to any 6502 machine language program that may reside in memory concurrently with TRACE.

TRACE follows the inherent instruction sequence of the object program and executes or emulates each instruction that it comes upon. Tracing may be stopped at any point in the object program to permit register and memory contents to be inspected and modified. Inspection feature enables the actual performance of any instruction sequence of the object program to be observed and compared with the intended performance, so that discrepancies may be pinpointed. The modification feature enables the programmer to make "on the spot" revisions to the program or to the data so that the effects of proposed corrections to the program may be observed.

Both single-step and continuous-trace modes are provided. In single-step mode, each command from the operator to resume the trace causes only the next instruction of the object program to be observed. In continuous mode, one intruction is traced after the other without stopping, so the object program is effectively run in slow motion. This continues until one of four events occurs:

(1) Depression of the "control A" key is recognized
(2) An illegal instruction is encountered
(3) A stack overflow condition is threatened
(4) A preset breakpoint is encountered.

It makes no difference whether the object code being traced is in Read Only Memory (ROM) or Read-and-Write (RAM) Memory.

TRACE may be operated in a mode where the contents of the 6502 microprocessor registers are visually displayed automatically for each instruction traced, along with the hexadecimal representation of the instruction, its address, and a mnemonic display of the instruction. Since each instruction so displayed requires only ome line of printing, preceding lines on the output device show the sequence of instrutions that led up to the last instruction traced.

INITIALIZING REGISTER CONTENTS

Before tracing the initial instruction of an object program it may be necessary to set the contents of one or more registers and breakpoints. The contents of A, P, S, X, and Y will be the values that existed upon entering TRACE, and often these will be acceptable for beginning the desired trace.

The "I" command is used to designate the starting address of the program to be traced, so it must be set to begin.

STARTING A TRACE

TRACE recognizes two trace modes which are selected by a single character as follows:

T Trace Continuously
W "Walk" Thru a Single Instruction

After typing either of these commands, depressing either "RETURN" or the space bar will put the command into effect.

STOPPING A TRACE

During a continuous trace one instruction is traced after another without stopping, so the object program is effectively run in slow motion. This continues until one of four events occurs:

(1) Depression of the "Control A" key is recognized
(2) An illegal instruction is encountered
(3) A stack overflow condition threatens, or
(4) A preset breakpoint is encountered.

If an attempt is made to trace an illegal instruction, tracing will halt and the word "BAD" will be displayed on the screen followed by the address and contents of the illegal instruction and a new prompt character will be displayed.

If TRACE finds the contents of the Stack Pointer to be in the range ØØ thru Ø9 after tracing any instruction then the normal operation of TRACE is sidetracked. This is because there is a danger that continued operation of TRACE would cause a stack overflow condition if such a condition has not already occurred. The character "S", the contents of the stack pointer, and a new prompt character are displayed on the screen. It is then necessary to reset "S" to a suitable value before starting another trace.

Once entry to TRACE has been accomplised, it will immediately display its prompt character at the left edge of the screen:

TRACE is awaiting keyboard input.

INSPECTING AND MODIFYING MEMORY CONTENTS

A memory location is inspected by typing its hexadecimal address, using no more than four characters, followed by a space. The contents of the addressed byte will immediately be displayed on the screen, in hexadecimal form, next to the address. If no change is to be made in the byte's contents, the operator may press the "RETURN" key. A new prompt character will appear on the next line. If inspection of the next successive memory location is desired, the operator depresses the "↑" key instead of the "RETURN" key. If it is desired to change the contents of the byte, the desired contents are typed next to the existing contents prior to depressing the "RETURN" or "↑" key. The revised contents are stored in the addressed location when the "RETURN" or "↑" key is depressed.

INSPECTING AND MODIFYING REGISTER CONTENTS

A single character is used to designate each register as follows:

@	Accumulator
I	Program Counter (Instruction Pointer)
P	Process Status Register
S	Stack Pointer
X	Index Register X
Y	Index Register Y

The breakpoint pointers may be inspected or modified just like the registers, so their symbols will be listed here also:

HØ	First Breakpoint Pointer (Halt Location)
H1	Second Breakpoint Pointer
.	
.	
.	
H8	Eighth Breakpoint Pointer

The single character "H" may also be used in place of HØ.

The register contents that will apply when the next instruction is traced may be inspected by typing the appropriate symbol in the above list followed by a space. If it is desired to change the contents of the register, the desired contents are typed next to the existing contents and the "RETURN" key is depressed. The "↑" key should not be used when inspecting register contents.

If a breakpoint is encountered during a continuous trace, the trace will stop and the breakpoint number (H0 thru H7) will be displayed followed by a new prompt character.

INSPECTING THE MOST-RECENTLY-TRACED INSTRUCTION LIST

Typing an "R" and then pressing "RETURN" causes the first portion of the most-recently-traced instruction list to be displayed on the screen. The number of the list may range from one thru sixteen.

Each line of the list contains three entries. The first entry is a four character hexadecimal index to indentify the sequence of entries on the list. This facilitates finding a given place in the list when examining it a second time, or after doing supplemental tracing after a first examination of the list.

The second entry of the line is the value of the Stack Pointer just before the instruction was traced.

The third entry of the line contains the address of the instruction traced.

After the first portion of the most-recently-traced instruction list is displayed on the screen, successive 16-line additions are made to the display each time any key (other than a shift key) is depressed. When all 85 lines of the list have been displayed an 86th line is displayed that contains only the index of the next instruction to be stored in the list and a prompt character.

SELECTING THE INSTRUCTIONS SAVED IN THE MOST-RECENTLY-TRACED INSTRUCTION LIST

The address of every instruction traced is not normally stored in the most-recently-traced instruction list. The only addresses stored are those of instructions that are traced immediately after tracing "triggering" instructions. The normal triggering instructions are the BRK, JMP, JSR, RTI, RTS, and the branch instructions. These are sufficient for unambiguously reconstucting the program sequence from the most-recently-traced instruction list, together with a listing of the object program. This moderate skipping between triggering instructions lengthens the traced sequence over which the preceding program flow may be determined.

A command may be executed to make every instruction a triggering instruction, so that no skipping occurs between triggering instructions. Another command makes the skips between triggering instructions longer than normal by remomoving the branch instructions from the normal group of triggering instructions. These commands are summarized below:

L Long skips between triggering instructions - Details of program flow will be lost

M Moderate skips between triggering instructions - Program flow may be reconstructed

N No skips between triggering instructions - Every instruction is stored.

Typing the command letter and then pressing "RETURN" causes the command to be executed.

AUTOMATIC DISPLAY OF REGISTER CONTENTS

TRACE may be put into a mode where a visual display of register contents appears on the screen automatically each time an instruction is traced. The resultant heavy use of the screen by TRACE may prove to be an encumbrance when tracing object programs that use the screen, both from the standpoint of confusion on the screen and from the standpoint of program malfunction that may result if the object program calls the operating system's screen character printing subroutine and TRACE also calls it before the object program returns from it. TRACE attempts to recognize when this situation arises and prints a dashed line as a warning when such a conflict occurs. If a continuous trace is in progress whentthe conflict is recognized TRACE temporarily suspends the authomatic display of register contents until the object program returns from the screen character printing subroutine.

The commands associated with automatic display of register contents are as follows:

V Visual display of register contents occurs automatically
U Unencumbered mode - No automatic display of register contents.

Typing the command letter and then pressing "RETURN" causes the command to be executed.

EXITING FROM THE TRACE PROGRAM

When TRACE is not in the midst of a continuous trace, depressing a "Control C" will cause an exit from TRACE to the operating system.

4/13/81

Pegasus Software
p.o. box 10014 honolulu, hawaii 96816

LICENSE AGREEMENT

This software product is copyrighted by Pegasus Software. Any duplication or redistribution of this product is expressly prohibited and in violation of copyright laws. Each diskette containing this product has a unique serial number on file with Pegasus Software, both to protect the software and to serve as a means of notifying software package owners of updates in the form of corrections or improvements that may occur.

This product is sold on a one CPU basis. This means that purchasing one copy of this product entitles the user to utilize the product on one system only. The serial number of the computer as well as the license number of the software are to be registered with Pegasus Software.

Only licensed software owners will receive updates and corrections, so it is very important that the license form is completed and returned. Only licensed, Pegasus supplied diskettes can be restored.

WARRANTY

There is no warranty, either expressed or implied, for this product. It has been extensively tested by Pegasus Software and is believed to be reasonably error free. However, Pegasus does not guarantee this product in any fashion nor does Pegasus guarantee that all problems that may occur will be fixed. We will, however, under normal circumstances make an attempt to help with any problems that may occur and intend to support this software package to every extent possible. If you have any problems with this software or think you have found a problem please contact us. Please include a documented example of the problem.

Software License

Software Product: TRACE Version 1.0

Diskette Serial Number: 1008

System Serial Number: C3H 11933

Please keep this information for your records