7

DEBUGGER 5.0

Pages
1-6

DEBUGGER

GENERAL INFORMATION

The debugger, an octal debugging aid, enables the user to trace the actions of his program and to edit his program. It allows the user to examine the contents of any memory location and to change the contents if he so desires. The user can trace through his program, step by step or subroutine by subroutine. The trace portion also allows the user to test his program to find out if and when a certain instruction is executed or a certain memory location is referenced or reaches a given value. The debugger may also be used to slow down a user's program by running it interpretively.

Some of the nice features of the debugger are:

it only uses 1K of core

input from the keyboard is simple

the CRT output is fast and efficient

The debugger does not trace display processor instructions, but allows them to be executed.

TO USE

The user's program must be in core, but not using the top 1K of core (6000-7777 in a 4K machine).

Load the debugger with the TTY or PTR bootstrap in 40$_{(8)}$. The

The Debugger starts automatically. It can be restarted at location 6000 (for a 4K machine). 16000 FOR 8K MACHINE.
To stop the debugger switch DS bit 0 down. If the PDS-1 still doesn't stop, push STOP on the console.

EXAMINING AND CHANGING INSTRUCTIONS

Assume the user wishes to examine the contents of memory locations 100$_{(8)}$ through 105$_{(8)}$.

He should type control C (C^C) to activate the type in-type out. Then the user should type one, zero, zero, space. The PDS-1 will type a space, the octal contents of location 100$_{(8)}$, and another space. The user should type a period. The PDS-1 will then type CR-LF, 0101, space, the contents of 101$_{(8)}$, and another space. In this manner the user can continue to examine sequential memory locations by typing periods.

If the PDS-1 typed out the contents of 105$_{(8)}$ as 001032 and the user wanted the contents to be 001061, then immediately after the 001032 was typed out he should type zero, zero, one, zero, six, one, period or one, zero, six, one, period. The PDS-1 will store 001061 in 105 and type out location 106 and its contents.

In other words, to change a memory location, have the PDS-1 type out its old contents then type the new contents followed by a period.

TRACE

There are many ways to use the trace portion of the debugger program. However, in each method the user must tell the debugger at what address of core he would like to start tracing. To specify the starting address the user should type control S (S^C) followed by the octal address, followed by a period. The user is now ready to start tracing his program beginning at the address he just specified.

The simplest way to trace a program is instruction by instruction. To do this type a (A^C). The debugger will then execute the next instruction in the execution path of the user's program, and output the results via the CRT. The output is in octal format and consists of the instruction address, the instruction, the contents of the accumulator after the execution of the current instruction, and the contents of the link after execution. On all * instructions which alter or reference memory, the new contents or referenced contents are also printed. (The contents are actually altered.)

If the user desires to trace every statement of a program but also move through the program faster, he may type a (Q^C) which is the equivalent of 8 (A^C)'s.

* The execution of a JMS instruction alters memory, but is not printed as a memory altering statement. On an auto indexing command two memory locations are altered, but the index location is not outputted.

In some cases the user may have a long loop where he is only concerned with tracing a part of it each time the loop is executed. In a case like this it would be very convenient to have the debugger trace all statements in the loop, but only output information on a certain group of instructions. This is possible by defining output limits. For example, assume the users program has a loop from $2340_{(8)}$ to $2600_{(8)}$, and the critical portion is from $2505_{(8)}$ to $2522_{(8)}$ inclusive. To receive output on only the 2505 to 2522 portion, the user should type a (D^C) followed by "2505.2522." The user can then start the debugger at 2340, and when he types (A^C), the debugger will start tracing at 2340 but not output anything or stop until it has executed and outputted statement 2505. These outputting limits of $2505_{(8)}$ to $2522_{(8)}$ will stay in effect until they are changed.

There may also be cases where the user does not wish to trace the execution of a subroutine step by step, but would like to have that subroutine executed, find out the contents of the accumulator after the execution of the subroutine, and continue on in the program. The user can do this by tracing up to and including the jump subroutine (JMS) statement, and then typing a (W^c). The subroutine will be executed, but output will be suspended until the execution of the statement following the JMS. This method will not work on subroutines that return to places other than the normal place, one statement after their JMS call.

There are three kinds of events for which the user's program may be searched using the debugger. One event is the execution of the instruction in a particular memory location. To perform this kind of search, the user should type (Z^c) followed by the address of the instruction, followed by a period. The debugger will then trace from wherever it is in the user's program until it executes the instruction in the specified memory location. The debugger will output all information for the instructions it executes on the way which are within its currently defined output limits.

The second event which may be searched for, using the debugger, is the referencing of a specific memory location by the execution of a user's program's instruction. This search is very similar to the previous search, except that it is called by typing a (X^c) followed by the specific memory location address followed by a period, and it is terminated by the referencing of the specific memory location.

The third event is a specified location reaching a specified value. This search is initiated by typing a (E^c), location, value, period. It is terminated when the specified location reaches the specified value.

If the debugger is not coming back from one of these search commands, (control shift repeat ESC) will cause it to stop.

DISPLAY CONTROL

When the debugger is started at 6000, the screen is blank. As the program is used the screen gradually acquires character formation. (To save core, the only displayable characters are the octal digits, CR, LF, space, period, and burn screen - the latter available only as a special option). There is display buffer room for only about 350 characters. When this limit is exceeded a 3 line scroll takes place. At any time when the debugger is waiting for input, the screen may be cleared by typing (DEL^{C}).

OUTPUT (type in-type out)

address	contents	INPUT (period or new contents period)
0025	167234	.
0026	001061	001032.
0027	010534	

(trace)

address	instruction	C(AC)	(CL)	changed mem.address	new contents
0175	001033	000101	0		
0176	003003	001010	0		
0177	020743	001010	0	0743	001010
0200	060744	017777	0		
0201	003023	141777	1		

SUMMARY OF COMMANDS

DEL^{c}	clear screen
C^{c}	enter type in-type out
D^{c}	define tract output limits
S^{c}	start trace
A^{c}	another trace statement
Q^{c}	eight trace statements
W^{c}	trace until next address of user's program, no output in between (usually used to skip through subroutines).
Z^{c} [illegible]	search user's program for execution of specified instruction
X^{c} [illegible]	search user's program for reference to specified address
E^{c} [illegible]	search user's program until specified location reaches specified value
ESC^{csr}	stop searching