DECUS
PROGRAM LIBRARY

DECUS NO.	8-89
TITLE	XOD - Extended Octal Debugging Program
AUTHOR	Michael S. Wolfberg
COMPANY	Moore School of Electrical Engineering University of Pennsylvania Philadelphia, Pennsylvania
DATE	September 15, 1967
FORMAT	

University of Pennsylvania
THE MOORE SCHOOL OF ELECTRICAL ENGINEERING
Philadelphia, Pennsylvania

TO: Users of the PDP-8 and DEC-338

FROM: Michael S. Wolfberg

DATE: September 15, 1967

SUBJECT: XOD - Extended Octal Debugging Program

XOD is an octal debugging program for a PDP-8 with Extended Memory which preserves the status of program interrupt system at breakpoints. The program occupies locations 6430 through 7577 of any memory field.

From the on-line Teletype, the user can examine and modify the contents of any memory location. Positive and negative block searches with a mask may also be performed.

XOD includes an elaborate breakpoint facility to help the user run sections of his program. When this facility is used, the debugger also uses locations 0005, 0006, and 0007 of every memory field.

The ability to punch binary tapes is not included in XOD.

Loading, Starting, Restarting

XOD may be loaded into any memory field, and it occupies locations 6430 through 7577.

The STARTING ADDRESS of XOD is 7000.

During the operation of XOD or during the execution of a user's program, XOD may be restarted at 7000. When XOD is restarted at a time when it was already in control, none of the registers or indicators associated with breakpoints are affected.

Returning to Monitor

For use with PDP-8 operating systems, typing ctrl C causes XOD to jump to location 07600 with the accumulator clear.

Octal Numbers

An octal number may consist of any number of octal digits (one of the characters 0,1,...,7). Only the last four digits are taken as the number.

Illegal Characters, Errors

Whenever an illegal character is inputted or an improper command is typed, XOD responds with a question mark followed by CR/LF, and the line is ignored.

Field Specification

Bits 6 through 8 of location 6612 within XOD determine the highest memory field with which XOD is operating. The supplied tape of XOD has 0010 in location 6612, so that an 8-K memory is assumed. This number is used when XOD is running a user's program and is checked when a field specification is typed.

At any particular time, the attention of XOD is directed towards one memory field, and all specified locations refer to that field.

Typing an octal digit followed by # causes XOD to change its field of attention to the field specified by the digit. The MASK is set to 7777. If the field is the same one as the field in which XOD is running, the lower and upper limits are set to 0000 and 6427. Otherwise, the limits are set to 0000 and 7777.

When XOD is waiting for type-in, the Data Field lights indicate the current field of attention.

Examinations

Typing an octal number followed by / causes XOD to open the location specified by the number, and type out the contents of the location. If no location was already open, the current location (*) is set.

Current Location

* has the value of the location last opened when no location was already opened. (This includes searches.)

Although it is usually typed alone, * can be typed immediately following an octal number to signify addition. For example, typing 5*/ will open location *+5.

Carriage Return

If a location is open, typing an octal number followed by a CR causes XOD to set the contents of that location to the value of the octal number.

Line Feed

LF first acts like a CR and then opens location *+1.

Limits, MASK

The lower and upper limits (LL and UL) for zeroing and searching are stored within XOD at locations 7400 and 7401. The MASK used in searches is stored at 7402. These quantities are initialized when a user specifies a field of attention (by #). If the field of attention is set to the field where XOD is residing, the user may examine and modify these quantities. The following commands to XOD permit the user to set either of the limits or the MASK independent of the field of attention.

Setting LL

Typing an octal number followed by L causes XOD to set the lower limit (LL) to the value of the octal number.

Setting UL

Typing an octal number followed by U causes XOD to set the upper limit (UL) to the value of the octal number.

Setting MASK

Typing an octal number followed by M causes XOD to set the MASK to the value of the octal number.

Zeroing

Typing an octal number followed by % causes XOD to set the contents of locations LL through UL to the value of the octal number. If no number is typed preceding the %, a value of 0 is assumed.

Positive (Negative) Searching

Typing an octal number followed by > (<) causes XOD to search locations LL through UL. All locations whose contents (do not) equal the value of the octal number in the bits specified by the MASK are typed out. During the course of a search the user may stop the search by striking any key.

Reading Binary Tape

In order to read a binary tape, place the tape in the reader in the leader area and START the reader. XOD immediately jumps to location 7777 of the field where XOD is residing. There is presumably some type of binary loader at this location which then reads in the tape. After reading is finished, XOD may be restarted at its starting address.

Program Execution and Breakpoints

Most of the space occupied by XOD consists of the coding necessary for a thorough breakpoint facility. The user may command XOD to assign a breakpoint at any location of any field. This assignment has no immediate effect, but is important when the user commands XOD to start running his program. When he does this, XOD saves the contents of the location where the breakpoint was assigned and replaces it by a special breakpoint instruction (5005). XOD also plants special instructions into locations 0005, 0006, and 0007 of every memory field (without saving their contents). It then appropriately sets the accumulator, link, data field, Teletype output flag, and program interrupt status and jumps to the user's program.

The breakpoint instruction must not be inserted into the user's program as an instruction by location modification.

The user's program will execute normally unless control passes to the breakpoint location. In this event, a "breakpoint trap" occurs, which consists of a jump back to XOD. When XOD is re-entered, the location of the trap, accumulator, link, data field, Teletype output flag, and program interrupt status are all saved. At this time XOD types out the location of the breakpoint trap (including the field) followed by a right parenthesis followed by the contents of the link and accumulator.

The user may then use the facilities of XOD for examination and modification of any memory location. He may also remove the breakpoint assignment or reassign it to some other location. He may restart XOD at its starting address without affecting the saved status of the user's program.

If the user wishes to resume his program where it left off, he may command XOD to proceed. The proceed command causes XOD to restore the saved status, perform the instruction which is at the location of the last breakpoint trap, and to continue to run the user's program.

If a user's program doesn't return to XOD through a breakpoint trap, the user may restart XOD at its starting address. When this is done, the contents of the breakpoint location are restored and the saved accumulator, link, and Teletype output flag are all cleared. XOD will not allow the user to perform a proceed command until after the next breakpoint trap.

There is a HLT instruction at location 6430 of XOD to catch control in case a user's program runs wild.

Running a Program

Typing an octal number followed by ' causes XOD to:

1) if a breakpoint is assigned, set it up and also set up locations 0005 , 0006, and 0007 of every memory field.

2) set the data field to the field of attention.

3) have program interrupt off.

4) restore the saved accumulator, link, and Teletype output flag.

5) jump to the location (in the field of attention) specified by the number preceding the ' .

Breakpoint Assignment

Typing an octal number followed by " causes XOD to assign a breakpoint at the location (in the current field of attention) specified by the number. If no expression precedes the ", any breakpoint assignment is removed. Only one breakpoint may be assigned at a time, but it may be changed, even before proceeding back to the user's program.

A breakpoint should not be assigned at a location which is either modified or whose contents are used as data. Otherwise, there are no restrictions on breakpoint placement, as far as the breakpoint trap occurring. There are, however, three restrictions on the breakpoint assignment if the user wishes to proceed back to his program:

1) A breakpoint must not be assigned at any CIF instruction, nor at any instruction which follows a CIF instruction until after the next JMP or JMS instruction. More precisely, the restriction exists at locations where the contents of the instruction field register differ from the contents of the instruction buffer register.

2) A breakpoint must not be assigned at a location where the interrupt system could be on and where the program depends upon the preservation of the contents of the save field register.

3) A breakpoint must not be assigned at any of the following EAE instructions: MUY, DVI, SHL, ASR, LSR.

Proceeding

Typing a ! causes XOD to:

1) if a breakpoint is assigned, set it up and also set up locations 0005, 0006, and 0007 of every memory field.

2) restore the saved accumulator, link, data field, Teletype output flag, and program interrupt status.

3) perform the instruction which is at the location of the last breakpoint trap, and proceed from there.

If an octal number is typed preceding the ! , automatic proceeds are generated by XOD for the number of times equal to the value of that octal number before a breakpoint trap causes the debugger to regain control. (This facility is called multiple proceeding.)

Program Interrupt

When a breakpoint trap occurs, XOD determines the status of the program interrupt system and then the interrupt system is kept off during XOD operations.

When a breakpoint trap occurs, further program interrupts are disabled within a couple of memory cycles by the execution of a CIF instruction. Thus if a user wishes to use XOD on a 4-K PDP-8 to debug a program using the interrupt system, he must have an interrupt service routine which works.

If XOD is being used to debug a program occupying only memory field zero and using the program interrupt system, and if XOD is in another memory field, it is advisable to include an RMF instruction in the interrupt service routine. The chances of an interrupt occurring during one of the critical two machine cycles are rather small. However, if the user performs multiple proceeds, then there are many cycles during which the interrupt system is not disabled while control is within XOD. This necessitates a working interrupt service routine which includes an RMF instruction and preserves the accumulator and link.

State of XOD

Whenever XOD is in control, the user may examine and alter locations within the debugger which contain useful information:

Location	Contents
6600	saved link (may be only 0 or 1)
6601	saved accumulator
6602	saved data field (bits 6-8)
6603	saved Teletype output flag (zero = clear, non-zero = set)
6604	saved interrupt status (may be only 0 for off or 1 for on)
6605	proceed counter (is indexed by an ISZ instruction)
6606	breakpoint assignment field (bits 6-8) or if sign is minus, there is no breakpoint assigned
6607	breakpoint assignment location
6610	last breakpoint trap field (bits 6-8) or if sign is minus, proceeding is not allowed
6611	last breakpoint trap location
6612	highest field being used (bits 6-8)
7400	lower limit (LL)
7401	upper limit (UL)
7402	MASK