DECUS
PROGRAM LIBRARY

DECUS NO.	8-78
TITLE	DIAGNOSE: A Versatile Trace Routine for PDP-8 and EAE
AUTHOR	Keith B. Oldham
COMPANY	North American Aviation Science Center Thousand Oaks, California
DATE	July 19, 1967
SOURCE LANGUAGE	BIN

Diagnose: A Versatile Trace Routine for the PDP 8 Computer

K. B. Oldham
North American Aviation Science Center
Thousand Oaks, California 91360

Introduction

Trace routines are employed to track down logical errors in a program (the "sick" program). Starting at any convenient location in the sick program, instructions are executed, one at a time, and a record of all operations is printed out via the teletype. To avoid the waste of time that would ensue from tracing proven subroutines, an option is provided to omit subroutine tracing.

Trace routines have advantages over the more usual "break trap" debugging routines in being completely systematic in operation, in not requiring an operator's presence at the console, and in providing a complete and permanent record of the sick program's operation. On the other hand, they are purely diagnostic and do not enable the operator to make any corrections or modifications to his program.

Two trace routines (Decus Nos 8-56 and 8-57) for the PDP 8 computer were recently published by Biavati. The present routine is significantly more versatile than those of Biavati, being able to trace sick programs containing floating-point, extended arithmetical, and a variety of input/output instructions. Diagnose is, however, at a disadvantage compared with Biavati's first routine in requiring more memory space (five pages as opposed to two); and compared with his second routine in not possessing

the trace-suppression features of the latter. The mode of operation of Diagnose is quite different from that of the trace routines of Biavati.

Requirements

Diagnose is designed for a PDP8 computer equipped with the Extended Arithmetical Element. Five consecutive memory pages are needed (four for solely fixed-point operation). The tapes supplied use pages 17 through 22 (locations 34ØØ through 4577) but a single redefinition of ORIGIN (source tape I) will reposition the routine to occupy locations ORIGIN-2ØØ through ORIGIN+777.

Diagnose must not, of course, overlap the sick program, or any of its subroutines. A floating-point package is not incorporated into Diagnose and must be provided if floating-point tracing is required.

The floating-point section of Diagnose occupies its first page (locations ORIGIN-2ØØ through ORIGIN-1) and is programmed on source tape V. However, fixed-point operation of Diagnose in no way depends upon the floating-point section. Hence, an object tape assembled from source tapes I through IV will function perfectly for fixed-point operation and will use only four memory pages. Alternatively, if a full (five page) object tape is used for fixed-point tracing the first page may be overwritten with impunity.

Operation

Load Diagnose into memory (either before or after the loading of the sick program) by means of the binary loader.

Set the switch register to ORIGIN or ORIGIN+1 (36ØØ or 36Ø1 for the tape supplied) and press "LOAD ADD" followed by "START." The ASR-33 teletype will now print

TYPE THE INITIAL C(L):

and the operator should respond by typing either ØØØØ or ØØØ1 according as he wishes the trace operation to commence with the link cleared or set. The teletype will next ask for the initial accumulator contents, and then for the initial contents of the multiplier quotient. Each of these requests should be answered by typing the desired four octal digits, producing a record such as

```
TYPE THE INITIAL C(L):   ØØØ1
TYPE THE INITIAL C(AC):  1234
TYPE THE INITIAL C(MQ):  4567
```

At this stage the teletype will ask for the address in the sick program at which tracing is to begin, i.e., the initial contents of the program counter, by printing

TYPE THE INITIAL C(PC):

On receiving four octits (octal digits) in response to this request, the tracing procedure will commence without further user intervention, producing output formats as explained in the following section.

Most commonly the user will wish to have the link, accumulator, and multiplier quotient registers cleared at the start of the trace routine. To save the trouble of typing ØØØØ three times to achieve this clearance, the Diagnose routine should be started at ORIGIN+2 or ORIGIN+3. The teletype will then make the sole request

TYPE THE INITIAL C(PC):

Note that all these responses must be in the form of four octits; leading zeros cannot be omitted. Note also that no printout will be provided by the teletype until all four octits have been inputted.

The position of switch register bit 11 indicates whether or not subroutines are to be traced. If the bit is set (switch up) no tracing will be carried out. Either position of switch 11 may be selected initially and the switch may be repositioned at any time during the operation of Diagnose.

Tracing will continue either until the "stop" key is depressed or until a HLT instruction is encountered in the sick program.

Output Format

This format varies somewhat according to the type of instruction being executed, as follows

Operate Instructions

The output format when an operate instruction (group 1 microinstruction, group 2 microinstruction or combined microinstruction) is encountered is

```
ℓ accu     mquo
                    PROG     INST
L ACCU     MQUO
```

where ℓ, accu, mquo, L, ACCU, and MQUO are the contents of the link, accumulator, and multiplier quotient before and after the execution of the instruction, PROG is the contents of the program counter and INST is the instruction code. For example, if the RAR instruction was encountered at location 2Ø7 the output might be

```
1 Ø224    4567
                  Ø2Ø7    7Ø1Ø
Ø 4112    4567
```

EAE Instructions, First Group

This group, comprising the instructions NMI, MQL, SCA, MQA, CLA, and CAM, is handled in exactly the same was as operate instructions. For example, the MQL instruction could produce the printout

```
Ø 4112    4567
                  Ø21Ø    7421
Ø ØØØØ    4112
```

EAE Instructions, Second Group

The remaining EAE instructions, MUY, DVI, SHL, ASR, and LSR, require two memory locations to specify their operation and so need different treatment. The output format is best illustrated by an example. If the instructions MUY and 2 are contained respectively in locations 211 and 212, the following printout could result

```
Ø ØØØØ    4112
                  Ø211,12 74Ø5,ØØØ2
Ø ØØØ1    Ø224
```

Here we have an exception to the general rule that instructions are executed one at a time; in this case the two instructions 74Ø5 and ØØØ2 are treated together as a unit and the program counter will next contain Ø213.

AND, TAD, DCA, ISZ, and JMP Instructions

The output format in these cases is

```
 l accu     mquo
                      PROG     INST=MNE ADDR   PREV >POST
 L ACCU     MQUO
```

where MNE is the instruction mnemonic (AND, TAD, DCA, ISZ, or JMP). ADDR is the effective memory reference address, i.e., in the case of direct addressing it is the location (on the current page or page zero) specified in INST, whereas when indirect addressing is used it is the contents of the specified location. The contents of ADDR before and after the instruction is executed are represented by PREV and POST. For example, a sequence of memory reference instructions could produce the printout:

```
 Ø ØØØ1     Ø224
                      Ø213     53ØØ=JMP Ø3ØØ   1277 >1277
 Ø ØØØ1     Ø224
                      Ø3ØØ     1277=TAD Ø277   7776 >7776
 Ø 7777     Ø224
                      Ø3Ø1     3Ø66=DCA ØØ66   ØØØØ >7777
 Ø ØØØØ     Ø224
                      Ø3Ø2     2Ø66=ISZ ØØ66   7777 >ØØØØ
 Ø ØØØØ     Ø224
                      Ø3Ø4     1277=TAD Ø277   7776 >7776
 Ø 7776     Ø224
                      Ø3Ø5     1676=TAD 54ØØ   3451 >3451
 1 3447     Ø224
                      Ø3Ø6     Ø276=AND Ø276   54ØØ >54ØØ
 1 14ØØ     Ø224
```

JMS Instructions

If bit 11 of the switch register is clear, JMS instructions are handled exactly the same was as are the other memory reference instructions, leading to a printout such as

```
 1 14ØØ     Ø224
                      Ø3Ø7     424Ø=JMS Ø24Ø   ØØØØ >Ø31Ø
 1 14ØØ     Ø224
                      Ø241     ....
```

If, however, switch 11 is in the up position when a JMS instruction is encountered, the subroutine will be executed but not traced. The printout differs in having two linefeeds following the "JMS." The previous example would thus produce the following printout

```
l 14ØØ     Ø224
                    Ø3Ø7     424Ø=JMS
*
                                        Ø24Ø  ØØØØ >Ø31Ø
L ACCU     MQUO
                    Ø31Ø     ....
```

where L, ACCU, and MQUO are the register contents when the subroutine is exited. The asterisk shown above is not printed: it indicates the point at which any output or input called for by the subroutine occurs.

Input/Output Instructions

Diagnose is designed to handle IOT instructions for the following equipment: ASR33 teletype keyboard/reader/printer/punch, 750C high-speed reader, and the 75E high-speed punch. When the routine encounters such an instruction, it and the next two or three instructions are treated as a unit and executed in a single operation of the Diagnose routine. This procedure is necessary to ensure that the flags are correctly set. The instruction unit has four members in the case of the 750C reader, three members for the other equipment. Thus on encountering a TSF, JMP .-1, TLS sequence at locations 234 through 236 with Ø252 as accumulator contents, the printout would be

```
l Ø252     mquo
*                   Ø234,36 6Ø41,5234,6Ø46
 l Ø252    mquo
                    Ø237     ....
```

* is the character printed by the sequence. The output of a non-printing

character (line-feed, carriage-return or space) can be detected on the printout as a displacement (downwards, to the left, or to the right) of the next two lines of print. A printed character also causes a rightwards displacement, as in the above example.

JMS I 7 Instruction

The JMS I 7 instruction is interpreted by Diagnose as a call for entry to floating-point. It produces the format

```
ℓ accu     mquo
                    PROG     44Ø7=FENT
±Ø.XXXXXXXE±XX
```

where FENT is a mnemonic for floating-point entry and ±Ø.XXXXXXXE±XX denotes the contents of the floating accumulator on entry.

Floating-point Memory Reference Instructions

The output format for these instructions is very similar to that for fixed-point memory reference instructions, viz:

```
±Ø.xxxxxxxE±xx
                    PROG     INST=FMNE ADDR    ±Ø.yyyyyyyE±yy > ±Ø.YYYYYYYE±YY
±Ø.XXXXXXXE±XX
```

where ±Ø.xxxxxxxE±xx and ±Ø.XXXXXXXE±XX represent the contents of the floating accumulator before and after the floating-point operation. FMNE is the floating-point instruction mnemonic (FADD, FSUB, FMPY, FDIV, FGET, FPUT, or FNOR) and ADDR is the effective memory reference address. The contents of ADDR before and after execution of the floating-point instruction are ±Ø.yyyyyyyE±yy and ±Ø.YYYYYYYE±YY.

Expandable Floating-point Instructions

These instructions are the commands ØØØ1 through ØØ17 which the floating-point interpreter decodes according to the contents of the "look up" table." The output of Diagnose on encountering such an instruction is illustrated by the example of the square root operation:

```
+Ø.121ØØØØE+Ø9
                2311    ØØØ2=FØ2
+Ø.1Ø99999E+Ø5
                2312    ....
```

FEXT Instruction

This floating-point exit instruction produces an output format similar to that for an expandable instruction. The FEXT mnemonic is outprinted. Fixed-point operation is then resumed until the next encounter of a JMS I 7 instruction.

Restrictions

1. Diagnose cannot commence operation within a floating-point sequence.

2. Indirectly addressed autoindices are incorrectly traced.

3. Input/output sequences other than those for which Diagnose was specifically designed may encounter flagging difficulty.

4. A traced subroutine must be exited normally (i.e., return to the instruction following the JMS instruction) or suffer not more than three advances (i.e., return to up to the fourth instruction following JMS).

5. An untraced subroutine must be exited normally.

6. ORIGIN must be the first line of a memory page.