DOCUMENT RESUME

ED 092 153 IR 000 714

AUTHOR	Kirsch, Barry M.
TITLE	An Improved Error Diagnostics System for IBM System/360 -- 370 Assembler Program Dumps.
INSTITUTION	Ohio State Univ., Columbus. Computer and Information Science Research Center.
SPONS AGENCY	National Science Foundation, Washington, D.C.
REPORT NO	OSU-CISRC-TR-74-3
PUB DATE	Jun 74
NOTE	66p.; Master's Thesis, Ohio State University
EDRS PRICE	MF-$0.75 HC-$3.15 PLUS POSTAGE
DESCRIPTORS	*Computer Programs; Programing Languages; *Programing Problems
IDENTIFIERS	*Computer Programing Aids; IBM System 360 370; National Science Foundation

ABSTRACT

A system to aid in the post-mortem debugging of assembler language programs written for the IBM System/360-370 series of computers is described in this master's thesis. A user's manual for the system, with descriptions of user requirements, Job Control Language (JCL) statements, and system output, comprises the first chapter. The program logic structure is explained in the second chapter. The third chapter discusses the termination analysis routine logic, with details for each of the 15 different program interruption types handled by the system. Conclusions and directions for future work are presented in the fourth section. Finally, an appendix of sample programs, with a listing of all the JCL statements needed to use the diagnostics systems, is provided. (WDR)

TECHNICAL REPORT SERIES

[redacted]

[redacted]

COMPUTER & INFORMATION SCIENCE RESEARCH CENTER

THE OHIO STATE UNIVERSITY COLUMBUS, OHIO

(OSU-CISRC-TR-74-3)

AN IMPROVED ERROR DIAGNOSTICS SYSTEM FOR IBM SYSTEM/360 - 370 ASSEMBLER PROGRAM DUMPS

by

Barry M. Kirsch

Work performed in part under

Grant No. 534.1, National Science Foundation

The Computer and Information Science Research Center

The Ohio State University

Columbus, Ohio 43210

June 1974

PREFACE

This work was done in partial fulfillment of the requirements for the Master of Science degree in Computer and Information Science from The Ohio State University. It was supported in part by Grant No. GN 534.1 from the Office of Science Information Service, the National Science Foundation, to the Computer and Information Science Research Center of The Ohio State University.

The Computer and Information Science Research Center of The Ohio State University is an interdisciplinary research organization which consists of the staff, graduate students, and faculty of many University departments and laboratories. This report is based on research accomplished in cooperation with the Department of Computer and Information Science.

The research was administered and monitored by The Ohio State University Research Foundation.

ACKNOWLEDGEMENTS

I wish to express my thanks and appreciation to my adviser, Dr. Robert F. Mathis, for his advice, assistance, and encouragement in the preparation of this thesis.

Thanks are also due to Dr. Clinton R. Foulk for serving on my reading committee, and to Rick Baum for allowing me to use, essentially unmodified, his hexadecimal-to-decimal floating-point register conversion routine. Computer time for the development and testing of the system was provided by The Ohio State University Instruction and Research Computer Center. I would also like to acknowledge my support as a University Fellow by the Graduate School at The Ohio State University.

I wish to thank my parents for their moral support and encouragement during my educational career, and indeed, throughout my life. Finally, special thanks are due to my wife Debbie for her support, patience, and many good-natured sacrifices during the preparation and completion of this work.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	ii
INTRODUCTION	1
CHAPTER I - USER'S MANUAL	4
1) General	5
2) User Requirements	5
3) Necessary JCL Statements	9
4) Output Description	10
CHAPTER II - PROGRAM LOGIC STRUCTURE	13
1) General	14
2) Description of INTERUPT	17
3) General Description of the Termination Analysis Routines	19
CHAPTER III - TERMINATION ANALYSIS ROUTINE LOGIC	22
1) General	23
2) OC1 - Invalid Operation	23
3) OC2 - Privileged Operation	24
4) OC3 - Invalid EXecute	25
5) OC4 - Protection	26
6) OC5 - Addressing	27
7) OC45 - Traces Instructions for OC4 and OC5 Interruptions	27
8) OC6 - Specification	29
9) OC7 - Data	30
10) OC8 - Fixed-Point Overflow	31
11) OC9 - Fixed-Point Divide	31

TABLE OF CONTENTS - continued

Page

12) OCA - Decimal Overflow 32

13) OCB - Decimal Divide 32

14) OCC - Exponent Overflow 32

15) OCD - Exponent Underflow 33

16) OCE - Significance 33

17) OCF - Floating-Point Divide 34

CHAPTER IV - CONCLUSIONS AND DIRECTIONS FOR FUTURE WORK 35

1) Conclusions 36

2) Directions for Future Work 37

APPENDIX - SAMPLE PROGRAMS 39

BIBLIOGRAPHY 60

INTRODUCTION

Post-mortem debugging of assembler language programs is, for the most part, a difficult, time-consuming task requiring in many cases detailed familiarity with the system being used. Most assembler language programmers would surely benefit from improved post-mortem error diagnostics, yet very little practical work has been done in this area. With few exceptions, the standard manufacturer-supplied system dump is all the assembler programmer has recourse to in determining the reason for his program's abnormal termination. The system dump generally contains all the information necessary for the user to accurately deduce the cause of his error, but cloaks it among much irrelevant data in an extremely hard-to-read format. It is only the experienced programmer, knowledgeable in reading dumps and accustomed to the different sources of program errors, who does not have difficulty in poring over the pages upon pages of data in the typical system dump. Even he should benefit from a more compact, informative, and easier-to-read post-mortem diagnostic analysis.

Described herein is an error diagnostics system for assembler language programs written for the IBM System/360 - 370 series of computers. At the programmer's option, any or all of the 15 program interruptions can be trapped and handled by the error diagnostics package instead of or in addition to the standard system dump. Although the diagnostics system is not able to determine the precise cause of every possible program error, its use should facilitate the process of post-mortem program debugging.

The output produced by the diagnostics system is designed to provide only that information which would be of the greatest value to the user in debugging his program. No system control blocks or "raw" data are printed. Instead, relevant data items are taken from the control blocks, the program's core image, and data areas, and then programmatically analyzed; only that information which will aid the user most in the debugging process is printed, organized in a comprehensive, informative format.

The system was written for and tested on the IBM System/370 Model 165 computer operating under OS/MVT (operating system providing multiprogramming with a variable number of tasks), at the Instruction and Research Computer Center of The Ohio State University. Although designed specifically for use with this configuration, the system could be easily modified to run on other models of the IBM System/360-370 computer series operating under any of the IBM-supplied operating systems.

The author has attempted to incorporate into the system all of his knowledge of the various errors that may result in program interruptions on the computer in question, along with the many possible causes of these errors. Information of this type taken from a number of books and publications has also been included. However the logic contained in the programs comprising the error diagnostics system is by necessity incomplete. Other errors are sure to arise that are not covered by the program logic. In particular, among error conditions resulting from attempted input/output operations, logic related to only those caused by execution of the queued sequential access method (QSAM)

GET and PUT macros has been included. QSAM is by far the most commonly used access method in assembler language programs (especially in a student environment). Code covering the error conditions resulting from use of the macros of other access methods might be a desirable future enhancement. Since the system has been designed in a modular format, logic covering any additional error types and/or causes may be easily added at a later time.

The requirements for a programmer to use the system have been deliberately kept to a minimum. Although much of this thesis is written for the person with some knowledge of the details of the System/360 - 370 Operating System and its interrupt structure, little background is required to actually use the error diagnostics package. The reader with little or no experience is referred to the appendix where the first example given shows all the program and job control language statements necessary for the successful use of the system.

CHAPTER I

User's Manual

General

The error diagnostics system is capable of trapping and handling any or all of the fifteen program interruptions (system completion codes 0C1 - 0CF) produced by assembler language programs executing on the IBM System/360 - 370 series of computers. The user determines which interrupt conditions are to be trapped and whether or not a system dump is to be produced along with the diagnostic output. Only one macro instruction within the user's program, and two or three extra job control language (JCL) cards (described below) are required to use the system. Since, at the user's option, a post-mortem system dump is also produced, any user may use the package without fear of not getting some information that would otherwise be produced using the standard system interface. Just as with the system dump, the use of the error diagnostics system results in the termination of the user's job; recovery from an interrupt condition is not possible.

User Requirements

The macro instruction should be included in the first control section (CSECT) in the task for which the error diagnostics system is to be in effect. The exit routine will then be given control when any of the specified program interruptions occurs in any program of the task.

The macro instruction is written as follows:

SPIESET name,(interruptions)

where:

name is the name to be given to a control section automatically

generated in the user's region by the expansion of the macro. It may be 1 - 6 characters in length and should be distinct from all other names used in the user program. Three labels, name prefixed with SP, ST, and SV, will also be generated - these too should be distinct.

interruptions is one or more decimal numbers, separated by commas, indicating the corresponding interruption types shown below. The interruption types can be designated in any order as follows:

a) one or more single numbers, each indicating the corresponding program interruption type, or

b) one or more pairs of decimal numbers, each pair indicating a range of corresponding interruption types. The second number must be higher than the first. The pairs of numbers must be separated from each other by commas and enclosed in an additional set of parentheses.

For example, a second operand of (4,8) indicates interruption types 4 and 8; ((4,8)) indicates interruption types 4 through 8, inclusively. ((1,15)) indicates all interruption types 1 through 15; ((3,11),(13,15)) indicates all interruption types except 1, 2, and 12. The interruption types are as follows:

Number	Interruption Type
1	Invalid Operation
2	Privileged Operation
3	Invalid EXECUTE
4	Protection
5	Addressing
6	Specification
7	Data
8	Fixed-Point Overflow (maskable)
9	Fixed-Point Divide
10	Decimal Overflow (maskable)
11	Decimal Divide

Number	Interruption Type
12	Exponent Overflow
13	Exponent Underflow (maskable)
14	Significance (maskable)
15	Floating-Point Divide

Care should be taken in the specification of the interruption types. If a maskable interruption type is specified, the corresponding program mask bit in the program status word is set to 1, thus allowing the interruption to occur. If the programmer desires to mask any of these conditions (i.e., not permit an interruption when the condition is present), he should not specify the corresponding interruption number. Both operands of the SPIESET macro must always be coded; there are no default values.

The result of the macro call is the setting of a SPIE macro (which is included in the expansion of the SPIESET macro), and the production of a small control section in the user's region which, as a result of the SPIE, is given control at time of interrupt and then links to the error diagnosis routines. Although all the other routines comprising the error diagnosis system are reentrant, the CSECT produced is not in order to minimize in-core code and execution time overhead. Since the SPIESET macro is designed for use primarily in a "driver" program within a testing environment, the fact that the generated control section is not reentrant should not be considered a significant drawback. However, the error diagnostics system should not be used by a program which will actually be run simultaneously by multiple users.

If the user desires to change the interruption type specifications after issuing a SPIESET macro instruction, he may issue another SPIESET,

again specifying the name to be given to a generated control section and the interruption types upon the occurance of which the error diagnosis system is to be given control. The control section name specified must follow the rules given above and must be distinct from the name(s) specified in any previously issued SPIESET macro instruction(s), since another control section will now be generated with the name as given. The interruption types specified in the newly issued SPIESET should be complete - they totally override any previous specifications, and the previously generated control section(s) are no longer given control under any circumstances.

To cancel the effect of a SPIESET macro instruction, a SPIE macro with no operands may be issued. After a "cancel" SPIE has been issued, the standard control program exit routine is once again given control upon any program interrupt condition.

Since the SPIESET macro is generally issued only in a main or "driver" program within a testing environment, it is usually not necessary to reestablish the effect of any previously-issued SPIE macro before returning control. However, the user should make sure that the program that issues the SPIESET macro instruction does not return control to a calling program, or transfer control (by issuing an XCTL macro instruction) to another program that is not fully debugged, and for which the error diagnosis system is not to be in effect, before issuing a "cancel" SPIE.

Necessary JCL Statements *

In the JCL for the assembler step in which the macro instruction is included, the user should concatenate the FRA380.KIRSCH macro library with the system macro library (SYS1.MACLIB) and any other macro libraries he might be using. This data set contains the definition of the SPIESET macro.

A JOBLIB or STEPLIB DD card specifying the FRA380.KIRSCH2 data set must also be included in the user's JCL stream. It is on this data set that the object modules of the various termination analysis routines are kept. If a STEPLIB DD card is used, it should be included in the JCL for the user program's execution (GO) step.

A SYSUDUMP or SYSABEND DD card should be included in the JCL statements for the user program's execution step if a system dump is desired in addition to the error diagnosis output. If no SYSUDUMP or SYSABEND DD card is provided, no dump will be produced.

The error diagnostics system writes its output on a data set whose DDNAME is SYSPRINT. The user must thus provide a SYSPRINT DD card in the execution step directing the error diagnosis output to the desired device. Normally the output would be directed to the printer; in this case SYSOUT=A should be coded. Note that SYSPRINT is the DDNAME commonly assigned to the normal program printer output. This has been designed so that in many cases an additional JCL card will not be required, and the error diagnosis output will follow directly any output the user program may have produced before its termination.

---

* The data set names given in this section are those in use at the Instruction and Research Computer Center at The Ohio State University as of May, 1974. It is expected that these data sets will continue to be available.

A sample deck of a user job including assembler and loader (execute) steps, employing the ASMGRUN catalogued procedure, is given below. All the necessary JCL cards are included:

Notes

```
      //      Job Card
1)    //JOBLIB DD DSN=FRA380.KIRSCH2,DISP=SHR
      //STEP1 EXEC ASMGRUN
      //CMP.SYSLIB DD
      //           DD DSN=FRA380.KIRSCH,DISP=SHR
      //CMP.SYSIN  DD *

          User program including SPIESET macro instruction

      /*
1)    //GO.STEPLIB  DD DSN=FRA380.KIRSCH2,DISP=SHR
      //GO.SYSPRINT DD SYSOUT=A
2)    //GO.SYSUDUMP DD SYSOUT=A
3)    //
```

Notes: 1) Either of the JOBLIB or STEPLIB DD cards, but not both, is required.

2) The SYSUDUMP DD card should be included only if a system dump, in addition to the error diagnosis output, is desired.

3) Other DD cards may be needed in the GO step depending on the individual user program requirements.

The reader is referred to the first example in the appendix, where an actual computer run is shown including all required user-supplied JCL statements and program instructions necessary to use the diagnostics system.

Output Description

The output produced by the error diagnostics system is in two parts. First is general information, common to all 15 interruption types.

In this section output first is the message EXECUTION ERROR followed by the system completion code (OC1-OCF, corresponding to program interruption types 1-15, respectively) and a brief description of the interruption type. Next the location of the interrupt is printed with the name of the CSECT in which it occurred and its relative address, if it occurred within the user's program area. If the interrupt location falls outside this area, a note to that effect is printed. Next output are the contents of the 16 general purpose registers and four floating-point registers at time of interrupt in both hexadecimal and decimal representations. This is followed by the hexadecimal representation of the eight bytes of memory starting at the interrupt location, and a reconstruction of the instruction causing the interruption when this can be positively determined - i.e., for all interrupt types other than OC1 (invalid operation), OC4 (protection), and OC5 (addressing).

The second part of the output is diagnostic information keyed to the specific interruption type and its possible causes. This information includes a message explaining the cause of the interrupt in general (the messages produced for each interrupt type are described in Chapter III). This is followed by a detailed explanation of the precise cause of the error, insofar as this can be determined, and, whenever possible, debugging hints aimed at correcting the error condition.

In the case of an invalid operation interruption (OC1), reconstruction of the instruction causing the exception is impossible, but an error diagnosis is printed and debugging aids are provided. Due to the instruction look-ahead feature of many models of the System/360 and 370 computers, protection and addressing interruptions (types OC4 and

OC5) are imprecise, and it is thus impossible to determine exactly the instruction causing the interruption. In these cases the instructions immediately preceding the interrupt location are reconstructed and printed, those instructions capable of causing the interruption being flagged. This procedure is explained in more detail in Chapter III.

CHAPTER II

Program Logic Structure

General

This chapter describes the structure of and the linkages between the various modules that combine to make up the error diagnostics system.

The first component of the system is the user program interface. In order to utilize the package, a programmer must include a SPIESET macro instruction in the first control section of his job for which he desires the error diagnostics system to be in effect. This would generally (but need not) be the main or "driver" control section of the job.

The SPIESET macro instruction expansion includes the following:

1) the expansion of a SPIE (Set Program Interrupt Element) macro instruction, and

2) the creation of a small control section in the user's program area.

The SPIE macro instruction is used to specify an alternative exit routine to be given control when any of the specified program exceptions occur. This allows the user to bypass the processing of the standard control program exit routine which would otherwise be given control and abnormally terminate the task, producing a standard system dump. After the SPIESET macro instruction is executed control is transferred instead to the generated control section upon any specified program interruption. The generated CSECT then brings the major routine of the error diagnostics system into main storage and passes control to it by issuing a LINK macro.

The expansion of the SPIE macro instruction results in the formation of a program interruption control area (PICA) which contains the new program mask for the interruption types that can be disabled, and a

code for the interruption types specified in the SPIESET macro. Also included in the PICA is the address of the new CSECT (Figure 1). If the SPIESET macro instruction specifies an exception for which the interruption has been disabled, the control program enables the interruption when the macro instruction is issued.

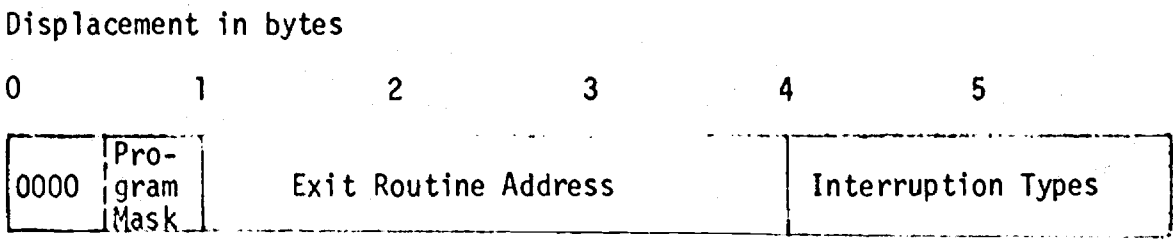

Figure 1. Program Interruption Control Area

Upon execution of the SPIESET, and thus the SPIE, macro, the control program creates a 32 byte program interruption element (PIE) in the main storage area assigned to the job step (Figure 2).

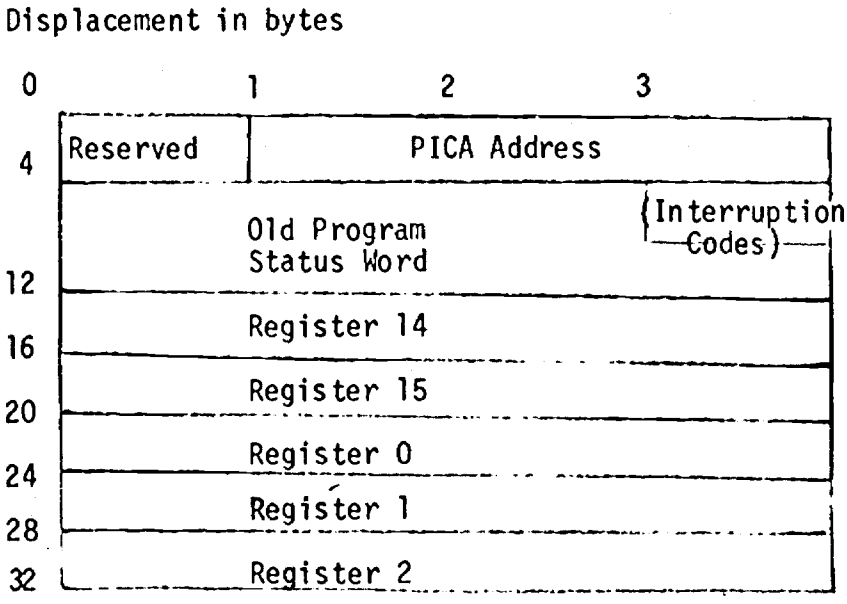

Figure 2. Program Interruption Element

The PICA Address in the program interruption element is the address of the program interruption control area created by the expansion of the SPIE macro. When control is passed to the generated control section (designated in the PICA and hereinafter referred to as the exit routine) the program status word and the contents of general purpose registers 14, 15, 0, 1, and 2 at time of interrupt are stored in the PIE by the control program as indicated. The register contents are as follows when the exit routine gains control:

Register 0: internal control program information

Register 1: address of the program interruption element

Registers 2 - 12: same as when the program interruption occurred

Register 13: address of the save area for the user program causing the interruption

Register 14: return address (to the control program)

Register 15: address of the exit routine

When control is passed to the exit routine it stores these register contents in a special 16 word save area and then executes a LINK macro passing control to the major control section of the error diagnostics system (INTERUPT). When INTERUPT returns control, the exit routine restores the registers from the special save area and then returns control to the control program using the address passed in register 14. This does not result in a normal return to the user program, however, since the old program status word and some of the register fields of the PIE are changed by the error diagnostics system before returning control. This procedure is explained in more detail at the end of this chapter.

Description of INTERUPT

The INTERUPT control section forms the mainstay of the error diagnostics system. Passed control by the user program's exit routine, it performs all the general error diagnosis functions, determines the interruption type, and then passes control to the appropriate termination analysis routine.

One of the functions of the INTERUPT control section is the printing of both the general purpose and the floating-point register contents at the time of the interruption. To this end, it creates two special register save areas. Since the contents of the floating-point registers have not been changed since the time of interruption, their contents are stored directly into the floating-point register save area. The contents of general purpose registers 3 - 13 at time of interrupt are moved into the special save area from the save area in the user program's exit routine. The contents of registers 14, 15, 0, 1, and 2 at time of interrupt are found in the program interruption element and moved into the special save area from there.

Bits 28 - 31 of the PSW at time of interrupt (found in the PIE) give the number (1 - 15) of the program interruption which occurred. The program interruption number is translated into the corresponding system completion code (0C1 - 0CF) and is printed along with a general description of the interruption type taken from a table in the routine.

The location of the interruption is next printed with the name of the user program control section within which it occurs and its relative address within that CSECT. If the location of the interrupt falls outside the user's program area, a message to that effect is printed in-

stead. In this case the name of the control section being executed at the time of interrupt is printed in the specific error diagnosis section of the output.

Next printed are the contents of the sixteen general purpose and four floating-point registers at the time of interruption (now stored in the two special register save areas previously mentioned). The register contents are printed out in both hexadecimal and decimal representations. For the general purpose registers, the decimal representation is in the form of simple signed integers; for the floating-point registers the form is standard scientific or "E" notation - a normalized decimal number with point raised to a power of ten.

For all precise interruption types (all those except OC4 and OC5, protection and addressing, respectively) the hexadecimal representation of the eight bytes starting at the location of the interruption is next printed. This is followed by a reconstruction of the instruction causing the interrupt whenever this is possible (for all interruption types other than OC1, OC4, and OC5). This is accomplished by executing a LINK macro passing control to routine DECODER which reconstructs the instruction from its object code and then returns control to INTERUPT from where the reconstructed instruction is printed. DECODER makes use of an instruction list (INSTRLST) which includes the mnemonic and instruction type for all System/370 instructions, as well as all the program interruptions each instruction is capable of causing. The instruction list is brought into main storage by means of a LOAD macro instruction executed from the INTERUPT control section. It is used by the termination analysis routines for the imprecise interruption types OC4 and OC5; for

all other interruption types its space is relinquished after DECODER returns control, by means of a DELETE macro also executed from INTERUPT.

This ends the functioning of the general error analysis section. The floating-point register contents are restored and control passes to one of the fifteen termination analysis routines, depending on the interruption type. Control is passed by means of a LINK macro.

General Description of the Termination Analysis Routines

There are fifteen termination analysis routines, one corresponding to each of the program interruption types. Rather than being general in nature as is the INTERUPT control section, each of the termination analysis routines is keyed to one specific interruption type and the possible causes for that error condition. They are brought into main storage only when needed and invoked by means of a LINK macro executed in the INTERUPT control section. Although each routine is a separate entity in itself, similarities exist among the functions performed. This common functioning is the subject of this section.

When each routine receives control it prints out a short message generally describing the error condition. This message does not attempt to give the cause of the error - it is simply a description of the interruption type. In the case where the interruption occurred outside the user's program area, this line is followed by a line giving the name of the control section being executed at the time the interruption occurred.

Following this is a detailed analysis of the probable cause(s) for the error condition, based on the interruption type, the instruction causing the exception, its location, and other factors. This processing

is detailed in Chapter III where the functioning of each of the fifteen termination analysis routines is described.

At the conclusion of each routine's detailed error analysis, control is returned in such a way as to force the production of a system dump reflecting the original program conditions at time of interruption. This is done by placing the address of the instruction causing the interrupt in the second half of the old program status word stored in the program interruption element. A SPIE macro is then issued specifying a PICA address of zero, thus cancelling the effect of the SPIESET macro. When control is returned the PSW is reloaded from the old PSW field of the PIE; this effects a return to the interrupt-causing instruction and causes its subsequent attempted re-execution. When the interruption occurs this time, control is not passed to the exit routine (due to the effect of the "cancel" SPIE), but instead a system dump is produced. This is the standard action taken. The system does not allow for the resumption of the user program's execution from the point of interruption.

In the case of imprecise interruptions (types OC4 and OC5) this procedure is modified somewhat. Since it is not always possible to determine the instruction causing the exception, it is similarly not always possible to force the interruption to recur. In this case the hexadecimal code for the ABEND SVC (0A0D) is placed at the interrupt location. The contents of the old PSW field in the program interruption element are changed to reflect this address and the saved contents of register 1 in the PIE are altered to reflect the desired system completion code. Control is still returned to the interrupt-causing location,

but now an ABEND macro is executed, again producing a system dump reflecting in almost all respects the program conditions at the time of the actual interruption.

This is also done in the case of those interruptions for which the operation causing the exception is not suppressed (types 0C7, 0C8, 0CA, 0CC, 0CD, and 0CE). This is necessitated by the possible elimination of the interrupt-causing condition by the total or partial completion of the operation. The programmer is notified when this special procedure is used by a message included at the end of the error diagnosis output.

CHAPTER III

Termination Analysis Routine Logic

General

This chapter details the logic of the 15 termination analysis routines corresponding to the 15 program interruption types handled by the error diagnostics system (system completion codes 0C1 - 0CF). The routines are named 0C1 - 0CF, each name reflecting the interruption type handled by the program. There is also an 0C45 routine which performs the instruction traces required by the 0C4 and 0C5 error diagnosis programs.

Although actually a description of program logic, this chapter is written so that it could be used in itself as a non-automated assembler program debugging manual.

In each case, the standard error message which is printed out is given first, followed by a breakdown of the logic used to arrive at the possible causes of the interrupt condition.

0C1 - Invalid Operation

The operation code in the instruction causing the interrupt is invalid - no such operation exists on this computer.

Check the location of the interrupt (PSW at entry to ABEND):

1) If the interrupt location is within the user program area, the cause of the exception is most probably an attempt to execute data as an instruction. A previous move instruction may have placed data in the program area, or the contents of the program base register may have been destroyed.

The ABEND PSW will point to the next instruction to be executed

in the program.

2) If the interrupt location is 50, 52, 5002, or 5200 the most probable cause of the interrupt condition is an attempt to access an unopened or improperly opened data set:

If a GET was attempted, the interrupt location will equal 5002 or 5200; if a PUT was attempted, the interrupt location will equal 50 or 52. In either case, the contents of the following registers at time of interrupt should be investigated:

Register 1 points to the Data Control Block (DCB) of the data set involved;

Register 1 + 40 (decimal) points to its DDNAME;

Register 14 points to the next instruction to be executed in the user program.

3) If the interrupt location is outside the user program area and is not one of the above noted locations, the most probable cause of the error condition is a branch to a data area resulting in the fetching of data as the operation code for an instruction. A possible cause of this condition is the alteration of the program's base register contents. Check the contents of register 14 at entry to ABEND for (possibly, but not always) the next instruction to be executed in the program or the instruction following the last BALR branch to a subroutine.

OC2 - Privileged Operation

The instruction causing the interrupt is valid only in the supervisor state - it cannot be executed by the user program.

Check the location of the interrupt (PSW at entry to ABEND):

1) If the interrupt location is within the user program area, the cause is most probably an attempt by the problem program to issue a privileged instruction.

A less probable cause would be an attempt to execute data as an instruction (an OC1 interrupt would much more commonly result from this condition). This might be the result of a previous move instruction placing data in the program area, or the alteration of the program's base register contents.

In either case, the ABEND PSW will point to the next instruction to be executed in the user program.

2) If the interrupt location is outside the user program area, the most probable cause of the error condition is a branch to a data area resulting in the fetching of data as the operation code for an instruction (although an OC1 would, again, much more commonly result from this condition). This might be the indirect result of the alteration of the user program's base register contents. Check the contents of register 14 at entry to ABEND for (possibly, but not always) the next instruction to be executed in the program or the instruction following the last BALR branch to a subroutine.

OC3 - Invalid EXecute

The subject of the EXecute instruction causing the interrupt is another EXecute instruction.

The instruction causing the interruption is the first EXecute instruction. It, in turn, points to the subject EXecute instruction.

OC4 - Protection

The user program has attempted to change the contents of a protected area of main storage.

Possible causes of this error are:

1) If register 15 at time of interrupt contains 02005000 or 02005200, the cause is most probably an attempt to execute a GET macro referring to an unopened or improperly opened data set.

2) If register 15 at time of interrupt contains 02000050 or 02000052, the cause is most probably an attempt to execute a PUT macro referring to an unopened or improperly opened data set.

In either case, register 14 at time of interrupt will point to the next instruction to be executed in the user program.

3) The error might also result from an uninitialized index, or an attempt to index outside the program's assigned limits. This condition might be indirectly caused by the alteration of the user program's base register contents.

If the interrupt location is outside the user program area, the name of the CSECT being executed at time of interrupt is printed.

Due to the instruction look-ahead feature of many models of the System/360 - 370 computer series, protection interruptions are imprecise. If the interrupt location is within the user program area, control is passed to the OC45 routine which prints out the instructions immediately preceding the interrupt location, flagging those that could have caused the interruption. Only those instructions that attempt to store data in main storage can cause a protection interruption.

OC5 - Addressing

An address of a specified instruction or data is outside the limits of the computer's available storage.

Possible causes of this error are the specification of an invalid data address, an invalid index, or an attempt to index outside the program's assigned limits. Any of these causes might be the indirect result of the alteration of the user program's base register contents.

If the interrupt location is outside the user program area, the name of the CSECT being executed at time of interrupt is printed.

Due to the instruction look-ahead feature of many models of the System/360 - 370 computer series, addressing interruptions are imprecise. If the interrupt location is within the user program area, control is passed to the OC45 routine which prints out the instructions immediately preceding the interrupt location, flagging those that could have caused the interruption. Any instruction that accesses main storage in any way can cause an addressing interruption.

OC45 - Traces Instructions for OC4 and OC5 Interruptions

Due to the instruction look-ahead feature of many models of the IBM System/360 - 370 computer series, addressing and protection interruptions are imprecise, and it is thus often impossible to determine the exact instruction causing the interrupt condition. If the location of the interrupt occurs inside the user program area, both the OC4 and OC5 termination analysis routines pass control to the OC45 module, whose function is to reconstruct (using the DECODER routine) and print out the instructions immediately preceding the interrupt location.

Starting 36 bytes back from the interrupt location, the routine attempts to decode all the object code down to and including the interrupt location. If the attempt fails at some point (i.e., if there is object code not corresponding to a valid instruction), the process begins again, starting 34 bytes back from the interrupt location. If this attempt fails, the process begins again at 32 bytes, 30 bytes, etc., until all the object code down to and including the interrupt location can be successfully translated to valid IBM System/370 assembler instructions. All the reconstructed instructions are printed along with their object code and absolute and relative memory addresses.

The routine also flags all those instructions that possibly could have caused the interrupt condition. It does this by comparing the list of program interruptions each instruction is capable of causing (coded along with the mnemonic and instruction type code in the instruction list) with the interrupt condition that has occurred. If a match is found the instruction is flagged. This is designed to aid the debugging process by limiting the number of instructions a programmer need consider in determining the reasons for his program's abnormal termination. In some instances only one instruction will be flagged, most probably indicating precisely the interrupt-causing instruction.

It should be noted that only those instructions <u>physically</u> preceding the interrupt location are decoded and printed. It is possible that a branch might have been executed to the region of the interrupt location after the interrupt-causing instruction was issued, but before the interruption condition was detected. This circumstance would invalidate the chain of instructions printed, but it is impossible to programmati-

cally determine if this has, indeed, occurred. The user should consider this possibility before relying on the diagnostic data provided.

OC6 - Specification

An operand specification in the instruction causing the interruption is incorrect.

Possible causes of this error are:

1) An odd register operand address is specified in an instruction requiring an even register operand address (i.e., D, DR (first operand), M, MR (first operand), and the double shift operations).

2) An invalid floating-point register address (1, 3, 5, 7-15) is specified. Only 0 or 4 can be specified for an extended operand.

3) A branch has been attempted to an odd address. The ABEND PSW points to the branch destination.

4) The length of the second operand of a MP or DP instruction is greater than 8.

5) The first-operand field is shorter than or equal to the second-operand field in a MP or DP instruction.

6) An instruction address does not designate a location on an even-byte boundary.

7) The block address in an SSK or ISK instruction does not have zeroes in the four low-order bit positions.

8) An operand address does not designate an integral boundary in an instruction requiring such integral boundary designation. This condition cannot occur on System/370 computers for any instructions other than Compare and Swap (CS) and Compare Double and Swap (CDS) due to the

byte-oriented operand feature.

9) A PUT or GET macro instruction has been attempted referring to an unopened or improperly opened data set, or a data set with a DDNAME whose corresponding DD JCL card is missing or misspelled. All system indicators upon either of these two conditions are identical, and all addresses and register contents at time of interrupt are the same as those for the OC1 interrupts caused by the same conditions.

OC7 - Data

Data in a field is of incorrect format for the instruction attempting to process it.

Possible causes of this error are:

1) The sign and/or digit codes of operand(s) used by a CVB, AP, SP, ZAP, CP, MP, DP, ED, or EDMK instruction are invalid, i.e., not in the packed decimal format.

2) The operand fields in a AP, CP, DP, MP, or SP instruction overlap in a way other than with coincident rightmost bytes; or operand fields in a ZAP instruction overlap, and the rightmost byte of the second operand is to the right of the rightmost byte of the first operand.

3) The first operand of a MP instruction has too few high-order zeros.

Any of these three error causes may result from:

a) an uninitialized data field (e.g., blanks might have been read into a field designed to be processed with packed decimal instructions), or

b) an incorrect or uninitialized index, resulting in invalid data

being referenced.

A data exception occurs during the execution of an instruction, and the operation is suspended at that point; the contents at the result field are unpredictable. An exception to this rule is that, except for ED and EDMK instructions, the operation is suppressed when a sign code is invalid, regardless of whether any other condition causing the exception is present.

OC8 - Fixed-Point Overflow

Two distinct causes of this error condition are possible:

1) A carry has occurred out of the high-order bit position of the result register in a fixed-point arithmetic operation - the result of the operation causing the interrupt is too large to be expressed in 32 bits in the 2's complement form.

2) High-order significant bits have been lost during an algebraic left shift operation.

In either case, the operation is completed with the result left in the register too large or too small by an even multiple of 2**31. The interruption could be suppressed by setting PSW bit 36 to a zero. This would cause the overflow condition to be ignored.

OC9 - Fixed-Point Divide

Two distinct causes of this error condition are possible:

1) The program has attempted a binary integer division by zero, or the development of a quotient too large to be expressed in 32 bits

in the 2's complement form (caused by either a D or DR instruction). The operation has been suppressed.

2) The result of a CVB instruction is too large to be expressed in 32 bits in the 2's complement form. The operation has been completed by ignoring the high-order bits that cannot be placed in the register.

OCA - Decimal Overflow

The destination field in the decimal operation AP, SP, or ZAP is too small to contain the result, forcing one or more significant high-order digits to be lost.

The result has been truncated on the left but lower-order digits and the sign are exactly as they would be in a longer field sufficient to hold the result.

The interrupt could be suppressed by setting PSW bit 37 to a zero. This would cause the overflow condition to be ignored.

OCB - Decimal Divide

The quotient formed by the decimal division (DP) instruction causing the interruption exceeds the specified data field size. This might be the result of an attempted division by zero.

OCC - Exponent Overflow

The result of the floating-point operation causing the interruption is 16**64 or greater - the result characteristic exceeds 127 (hexadecimal 7F) and the result fraction is non-zero.

The overflow can occur as the result of a carry-out of the high-order fraction position during normalized or unnormalized addition or subtraction and the following characteristic adjustment after a right shift, or during the characteristic computations in multiplication or division.

The operation has been completed. The fraction is normalized, and the sign and fraction of the result remain correct. The result characteristic has been made 128 smaller than the correct characteristic.

OCD - Exponent Underflow

The result of the floating-point operation causing the interruption is smaller than 16**-64 - the result characteristic is less than zero and the result fraction is not zero.

This condition could occur as the result of normalization during normalized addition or subtraction. It might also result from multiplication, division, or halving.

The operation has been completed with a result whose characteristic is 128 larger than the correct characteristic. The fraction is normalized, and the sign and fraction remain correct.

The interrupt could be suppressed by setting PSW bit 38 to a zero. This would cause the operation to be completed by replacing the result with a true zero.

OCE - Significance

The result fraction of a floating-point addition or subtraction (normalized or unnormalized) is zero - all significant digits of the

result have been lost.

The operation has been completed without further change to the characteristic and sign of the result.

The interrupt could be suppressed by setting PSW bit 39 to a zero. This would cause the operation to be completed by replacing the result with a true zero.

OCF - Floating-Point Divide

A floating-point division by a number with a zero fraction has been attempted. The operation has been suppressed with the dividend left unchanged.

CHAPTER IV

CONCLUSIONS AND DIRECTIONS FOR FUTURE WORK

Conclusions

The major goal of this thesis has been the development of an improved error diagnostics system for use in the post-mortem debugging of assembler language computer programs. That goal has largely been reached, and the system described herein represents a significant improvement over the facilities previously available.

The long range goal of debugging research should be the elimination or at least a greatly reduced reliance on system dumps for post-mortem debugging analysis. While this objective is still to be realized, this system hopefully represents an important first step in this direction. For many error conditions, the diagnostic output provided will be sufficient for the user to accurately determine the source of his error. In those cases where it does not suffice, its use in conjunction with the system dump (which is still available at the programmer's option) should speed the debugging process.

While not being all inclusive, the system presently handles many of the errors commonly made by beginning and intermediate assembler language programmers, and is especially well-suited for use in a student environment. As the system is enlarged to handle more termination conditions, and those that it does handle in more detail, it should find application in numerous other environments.

One looks forward to the day when systems such as this one will replace the system dump as the standard manufacturer-supplied post-mortem error diagnosis package. This will only come to pass when the error diagnoses provided yield as much information as do the present system dumps, but with far less extraneous data and in a much more

intelligible format. A great deal more work will need to be done before this aim becomes reality. One may even visualize operating systems and other software packages being designed in the future with the debugging function specifically in mind. Much more research is needed in this very practical area.

Directions for Future Work

The error diagnostics system as currently configured represents only a starting point towards the development of a truly comprehensive post-mortem debugging system for the IBM System/360 - 370 computer series.

The present system is capable of trapping and handling only the program interruption types 0C1 - 0CF. An obvious extension to the system would be to include in it capabilities for handling all the other abnormal termination error conditions. However this would necessitate revamping much of the system structure as the SPIE macro allows for handling only the fifteen program interruption types. Also, the system as currently configured is not designed for handling input/output related errors well since buffers are not saved. This would have to be changed in a future version of the system.

Less far-reaching and more immediately realizable extensions would be to expand the system to cover more error conditions resulting in program interruptions; this will in fact become necessary as error conditions are discovered that the program is not designed to handle. In particular, there has been no attempt to include logic covering error conditions related to the improper use of the macros of any access method other than QSAM, and even some of the less commonly used QSAM configu-

rations have not been fully tested. Inclusion of such features would be extremely desirable.

Significant improvements might also be made to the way in which the imprecise interruption types are handled. Through a process of "instruction reversal" it might be possible to analyze each instruction preceding the interrupt location to determine which instruction actually did generate the invalid address causing the interruption, rather than simply noting all the instructions that could have. However this scheme would only work in some cases and an instruction reversing procedure would be extremely difficult to design and implement.

Many other improvements could also be made in the user-system interface. At present this interface is extremely simple but allows the user very few options. Possible enhancements include letting the user specify his own exit routine for handling some interruptions while allowing him to use the error diagnostics system for other error types. In the case of maskable interruption types especially, it might be desirable to let the user specify whether he desires his program to terminate or resume execution after the diagnostic information is printed. Both of these enhancements could be implemented through the use of optional parameters in the SPIESET macro. A reentrant version of the SPIESET macro might also be desirable, allowing the system to be used by a program actually being executed simultaneously by multiple users.

APPENDIX

SAMPLE PROGRAMS

Sample Program # 1

The first sample program attempts to execute a PUT macro referring to an unopened data set. This results in an invalid operation (OC1) interruption. The SPIESET macro is set to trap all fifteen program interruptions. Also included in this example is a listing of all the job control language statements necessary to use the error diagnostics system. Following the diagnostic output are the first three pages of the standard system dump which is optionally produced.

```
//D6532    JOB PHENIX,                                                X JOB  90
//  'XXXXXXXXXXXXXXXX','.  BARRY M. KIRSCH  ',                         05/29/74
//  1000,CLASS=A
//STEP1 EXEC ASMGRUN
XXCMP EXEC PGM=ASMGASM,TIME=(0,10)                                     00000010
//CMP.SYSLIB DD
X/SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR                                   00000020
//          DD DSN=FRA380.KIRSCH,DISP=SHR
XXSYSLIN DD UNIT=SYSDA,SPACE=(CYL,(1,1)),DISP=(MOD,PASS),             *00000030
XX DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)                                 00000040
XXSYSPRINT DD SYSOUT=A                                                 00000050
XXSYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=BLKSIZE=7000              00000060
XXSYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=BLKSIZE=7000              00000070
XXSYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=BLKSIZE=7000              00000080
//CMP.SYSIN  DD *
IEF236I ALLOC. FOR D6532.  CMP      STEP1
IEF237I 332   ALLOCATED TO SYSLIB
IEF237I 433   ALLOCATED TO
IEF237I 435   ALLOCATED TO SYSLIN
IEF237I 186   ALLOCATED TO SYSPRINT
IEF237I 437   ALLOCATED TO SYSUT1
IEF237I 435   ALLOCATED TO SYSUT2
IEF237I 436   ALLOCATED TO SYSUT3
IEF237I 143   ALLOCATED TO SYSIN
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF285I   SYS1.MACLIB                                  KEPT
IEF285I   VOL SER NOS= IRCC82.
IEF285I   FRA380.KIRSCH                                KEPT
IEF285I   VOL SER NOS= IRCC73.
IEF285I   SYS74150.T085135.RV000.D6532.R0000441        PASSED
IEF285I   VOL SER NOS= IRCC75.
IEF285I   SYS74150.T085135.RV000.D6532.R0000443        DELETED
IEF285I   VOL SER NOS= IRCC77.
IEF285I   SYS74150.T085135.RV000.D6532.R0000444        DELETED
IEF285I   VOL SER NOS= IRCC75.
IEF285I   SYS74150.T085135.RV000.D6532.R0000445        DELETED
IEF285I   VOL SER NOS= IRCC76.
OEF501I  SUMMARY FOR STEP  CMP
 ___CPU TIME____      ___STORAGE____      ____________I/O____________      ___MOUNTABLE UNITS___
 |  00:00:02.78  |    |  REQ.  126K  |    | CARD      28  PRINT    128  PUNCH     0  |    |     DISK  0     |
 |_____________|    |__USED__126K__|    |__DISK_____190_____________TAPE_____0__|    |_____TAPE__0_____|
XXGO EXEC PGM=LOADER,TIME=(0,20),COND=(8,LE)                           00000090
XXSYSLIN DD DSN=*.CMP.SYSLIN,DISP=(OLD,DELETE)                         00000100
XXSYSLOUT DD SYSOUT=A                                                  00000110
***                                                                    00000120
//GO.STEPLIB  DD DSN=FRA380.KIRSCH2,DISP=SHR
//GO.SYSPRINT DD SYSOUT=A
//GO.SYSUDUMP DD SYSOUT=A
//
IEF236I ALLOC. FOR D6532    GO      STEP1
IEF237I 435   ALLOCATED TO SYSLIN
IEF237I 186   ALLOCATED TO SYSLOUT
IEF237I 433   ALLOCATED TO STEPLIB
IEF237I 187   ALLOCATED TO SYSPRINT
IEF237I 189   ALLOCATED TO SYSUDUMP
IEW199I ERROR - USER PROGRAM HAS ABNORMALLY TERMINATED
COMPLETION CODE - SYSTEM=0C1  USER=0000
IEF285I   SYS74150.T085135.RV000.D6532.R0000441        DELETED
IEF285I   VOL SER NOS= IRCC75.
IEF285I   FRA380.KIRSCH2                               KEPT
IEF285I   VOL SER NOS= IRCC73.
OEF501I  SUMMARY FOR STEP  GO
 ___CPU TIME____      ___STORAGE____      ____________I/O____________      ___MOUNTABLE UNITS___
 |  00:00:01.47  |    |  REQ.  126K  |    | CARD       0  PRINT    521  PUNCH     0  |    |     DISK  0     |
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    30 MAY 74

000000                              2 SAMPLE1  CSECT
                                    3          SAVE  (14,12),,*   STANDARD CSECT BEGINNING
000000 47F0 F00C            0000C   4+         B     12(0,15) BRANCH AROUND ID
000004 07                           5+         DC    AL1(7)
000005 E2C1D4D7D3C5F1               6+         DC    CL7'SAMPLE1' IDENTIFIER
00000C 90EC D00C            0000C   7+         STM   14,12,12(13) SAVE REGISTERS
000010 05C0                         8          BALR  12,0          REGISTER 12 USED AS BASE REGISTER
000012                              9          USING *,12          ESTABLISH ADDRESSABILITY
000012 4110 C03A            0004C  10          LA    1,SAVEAREA    POINTER TO CURRENT SAVE AREA
000016 501D 0008            00008  11          ST    1,8(13)       FORWARD CHAIN SAVE AREAS
00001A 5001 0004            00004  12          ST    13,4(1)       BACK CHAIN SAVE AREAS
00001E 1801                        13          LR    13,1          REG 13 <- ADDRESS OF CURRENT SAVE AREA
                                   14 *
                                   15 *   USE SPIESET MACRO TO TRAP ALL PROGRAM INTERRUPTIONS
                                   16 *
                                   17          SPIESET  SET,((1,15))
000020 0700                        18+         CNOP  2,4
000022 4110 C01C            0002E  19+         LA    1,*+12 LOAD BRANCH ADDRESS
000026 0511                        20+         BALR  1,1 BRANCH AROUND PARAMS.
000028 0F                          21+         DC    B'00001111' PROGRAM MASK BITS
000029 0000F8                      22+         DC    AL3(SET) EXIT ROUTINE ADDRESS
00002C 7F                          23+         DC    B'01111111'
00002D FF                          24+         DC    B'11111111' INTERUPTION MASK
00002E 0A0E                        25+         SVC   14 ISSUE SPIE SVC


0000F8                             27+SET      CSECT
0000F8                             28+         USING SET,15
0000F8 900F F050            00148  29+         STM   0,15,SPSET
0000FC 47F0 F090            00188  30+         B     STSET
000100                             31+SVSET    DS    18F
000148                             32+SPSET    DS    16F
000188 4100 F008            00100  33+STSET    LA    13,SVSET
                                   34+         DROP  15
000100                             35+         USING SVSET,13
00018C                             36+         CNOP  0,4
00018C 45F0 D0A0            001A0  37+         BAL   15,*+20 LOAD SUP.PARAMLIST ADR
000190 00000198                    38+         DC    A(*+8) ADDR OF EP PARAMETER
000194 00000000                    39+         DC    A(0) DCB ADDRESS PARAMETER        LCOA
000198 C9D5E3C5D9E4D7E3            40+         DC    CL8'INTERUPT' EP PARAMETER
0001A0 0A06                        41+         SVC   6 ISSUE LINK SVC
0001A2 980F D048            00148  42+         LM    0,15,SPSET
                                   43+         DROP  13
0001A6 07FE                        44+         BR    14


000000                             46+SAMPLE1  CSECT
                                   47 *
                                   48 *   ATTEMPT TO EXECUTE A PUT MACRO REFERRING TO AN UNOPENED DATA SET,
                                   49 *     THEREBY CAUSING AN INVALID OPERATION (OC1) INTERRUPTION
                                   50 *
                                   51          PUT   PRINTOUT,SAVEAREA
000030 4110 C082            00094  52+         LA    1,PRINTOUT LOAD PARAMETER REG 1
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    30 MAY 74

000034 4100 C03A            0004C   53+          LA     0,SAVEAREA LOAD PARAMETER REG 0
000038 58F0 1030            00030   54+          L      15,48(0,1) LOAD PUT ROUTINE ADDR.
00003C 05EF                         55+          BALR   14,15 LINK TO PUT ROUTINE
                                    56 *
00003E 58DD 0004            00004   57           L      13,4(13)     REG 13 <- ADDRESS OF PREVIOUS SAVE AREA
                                    58           RETURN (14,12),RC=0
000042 98EC 000C            0000C   59+          LM     14,12,12(13) RESTORE THE REGISTERS
000046 41F0 0000            00000   60+          LA     15,0(0,0) LOAD RETURN CODE
00004A 07FE                         61+          BR     14 RETURN
00004C                              62 SAVEAREA  DS     18F
                                    63 PRINTOUT  DCB    DSORG=PS,MACRF=PM,RECFM=FB,LRECL=132,BLKSIZE=660,      X
                                                        DDNAME=FILEOUT

                                    65+*                        DATA CONTROL BLOCK
                                    66+*
000094                              67+PRINTOUT DC      0F'0' ORIGIN ON WORD BOUNDARY

                                    69+*                        DIRECT ACCESS DEVICE INTERFACE

000094 0000000000000000             71+          DC     BL16'0' FDAD,DVTBL
0000A4 00000000                     72+          DC     A(0) KEYLE,DEVT,TRBAL

                                    74+*                        COMMON ACCESS METHOD INTERFACE

0000A8 00                           76+          DC     AL1(0) BUFNO
0000A9 000001                       77+          DC     AL3(1) BUFCB
0000AC 0000                         78+          DC     AL2(0) BUFL
0000AE 4000                         79+          DC     BL2'0100000000000000' DSORG
0000B0 00000001                     80+          DC     A(1) IOBAD

                                    82+*                        FOUNDATION EXTENSION

000084 00                           84+          DC     BL1'00000000' BFTEK,BFLN,HIARCHY
000085 0000C1                       85+          DC     AL3(1) EODAD
000088 90                           86+          DC     BL1'10010000' RECFM
000089 000000                       87+          DC     AL3(0) EXLST

                                    89+*                        FOUNDATION BLOCK

00008C C6C9C3C5D6E4E340             91+          DC     CL8'FILEOUT' DDNAME
0000C4 02                           92+          DC     BL1'00000010' OFLGS
0000C5 00                           93+          DC     BL1'00000000' IFLG
0000C6 0050                         94+          DC     BL2'0000000001010000' MACR

                                    96+*                        BSAM-BPAM-QSAM INTERFACE

0000C8 00                           98+          DC     BL1'00000000' REK1
0000C9 000001                       99+          DC     AL3(1) CHECK, GERR, PERR
0000CC 00000001                    100+          DC     A(1) SYNAD
0000D0 0000                        101+          DC     H'0' CIND1, CIND2
0000D2 0294                        102+          DC     AL2(660) BLKSIZE
0000D4 00000000                    103+          DC     F'0' WCPD, WCPL, OFFSR, OFFSW
0000D8 00000001                    104+          DC     A(1) IOBA
0000DC 00                          105+          DC     AL1(0) NCP
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                  30 MAY 74

000000D 000001                    106+         DC    AL3(1) EOBR, EOBAO

                                  108+*                            QSAM INTERFACE

000DE0 00000001                   110+         DC    A(1) RECAD
0000E4 0000                       111+         DC    H'0' QSWS
0000E6 0084                       112+         DC    AL2(132) LRECL
0000E8 00                         113+         DC    BL1'00000000' EROPT
0000E9 000001                     114+         DC    AL3(1) CNTRL
0000EC 00000000                   115+         DC    F'0' PRECL
0000F0 00000001                   116+         DC    A(1) EOB
000000                            117          END   SAMPLE1
```

EXECUTION ERROR

SYSTEM COMPLETION CODE = 0C1 INVALID OPERATION

INTERRUPT OCCURRED AT LOCATION ** 000050 ** WHICH IS OUTSIDE THE USER'S PROGRAM AREA

G P REGISTERS AT TIME OF INTERRUPT

REGISTER	HEX	DECIMAL
0	001CD05C	1888348
1	001CD044	1888420
2	801E8F64	-2145468572
3	001CD010	1888272
4	801E8FFE	-2145468418
5	FFFFFFFF	-1
6	001E8F68	2015080
7	000000FF	255
8	00000000	0
9	001E8E88	2014904
10	001E8FF8	2015224
11	001CD000	1888256
12	401CD022	1075630114
13	001CD05C	1888348
14	4F1CD04E	1327288398
15	02000050	33554512

FLOATING POINT REGISTERS AT TIME OF INTERRUPT

REGISTER	HEX	DECIMAL
0	FF000008 C0000001	-3.450873173495716E+69
2	801CD600 402D6402	-9.728885134549204E-79
4	011EC5E0 13015E50	1.661002413584987E-77
6	001EC7C8 901CD640	1.038377960862321E-78

HEX CONTENTS OF INTERRUPT LOCATION = 0000 4A00 0000 A5E8

********** ERROR DIAGNOSIS **********

THE OPERATION CODE IN THE INSTRUCTION CAUSING THE INTERRUPT IS INVALID - NO SUCH OPERATION EXISTS ON THIS COMPUTER

THE CSECT BEING EXECUTED AT TIME OF INTERRUPT WAS ** SAMPLE1 **

THE MOST PROBABLE CAUSE OF THIS ERROR IS AN ATTEMPT TO EXECUTE A PUT MACRO REFERRING TO AN UNOPENED OR IMPROPERLY OPENED DATA SET

THE DDNAME OF THE DATA SET IS ** FILEOUT **

THE LOCATION OF ITS DATA CONTROL BLOCK (DCB) IS
** 1CD0A4 ** (ABSOLUTE) ** 000094 ** (RELATIVE)

THE LOCATION OF THE NEXT PROGRAM INSTRUCTION TO BE EXECUTED IS
** 1CD04E ** (ABSOLUTE) ** 00003E ** (RELATIVE)

****

A STANDARD OS/MVT SYSTEM DUMP WILL FOLLOW IF A //SYSUDUMP DD CARD IS INCLUDED IN THE JCL STREAM

```
JOB 06532            STEP GO              TIME 091029   DATE 74150                                          PAGE 0001

COMPLETION CODE      SYSTEM = 0C1

PSW AT ENTRY TO ABEND  FFA5000D 40000052

TCB  0108C8  RGP   0001EAF0   PIE     00000000   DEB  00000000   TIO 00021E80   CMP   800C1000   TRN  00000000
             MSS   0002F488   PK-FLG  A0850000   FLG  00001F18   LLS 0002EE10   JLB   0002EB38   JPQ  00000000
             FSA   011E9E70   TCB     0001E600   TME  00000000   JST 0001E7A0   NTC   00000000   DTC  0001E7A0
             LTC   00000000   IQE     00000000   ECB  001EBF34   STA 20000000   D-PQE 0002F780   SQS  0001DF80
             NSTAE 00000000   TCT     8001EE78   USER 00000000   DAR 00000000   RESV  00000000   JSCB 8702E5D0


ACTIVE RBS

PRB  02E090  RESV   00000000   APSW   40000052   WC-SZ-STAB 00040082   FL-CDE 0002E670   PSW FFA5000D 40000052
             Q/TTR  00000000   WT-LNK 000108C8

SVRB 01F668  TAB-LN 00180220   APSW   59F0F1C3   WC-SZ-STAB 00120002   TQN    00000000   PSW 00040033 500093F2
             Q/TTR  00004628   WT-LNK 0002E090
             RG 0-7   001CD05C   001CD0A4   801EBF64   001CD010   801EBFFE   FFFFFFFF   001EBF6B   000000FF
             RG 8-15  00000000   001EBEB8   001CBFF8   001CD000   401CD022   001CD05C   4F1CD04E   02D00050
             EXTSA    000021BE   8F1EBE18   00000000   00000000   FF030000   0001F6E4   0001F6EC   E2E8E2C9
                      C5C1F0F1   C9C5C1C8   C1C2C5D5   C4F90C10

SVRB 01EAF0  TAB-LN 000803C8   APSW   F1F0F5C1   WC-SZ-STAB 0012D002   TQN    00000000   PSW FF04000C 401EB7A6
             Q/TTR  00004825   WT-LNK 0001F668
             RG 0-7   00000007   0001F6C8   8000930A   0000A368   0001DBC8   0001F668   0401E7A0   0001F668
             RG 8-15  0001E7A0   40005252   0001E7A0   8F1EBE18   00021EEC   0001F6EC   40009CEC   00000000
             EXTSA    E2E8E2C9   C5C1F0F1   00000000   00000085   C109C102   C303C3C6   C3C20000   00000000
                      00000000   00000000   00000000   00000000


LOAD LIST

      NE 00000000   RSP-CDE 02020C08


CDE

    02E670      ATR1 09   NCDE 000000   ROC-RB 0002E090   NM **GO       USE 01   EPA 1CD010   ATR2 A0   XL/MJ 02E688
    02DC08      ATR1 30   NCDE 02AB20   ROC-RB 00000000   NM IGC0A05A   USE 02   EPA 1EBD58   ATR2 28   XL/MJ 02E9C0

XL                                                LN          ADR          LN          ADR          LN          ADR

    02E688   SZ 00000010   NQ 00000001      80000188    001CD000
    02E9C0   SZ 00000010   NQ 00000001      800007A8    001E8058

TIOT   JOB  06532      STEP GO                PROC  STEP1
    DD            14040100    SYSLIN     00380B00    80002D08
    DD            14040100    SYSLOUT    00380F00    80002070
    DD            14040140    STEPLIB    00381100    80002C88
```

```
     DD            14040100    SYSPRINT    00381300    80002090
     DD            14040140    SYSUDUMP    00381500    80002000

MSS            ************ SPQE ************     *************** DQE ***************     ******* FQE ********
               FLGS  NSPQE      SPID      DQE        BLK       FQE       LN       NDQE        NFQE        LN

     02F488    80    02F588     000       02A038
     02A038    60    000000     000       02F310     001EB800 001EB800 00000800 0002EF40    00000000    00000500
                                                     001CD000 001CD188 00000800 00000000    00000000    00000100
     02F588    C0    000000     078       02F588
     02F588    20    02F608     078       CC0000

D-PQE  0002F780   FIRST 0002EC80   LAST 0002EC80
PQE  02EC80   FFB 001CF800   LFB 001CF800   NPQ 00000000   PPQ 00000000
              TCB 00026F00   RSI 0001F800   RAO 001C0000   FLG 0000

FBCE 1CF800   NFB 0002EC80   PFB 0002EC80,  SZ 00018800

QCB TRACE

MAJ 02DEB8   NMAJ 00023038   PMAJ 00016798   FMIN 0002DE50   NM  SYSDSN

MIN 0248F0   FQEL 000248E0   PMIN 00024910   NMIN 000248C8   NM FF  SYS1.MACLIB

             NQEL 00000000   PQEL 8002260    TCB  00026FD0   SVRB 0001EAF0

MIN 02F030   FQEL 0002F618   PMIN 0002F0D8   NMIN 0002E698   NM FF  FRA380.KIRSCH

             NQEL 00000000   PQEL 8002F030   TCB  00026FD0   SVRB 0001EAF0

MIN 02E698   FQEL 0002F108   PMIN 0002F030   NMIN 00000000   NM FF  FRA380.KIRSCH2

             NQEL 00000000   PQEL 8002E698   TCB  00026FD0   SVRB 0001EAF0

MAJ 02AA50   NMAJ 0002F088   PMAJ 00023038   FMIN 00029CA8   NM  SYSIEA01

MIN 029CA8   FQEL 0002AC00   PMIN 0002AA50   NMIN 00000000   NM AO  IEA

             NQEL 00000000   PQEL 00029CA8   TCB  0001D8C8   SVRB 0001EAF0

SAVE AREA TRACE

**GO     WAS ENTERED VIA LINK          AT EP SAMPLE1

SA   1EBE70  WD1 00000000   HSA 00000000   LSA C01CD05C   RET 00014824   EPA 011CD010   R0  FF000008
             R1  001EBF60   R2  801EBF64   R3  C01CD010   R4  801EBFFE   R5  FFFFFFFF   R6  001EBF68
             R7  000000FF   R8  00000000   R9  001EBEB8   R10 001EBFF8   R11 001C0000   R12 402FA12C

SA   1CD05C  WD1 C54C5850   HSA 001EBE70   LSA 506C4130   RET 30001233   EPA 4780C056   R0  D2025031
             R1  50604120   R2  20041255   R3  4720C03E   R4  4510C08C   R5  101CDBF8   R6  001CDC68
             R7  001CD9C8   R8  001CDB18   R9  001C0B88   R10 301C0AA8   R11 301CDA38   R12 101CD878

INTERRUPT AT 000052
```

```
PROCEEDING BACK VIA REG 13

SA   1CD05C  WD1 C54C5850   MSA 001EBE70   LSA 506C4130   RET 30001233   EPA 4780C056   R0  D2025031
             R1  50604120   R2  20041255   R3  4720C03E   R4  4510C08C   R5  101C0BF8   R6  001C0C68
             R7  001CC9C8   R8  001CDB18   R9  001CDB88   R10 301CDAAB   R11 301CDA38   R12 101CD878

**GO     WAS ENTERED VIA LINK          AT EP SAMPLE1

SA   1EBE70  WD1 00000000   MSA 00000000   LSA 001CD05C   RET 00014824   EPA 011CD010   R0  FF000008
             R1  001EBF60   R2  801EBF64   R3  001CD010   R4  801EBFFE   R5  FFFFFFFF   R6  001EBF68
             R7  000000FF   R8  00000000   R9  001EBEB8   R10 001EBFF8   R11 001CD000   R12 402FA12C

REGS AT ENTRY TO ABEND

     FLTR 0-6      FF00000800000001      801CD60040206402              011EC5E013015E50      001EC7C8901CD640

     REGS 0-7      001CD05C   001CD0A4   801EBF64   001CD010           801EBFFE   FFFFFFFF   001EBF68   000000FF
     REGS 8-15     00000000   001EBEB8   001EBFF8   001CD000           401CD022   001CD05C   4F1C004E   02000050

LOAD MODULE    **GO

1CD000   5C5CC7D6 40404040 00000000 00000000    47F0F00C 07E2C1D4 D7D3C5F1 9CECD00C   *..GO     .........00..SAMPLE1....*
1CD020   05C04110 C0345010 00085001 00041801    07004110 C01C0511 0F1CD108 7FFF0A0E   *...........J...J.........J......*
1CD040   4110C032 4100C03A 58F01030 05EF580D    000498EC D00C41F0 000D07FF C54C5850   *.........0..............0....E...*
1CD060   001EBE70 506C4130 30001233 4780C056    D2025031 50604120 20041255 4720C03E   *...................K..............*
1CD080   4510C08C 101CD3F8 001CDC68 001CD9C8    001CDB18 001CDB88 301CDAAB 301CDA38   *........8......RH..............*
1CD0A0   101CD378 00000000 00000000 00000000    00000000 00000000 00000001 00004000   *..O........................ .*
1CD0C0   00000001 00000001 90000000 C6C9D3C5    D6F4E340 02000050 00000001 00000001   *............FILEOUT ............*
1CD0E0   00000294 00000000 00000001 00000001    00000C01 00000084 00000001 00000000   *................................*
1CD100   00000001 58F0F630 900FF050 47F0F090    07FF9101 10170712 001EBC38 00014824   *.....06...0..00................*
1CD120   011CD980 FFFE141C 001EBE50 801EBF64    001CD010 801EBFFE FFFFFFFF 001EBF68   *................................*
1CD140   000000FF 00000000 001CBEB8 001EBFF8    001CD000 401CD022 FFFE141C 001EBE50   *................8.... ............*
1CD160   801EBF64 001CD010 801EBFFE FFFFFFFF    001EBF68 000000FF 00000000 0C1CDEB8   *................................*
1CD180   001EBFF8 001CD000 401CD022 001CD05C    0000B38C 0F1CD108 41D0F008 45F0D0A0   *...8.... ..............J...0..0..*
1CD1A0   001CD1A8 00000000 C9D5E3C5 D9E4D7E3    0A06980F D04807FE                     *..J.....INTERUPT...............*

LOAD MODULE    IGC0A05A

1EB040                                                                 41800059 18114313   *DDUC. PR-EXT....S-LTLES ........*
1EB060   00001A31 41330001 95FF3000 47806068    18EC18FF 18001811 43E30000 43030001   *.........................T.......*
1EB080   8C000004 83F0001C 8C000004 8810001C    1A2044E0 60701A1E 41818001 44F06076   *.....0..........................0..*
1EB0A0   F3040069 D0690C07 D06962C6 41FFF001    44F0607C 47F066F8 413E3003 47F06010   *3...........F..0..0...0.8......0..*
1EB0C0   41330001 47F069E2 D2000C00 30020200    D0692000 D20C8000 D0695050 D08C5000   *.....0.SK......K......K...........*
1EB0E0   D12094FC D1235800 D1201A10 58000120    41110003 50100064 94FC0067 D703006C   *J...J...J......J..............P...*
1EB100   D06C1810 54006290 19014780 608C1810    4010D06C 58100064 4A10006C 18005000   *................ ................*
1EB120   66804000 D05A1211 47706110 4810006A    12114770 60E85850 D08C5860 D12407F5   *.. ......................Y......J..5*
1EB140   41200121 45806236 58200120 413062C4    48A0D06A 48A0006C 88A00002 4580623E   *..J........J.....0...............*
1EB160   46A06104 9640D112 4550620A 940FD112    47F0600E 18A15810 D1204810 D06C5010   *..... J.......J..0......J........*
1EB180   00704120 D0714580 62364810 D06C8810    00011A31 58200120 4580623E 50200120   *.......................J........J..*
1EB1A0   96400112 4550620A 940FD112 4810D06C    12114770 61984700 615E4110 0001191A   *. J.......J......................*
1EB1C0   47B061A6 18125810 6680D51F 10002000    477061A6 4810006E 411100C1 4010006E   *...........N.................. ...*
1EB1E0   41220020 50200120 46A0615E 47F06180    18114010 D06C46A0 611E47F0 60D44810   *......J.......0.... .........0.M..*
```

Sample Program # 2

The second sample program generates an imprecise protection (0C4) interruption by attempting a store into absolute location zero. The eleven instructions preceding and including the interrupt location are decoded and printed, with only the one instruction actually causing the interruption being flagged. The first reconstructed instruction SU 5,2574(9,0) does not correspond to an actual program instruction, but is a valid reconstruction of data flags expanded as part of the SPIESET macro. There is no way to programmatically avoid this. The succeeding ten instructions do correspond to actual program instructions. Following the diagnostic output are the first three pages of the slightly modified system dump produced at the programmer's option. In this example, the SPIESET macro is set to trap only the non-maskable program interruptions. All the maskable interruption types are disabled.

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    30 MAY 74

000000                                2 SAMPLE2  CSECT
                                      3          SAVE  (14,12),,*   STANDARD CSECT BEGINNING
000000 47F0 F00C            0000C     4+         B     12(0,15) BRANCH AROUND ID
000004 07                             5+         DC    AL1(7)
000005 E2C1D4D7D3C5F2                 6+         DC    CL7'SAMPLE2' IDENTIFIER
00000C 90EC D00C            0000C     7+         STM   14,12,12(13) SAVE REGISTERS
000010 05C0                           8          BALR  12,0          REGISTER 12 USED AS BASE REGISTER
000012                                9          USING *,12          ESTABLISH ADDRESSABILITY
000012 4110 C046            00058    10          LA    1,SAVEAREA    POINTER TO CURRENT SAVE AREA
000016 5010 0008            00008    11          ST    1,8(13)       FORWARD CHAIN SAVE AREAS
00001A 5001 0004            00004    12          ST    13,4(1)       BACK CHAIN SAVE AREAS
00001E 18D1                          13          LR    13,1          REG 13 <- ADDRESS OF CURRENT SAVE AREA
                                     14 *
                                     15 *   USE SPIESET MACRO TO TRAP ALL NON-MASKABLE PROGRAM INTERRUPTIONS
                                     16 *
                                     17          SPIESET  SET,((1,7),9,(11,12),15)
000020 0700                          18+         CNOP  2,4
000022 4110 C01C            0002E    19+         LA    1,*+12 LOAD BRANCH ADDRESS
000026 0511                          20+         BALR  1,1 BRANCH AROUND PARAMS.
000028 00                            21+         DC    B'00000000' PROGRAM MASK BITS
000029 0000C8                        22+         DC    AL3(SET) EXIT ROUTINE ADDRESS
00002C 7F                            23+         DC    B'01111111'
00002D 59                            24+         DC    B'01011001' INTERUPTION MASK
00002E 0A0E                          25+         SVC   14 ISSUE SPIE SVC


000C8                                27+SET      CSECT
000C8                                28+         USING SET,15
000C8 900F F050             00118    29+         STM   0,15,SPSET
000CC 47F0 F090             00158    30+         B     STSET
000D0                                31+SVSET    DS    18F
00118                                32+SPSET    DS    16F
00158 41D0 F008             00D0     33+STSET    LA    13,SVSET
                                     34+         DROP  15
00D0D                                35+         USING SVSET,13
00015C                               36+         CNOP  0,4
00015C 45F0 D0A0            00170    37+         BAL   15,*+20 LOAD SUP.PARAMLIST ADR
00160 00000168                       38+         DC    A(*+8) ADDR OF EP PARAMETER
00164 00000000                       39+         DC    A(0) DCB ADDRESS PARAMETER        LCDA
00168 C9D5E3C5D9E4D7E3               40+         DC    CL8'INTERUPT' EP PARAMETER
00170 0A06                           41+         SVC   6 ISSUE LINK SVC
00172 98CF D048             00118    42+         LM    0,15,SPSET
                                     43+         DROP  13
00176 07FE                           44+         BR    14


000000                               46+SAMPLE2  CSECT
                                     47 *
                                     48 *   CAUSE A PROTECTION (OC4) INTERRUPTION
                                     49 *
000030 9836 C08E            000A0    50          LM    3,6,=4F'1'
000034 0700                          51          NOPR  0
000036 5060 0000            00000    52          ST    6,0(0)        **** CAUSE PROTECTION INTERRUPT ****
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

00003A 0700                        53          NOPR  0
00003C 1436                        54          NR    3,6
00003E 5830 C0AE          000C0    55          S     3,=F'2'
000042 5860 0000          00000    56          L     6,0(0)
000046 9836 C09E          00080    57          LM    3,6,=4F'3'
                                   58 *
00004A 580D 0004          00004    59          L     13,4(13)     REG 13 <- ADDRESS OF PREVIOUS SAVE AREA
                                   60          RETURN (14,12),RC=0
00004E 98EC 000C          0000C    61+         LM    14,12,12(13) RESTORE THE REGISTERS
000052 41F0 0000          00000    62+         LA    15,0(0,0) LOAD RETURN CODE
000056 07FE                        63+         BR    14 RETURN
000058                             64 SAVEAREA DS    18F
000000                             65          END   SAMPLE2
0000A0 0000000100000001            66                =4F'1'
0000B0 0000000300000003            67                =4F'3'
0000C0 00000002                    68                =F'2'
```

EXECUTION ERROR

SYSTEM COMPLETION CODE = 0C4 PROTECTION INTERRUPT

INTERRUPT OCCURRED AT LOCATION ** 1CD05E ** WHICH IS ** 00004E (HEX) ** BYTES INTO ROUTINE ** SAMPLE2 **

G P REGISTERS AT TIME OF INTERRUPT

REGISTER	HEX	DECIMAL
0	FA000020	-100663264
1	00000000	0
2	801E3F64	-2145468572
3	00000003	3
4	00000003	3
5	00000003	3
6	00000003	3
7	000000FF	255
8	00000000	0
9	001E0E88	2014904
10	001E0FF8	2015224
11	001C0000	1888256
12	401C0022	1073630114
13	001C0068	1888360
14	00014824	84004
15	011CD010	18665488

FLOATING POINT REGISTERS AT TIME OF INTERRUPT

REGISTER	HEX	DECIMAL
0	FF000008 00000001	-3.450873173495716E+69
2	801CD6D0 402D6402	-9.728875134549204E-79
4	011EC5E0 13015330	1.661002413584987E-77
6	001EC7C8 901CD640	1.038377960862321E-78

********** ERROR DIAGNOSIS **********

THE USER PROGRAM HAS ATTEMPTED TO CHANGE THE CONTENTS OF A PROTECTED AREA OF MAIN STORAGE

POSSIBLE CAUSES OF THIS CONDITION ARE AN UNINITIALIZED INDEX OR AN ATTEMPT TO INDEX OUTSIDE THE PROGRAM'S ASSIGNED LIMITS
THIS MIGHT RESULT FROM THE CONTENTS OF THE BASE REGISTER HAVING BEEN DESTROYED

STARTING 34 BYTES BACK FROM THE INTERRUPT LOCATION ARE THE FOLLOWING PROGRAM INSTRUCTIONS :
(THOSE INSTRUCTIONS THAT COULD HAVE POSSIBLY CAUSED THIS INTERRUPTION ARE FLAGGED AT THE RIGHT)

ABSOLUTE LOCATION	RELATIVE LOCATION	OBJECT CODE	RECONSTRUCTED INSTRUCTION		INSTRUCTION FLAG
1CD03C	00002C	7F59 0A0E	SU	5,2574(9, 0)	
1CD040	000030	9836 C08E	LM	3, 6, 142(12)	
1CD044	000034	0700	BCR	0, 0	
1CD046	000036	5060 0000	ST	6, 0(0, 0)	****
1CD04A	00003A	0700	BCR	0, 0	
1CD04C	00003C	1436	NR	3, 6	
1CD04E	00003E	5B30 C0AE	S	3, 174(0,12)	
1CD052	000042	5860 0000	L	6, 0(0, 0)	
1CD056	000046	9836 C09E	LM	3, 6, 158(12)	
1CD05A	00004A	5800 0004	L	13, 4(13, 0)	
1CD05E	00004E	98EC D00C	LM	14,12, 12(13)	

****

A STANDARD OS/MVT SYSTEM DUMP WILL FOLLOW IF A //SYSUDUMP DD CARD IS INCLUDED IN THE JCL STREAM
THIS DUMP SHOWS CONDITIONS CLOSELY APPROXIMATING THOSE AT TIME OF INTERRUPT

```
JOB 06533          STEP GO          TIME 091053   DATE 74150                                PAGE 0001

COMPLETION CODE     SYSTEM = 0C4

PSW AT ENTRY TO ABEND  FFA50000 401CD060

TCB  01DBC8  RBP    00C1F668   PIE     00000000   DEB  00000000   TIO 0001E868   CMP   800C4000   TRN  00000000
             MSS    00C2F0F8   PK-FLG  A0850000   FLG  00001F18   LLS 0002EE10   JLB   00022C18   JPQ  00000000
             FSA    011EBE70   TCB     0001EF88   TME  00000000   JST 0001F928   NTC   00000000   OTC  0001F928
             LTC    00000000   IQE     00000000   ECB  001EBF34   STA 20000000   D-PQE 0002F780   SQS  0001D700
             NSTAE  00000000   TCT     80021E80   USER 00000000   DAR 00000000   RESV  00000000   JSCB 8702EAE0


ACTIVE RBS

PRB  02E900  RESV   00000000   APSW    00000000   WC-SZ-STAB 00040082   FL-CDE 00029C98   PSW FFA50000 401CD060
             Q/TPR  00000000   WT-LNK  00010BC8

SVRB 021188  TAB-LN 00180220   APSW    F9F0F1C3   WC-SZ-STAB 0012D002   TCN    00000000   PSW 00040033 500093F2
             Q/TPR  00004628   WT-LNK  0002E900
             RG 0-7    FAC00020   80CC40C0   801EBF64   0C000003   00000003   00000003   00000003   000000FF
             RG 8-15   00000000   001EBE88   001EBFF8   001CD000   401CD022   001CD068   00014824   011CD010
             EXTSA     0000218E   8F1EBE18   00000000   00000000   FF030000   00021234   0002123C   E2E8E2C9
                       C5C1F0F1   C9C5C1C8   C1C2C5D5   C4F90C40

SVRB 01F668  TAB-LN C02803C8   APSW    F1F0F5C1   WC-SZ-STAB 0012D002   TCN    00000000   PSW FF04000C 401E87A6
             Q/TPR  00004825   WT-LNK  00021188
             RG 0-7    00C00007   00021218   80C0930A   0000A368   0001DBC8   00021188   0401F928   00021188
             RG 8-15   0001F928   40009252   0001F928   8F1EBE18   0001E804   0002123C   400098A4   00000000
             EXTSA     E2E8E2C9   C5C1F0F1   C9C7C3F0   E3F0F3C5   002C679E   A8060000   F1F1004D   5000E6F8
                       F0E2004F   08015E50   00000000   00000000


LOAD LIST

        NE 00000000   RSP-CDE 0202E3E8


CDE

     029C98       ATR1 09   NCDE 000000   ROC-RB 0002E900   NM **GO       USE 01   EPA 1CD010   ATR2 A0   XL/MJ 029CB0
     02E3E8       ATR1 30   NCDE 023A38   ROC-RB 00000000   NM IGC0A05A   USE 02   EPA 1EB058   ATR2 28   XL/MJ 02EFE0


XL                                                LN          ADR          LN          ADR          LN          ADR

     029CB0   SZ 00000010   NO 00000001       80000188    001CD000
     02EFE0   SZ 00000010   NO 00000001       800097A8    001EB058


TIOT   JOB  06533      STEP GO                PROC  STEP1
     DD            14040100    SYSLIN     00440E00    80002D08
     DD            14040100    SYSLOUT    00441200    80002070
     DD            14040140    STEPLIB    00441400    80002C88
```

```
     00          14040100   SYSPRINT   00441600   80002090
     00          14040140   SYSUDUMP   00441800   80002000

MSS            ************ SPQE ************   *************** DQE ***************   ******* FQE ********
               FLGS  NSPQE    SPID    DCE         BLK      FQE      LN       NDQE        NFQE       LN

     02F0F8    80    02F570   000     02AC38
     02AD38    60    000000   000     02F4E8      001E8800 001E8800 00000800 0002E4F0    00000000   00000500
                                                 001CD000 001CD188 00000800 00000000    00000000   00000130
     02F570    C0    C00000   078     02F578
     02F578    20    02F5A0   078     000000

D-PQE  0002F780   FIRST 0002EB50   LAST 0002EB50
PQE  02EB50   FFB 001CE800   LFB 00100800   NPQ 00000000   PPQ 00000000
              TCB 00026FD0   RSI 0001F800   RAD 001CD000   FLG 0000

FBQE 1CE800   NFB 00100800   PFB 0002EB50   SZ 00000800

FBQE 100800   NFB 0002EB50   PFB 001CE800   SZ 0001A800

QCB TRACE

MAJ 020E88   NMAJ 00023038   PMAJ 00016798   FMIN 0002DE50   NM  SYSDSN

MIN 0248F0   FQEL 000248E0   PMIN 00024910   NMIN 000248C8   NM FF  SYS1.MACLIB

             NQEL 00000000   PQEL 800226C0   TCB  00026FD0   SVRB 0001EE70

MIN 02ED10   FQEL 0002F090   PMIN 0002F288   NMIN 0002ECB0   NM FF  FRA380.KIRSCH

             NQEL 00000000   PQEL 8002ED10   TCB  00026FD0   SVRB 0001EE70

MIN 02ECB0   FQEL 0002EFD0   PMIN 0002ED10   NMIN 0002F0A0   NM FF  FRA380.KIRSCH2

             NQEL 00000000   PQEL 8002ECB0   TCB  00026FD0   SVRB 0001EE70

MAJ 02AC08   NMAJ 0002F620   PMAJ 0002F0E0   FMIN 00024008   NM  SYSIEA01

MIN 024008   FQEL 0002E298   PMIN 0002AC08   NMIN 00000000   NM A0  IEA

             NQEL 00000000   PQEL 00024008   TCB  0001DB08   SVRB 0001F668

SAVE AREA TRACE

**GO     WAS ENTERED VIA LINK          AT EP SAMPLE2

SA   1EBE70  WD1 00000000   HSA 00000000   LSA 001CD068   RET 00014824   EPA 011CD010   R0  FF000008
             R1  001EBF60   R2  801EBF64   R3  001CDC[illegible]   R4  801EBFFE   R5  FFFFFFFF   R6  001EBF68
             R7  000000FF   R8  00000000   R9  001EBE88   R10 001EBFF8   R11 001CD000   R12 402FA12C

SA   1CD068  WD1 30001233   HSA 001EBE70   LSA D2025031   RET 506D4120   EPA 20041255   R0  4720C03E
             R1  4510C08C   R2  1C1CDBF8   R3  001CDC68   R4  001CD9C8   R5  001CDB18   R6  001CDB88
```

```
          R7  301CDAA8   R8  301CDA38   R9  101CD878   R10 101CD8E8   R11 901CD958   R12 0A14D502

INTERRUPT AT 1CD060

PROCEEDING BACK VIA REG 13

SA   1CD068  WD1 30001233   HSA 001EBE70   LSA D2025031   RET 506D4120   EPA 20041255   R0  4720C03E
             R1  4510C08C   R2  101CDBF8   R3  001CDC68   R4  001CD9C8   R5  001CDB18   R6  001CDB88
             R7  301CDAA8   R8  301CDA38   R9  101CD878   R10 101CD8E8   R11 901CD958   R12 0A14D502

**GO     WAS ENTERED VIA LINK          AT EP SAMPLE2

SA   1EBE70  WD1 00000000   HSA 00000000   LSA 001CD068   RET 00014824   EPA 011CD010   R0  FF000008
             R1  001EBF6C   R2  801EBF64   R3  001CD010   R4  801EBFFE   R5  FFFFFFFF   R6  001EBF68
             R7  000000FF   R8  00000000   R9  001EBEB8   R10 001EBFF8   R11 001CD000   R12 402FA12C

REGS AT ENTRY TO ABEND

     FLTR 0-6      FF000C0800000001      801CD6D0402D6402              011EC5E013015E50      001EC7C8901CD640

     REGS 0-7      FA000020   800C4000   801EBF64   00000003           00000003   00000003   00000003   000000FF
     REGS 8-15     00000000   001EBEB8   001EBFF8   001CD000           401CD022   001CD068   00014824   011CD010

LOAD MODULE    **GO

1CD000   5C5CC7D6 40404040 00000000 00000000    47F0FC0C 07E2C1D4 D703C5F2 9CEC000C   *..GO     .........00..SAMPLE2....*
1CD020   05C04110 C0465D1D 000B5D01 000418D1    07004110 C01C0511 001CD0D8 7F590A0E   *............J...J...........Q....*
1CD040   9836CC8E 07005060 00000700 14365830    C0AE5860 00009836 C09E580D 00040A0D   *................................*
1CD060   000C41F0 000007FE 30001233 001EBE70    D2025031 50604120 20041255 4720C03E   *...0............K...............*
1CD080   4510C08C 101CD0F8 001CDC68 001CD9C8    001CDB18 001CDB88 301CDAA8 301CDA38   *........8......RH................*
1CD0A0   101CD878 101CD8E8 901CD958 0A14D502    00000001 00000001 00000001 00000001   *..Q...QY..R...N.................*
1CD0C0   00000003 00000003 00000003 00000003    00000002 4520C0F2 9CCFF050 47F0F090   *.......................2..0..00.*
1CD0E0   411DCC48 452CC0F2 001EBC38 00014824    011CDBB0 F590A0E0 001EBE50 801EBF64   *.......2............5...........*
1CD100   00000003 00000003 00000003 00000003    000000FF 00000000 001EBEB8 001EBFF8   *...............................8*
1CD120   001CD000 401CD022 F590A0E0 001EBE50    801EBF64 00000003 00000003 00000003   *.... ...5......................*
1CD140   00000003 000000FF 00000000 001EBEB8    001EBFF8 001CD000 401CD022 001CD068   *...................8.... .......*
1CD160   0000838C 001CD0C8 41D0F008 45F000A0    001CD178 00000000 C9D5E3C5 D9E4D7E3   *.......Q..0..0....J.....INTERUPT*
1CD180   0A06980F D04807FE                                                            *.................30..0...3Y....*

LOAD MODULE    IGC0A05A

1EB040                                                                 41800099 18114313   *ODUC. PR.EXT....S.LTLES ........*
1EB060   00001A81 413300C1 95FF3000 47806068    18EE18FF 18001811 43E30000 43030001   *........................T.......*
1EB080   8CF00004 BRF0001C 8C000004 8810001C    1A2044E0 60701A1E 41818001 44F06076   *.....0.......................0..*
1EB0A0   F3940059 D069DC07 D06962C6 41FFF001    44F0607C 47F066F8 413E3003 47F06010   *3..........F..0..0...0.8.....0..*
1EB0C0   41330001 47F063E2 D20030C0 30C2D200    D0692000 D20C0000 D0695050 D08C5000   *.....0.SK.....K.....K...........*
1EB0E0   012094FC 01235500 01201A10 5800D120    41110003 50100064 94FCD067 D703D06C   *J...J...J.....J............P...*
1EB100   D06C1810 54006290 19014780 60BC1810    4010D06C 5810D064 4A10D06C 18005000   *.............. ..............*
1EB120   66804000 D06A1211 4770611C 4810006A    12114770 6CE85850 D08C5060 012407F5   *.. ................Y......J..5*
1EB140   4120D121 45806236 5820D120 413062C4    48A0D06A 48A0D06C 88A00002 4580623E   *..J.......J....D...............*
1EB160   46A06104 96400112 4550620A 948FD112    47F060DE 18A15810 D1204810 D06C5010   *..... J.......J..0......J.......*
1EB180   D0704120 D07145B0 62364810 D06C8810    60011A31 5820D120 45B0623E 5020D120   *......................J.......J.*
```

Sample Program # 3

The third sample program generates a fixed-point overflow (0C8) interruption during an Add Register operation. The SPIESET macro is set to trap only 0C8 error types, thus enabling the interruption.

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                          20 MAY 74

000000                              2 SAMPLE3  CSECT
                                    3          SAVE  (14,12),,*   STANDARD CSECT BEGINNING
000000 47F0 F00C          0000C     4+         B     12(0,15) BRANCH AROUND ID
000004 07                           5+         DC    AL1(7)
000005 E2C1D40703C5F3               6+         DC    CL7'SAMPLE3' IDENTIFIER
00000C 90EC 000C          0000C     7+         STM   14,12,12(13) SAVE REGISTERS
000010 05C0                         8          BALR  12,0         REGISTER 12 USED AS BASE REGISTER
000012                              9          USING *,12         ESTABLISH ADDRESSABILITY
000012 4110 C032          00044    10          LA    1,SAVEAREA   POINTER TO CURRENT SAVE AREA
000016 5010 0008          00008    11          ST    1,8(13)      FORWARD CHAIN SAVE AREAS
00001A 5001 0004          00004    12          ST    13,4(1)      BACK CHAIN SAVE AREAS
00001E 18D1                        13          LR    13,1         REG 13 <- ADDRESS OF CURRENT SAVE AREA
                                   14 *
                                   15 *   USE SPIESET MACRO TO TRAP OC8 PROGRAM INTERRUPTIONS
                                   16 *
                                   17          SPIESET  SET,(8)
000020 0700                        18+         CNOP  2,4
000022 4110 C01C          0002E    19+         LA    1,*+12 LOAD BRANCH ADDRESS
000026 0511                        20+         BALR  1,1 BRANCH AROUND PARAMS.
000028 C8                          21+         DC    B'00001000' PROGRAM MASK BITS
000029 000098                      22+         DC    AL3(SET) EXIT ROUTINE ADDRESS
00002C 00                          23+         DC    B'00000000'
00002D 80                          24+         DC    B'10000000' INTERUPTION MASK
00002E 0A0E                        25+         SVC   14 ISSUE SPIE SVC


000098                             27+SET      CSECT
000098                             28+         USING SET,15
000098 900F F050          000E8    29+         STM   0,15,SPSET
00009C 47F0 F090          00128    30+         B     STSET
0000A0                             31+SVSET    DS    18F
0000E8                             32+SPSET    DS    16F
000128 41D0 F008          000A0    33+STSET    LA    13,SVSET
                                   34+         DROP  15
0000A0                             35+         USING SVSET,13
00012C                             36+         CNOP  0,4
00012C 45F0 00A0          00140    37+         BAL   15,*+20 LOAD SUP.PARAMLIST ADR
000130 00000138                    38+         DC    A(*+8) ADDR OF EP PARAMETER
000134 00000000                    39+         DC    A(0) DCB ADDRESS PARAMETER       LCOA
000138 C9D5E3C5D9E4D7E3            40+         DC    CL8'INTERUPT' EP PARAMETER
000140 0A06                        41+         SVC   6 ISSUE LINK SVC
000142 980F 0048          000E8    42+         LM    0,15,SPSET
                                   43+         DROP  13
000146 07FE                        44+         BR    14


000000                             46+SAMPLE3  CSECT
                                   47 *
                                   48 *   CAUSE A FIXED-POINT OVERFLOW (OC8) INTERRUPTION
                                   49 *
000030 9889 C07E          00090    50          LM    8,9,=2X'7FFFFFFF'
000034 1A89                        51          AR    8,9
                                   52 *
```

SAMPLE PROGRAM # 3 - ISSUES SPIESET AND THEN CAUSES OC8 INTERRUPT

30 MAY 74

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
000036 58DD 0004            00004   53          L     13,4(13)      REG 13 <- ADDRESS OF PREVIOUS SAVE AREA
                                    54          RETURN (14,12),RC=0
00003A 98EC 000C            0000C   55+         LM    14,12,12(13) RESTORE THE REGISTERS
00003E 41F0 0000            00000   56+         LA    15,0(0,0) LOAD RETURN CODE
000042 07FE                         57+         BR    14 RETURN
000044                              58 SAVEAREA DS    18F
000000                              59          END   SAMPLE3
000090 7FFFFFFF7FFFFFFF             60            =2X'7FFFFFFF'
```

EXECUTION ERROR

SYSTEM COMPLETION CODE = 0C8 FIXED-POINT OVERFLOW

INTERRUPT OCCURRED AT LOCATION ** 1C0044 ** WHICH IS ** 000034 (HEX) ** BYTES INTO ROUTINE ** SAMPLE3 **

G P REGISTERS AT TIME OF INTERRUPT

REGISTER	HEX	DECIMAL
0	FA000020	-100663264
1	00000000	0
2	801E8F64	-2145468572
3	001C0010	1888272
4	801E8FFE	-2145468418
5	FFFFFFFF	-1
6	001E9F68	2015080
7	0C0000FF	255
8	FFFFFFFE	-2
9	7FFFFFFF	2147483647
10	001E8FF8	2015224
11	001C0000	1888256
12	401C0022	1075630114
13	001C0054	1888340
14	00014824	84004
15	011C0010	18665488

FLOATING POINT REGISTERS AT TIME OF INTERRUPT

REGISTER	HEX	DECIMAL
0	C12F12A8 CC000000	-2.942055463790894E+00
2	C0568624 CFCA9CC0	-3.387167937568050E-01
4	4168SC75 000C0000	6.522572517395020E+00
6	406587AE 00000088	3.966015577316310E-01

HEX CONTENTS OF INTERRUPT LOCATION = 1A89 58DD 0004 98EC

THE INSTRUCTION CAUSING THE INTERRUPTION IS : AR 8, 9

********** ERROR DIAGNOSIS **********

A CARRY HAS OCCURRED OUT OF THE HIGH-ORDER BIT POSITION OF REGISTER 8 - THE RESULT OF THE OPERATION
CAUSING THE INTERRUPT IS TOO LARGE TO BE EXPRESSED IN 32 BITS IN THE 2'S COMPLEMENT FORM

THE OPERATION HAS BEEN COMPLETED WITH THE RESULT LEFT IN THE REGISTER TOO LARGE OR TOO SMALL BY 2**32

THE INTERRUPTION COULD BE SUPPRESSED BY SETTING PSW BIT 36 TO A ZERO (DO NOT SPECIFY INTERRUPTION TYPE 8 IN THE SPIESET MACRO)
THIS WOULD CAUSE THE OVERFLOW CONDITION TO BE IGNORED

****

A STANDARD OS/MVT SYSTEM DUMP WILL FOLLOW IF A //SYSUDUMP DD CARD IS INCLUDED IN THE JCL STREAM
THIS DUMP SHOWS CONDITIONS CLOSELY APPROXIMATING THOSE AT TIME OF INTERRUPT

BIBLIOGRAPHY

1. IBM System/360 Operating System Assembler Language, OS Release 21, File No. S360-21, Order No. GC28-6514-8.

2. IBM System/360 Operating System Assembler (F) Programmer's Guide, Program No. 360S-AS-037, OS Release 21, File No. S360-21 (OS), Order No. GC26-3756-6.

3. IBM System/360 Operating System: Programmer's Guide to Debugging, OS Release 21, File No. S360-20, Order No. GC28-6670-6.

4. IBM System/360 Operating System: Supervisor Services and Macro Instructions, OS Release 21, File No. S360-36, Order No. GC28-6646-6.

5. IBM System/370 Principles of Operation, Order No. GA22-7000-3.

6. Struble, George, Assembler Language Programming: The IBM System/360, Reading, Massachusetts: Addison-Wesley Publishing Company, 1969.