Robert Rannie
Milica Kozomara
Northern Illinois University - DeKalb, IL
Texas Instruments - Dallas, TX

Mainframe Operating Systems "Boot Camp"

SVCs and More SVCs

Part 3

Session #2897
SHARE 112 in Austin, March 2009

Our Agenda for the Week

#2895 - Part 1: The General Purpose Computer and Interrupts

#2896 - Part 2: From IPL to Running Programs

#2897 - Part 3: SVCs and More SVCs

#2898 - Part 4: Program Interrupts (You Want An Exit With That?)

#2899 - Part 5: FLIH: I/O INTERRUPTS

#2894 - Mainframe Operating System Boot Camp: Highlights

"Tell 'em what you're gonna tell 'em"

- SVC First Level Interrupt Handler (FLIH)
- Scheduling a Supervisor Request Block (SVRB)
- Some Type 1 SVCs
- SVC 8 (Type 2) - Loader
- SVC 13 (Type 4) - Abend

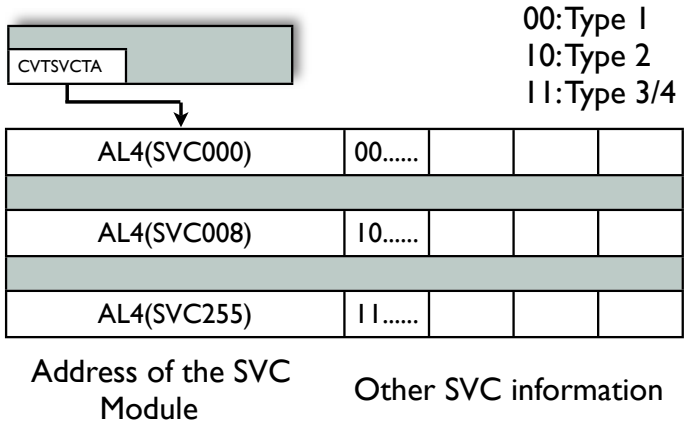
SVC First Level Interrupt Handler

Types of SVCs

Type	Runs?	Resident	Number of Loads
1	Disabled	Yes	N/A
2	Enabled	Yes	N/A
3	Enabled	No	1
4	Enabled	No	>1

SVC First Level Interrupt Handler

SVC Table



SVC First Level Interrupt Handler

Logic of SVC (all Types)

- First steps of FLIH must have PSW disabled for interrupts
- On entry to FLIH, current PSW has just been swapped in
- Some SVC routines must run disabled for their full length

SVC First Level Interrupt Handler

Logic of Type 1 SVCs

- Save the 'Essence' of the interrupted program
- Determine which SVC was called and BALR R14,R6 to that module.
- Following return, store the output parameters into the appropriate TCBGRS.
- Branch to the dispatcher.

SVC First Level Interrupt Handler

```
         IC    2,SVOPSW+3          Get Interruption Code from OldPSW
         SLL   2,3                 IC*8 = index of Addr. in SVC
*
         L     7,CVTSVCTA        get Loc(A(SVCMOD)) - R7 (convention)
         LA    7,0(2,7)            Address of this entry in table
         L     6,0(,7)           get     A(SVCMOD)  - R6 (convention)
*
         BALR  14,6              transfer control to SVCMOD
*
         STM   0,1,TCBGRS        Make new values R0, R1, R15
         ST    15,TCBGRS+15*4     available to user
*
         L     12,CVT0DS         Get A(DISPATCH)
         BR    12                Return to dispatcher
```

Scheduling a Supervisor Request Block

Reason for Multiple Types

- Not all SVCs run enabled
- Some SVCs need to be enabled for interrupts (at all times) and to be resident in the nucleus of the operating system. Why?
- RECALL:
  All dispatched programs run under control of a PSW with an I/O-External mask of X'FF': enabled for interrupts.

Scheduling a Supervisor Request Block

Definition of Process

- To minimize the number of SVCs running disabled for Interrupts, run most SVCs (Types 2,3, and 4):
  - Dispatched
  - Enabled for interrupts
  - by "Scheduling an SVRB"

Scheduling a Supervisor Request Block

Logic of SVC

- In FLIH, after obtaining A(SVC module):
- Determine SVC as Type 1 or Type 'Other'
- If (SVC is type 'other')
  - Acquire an SVRB
  - Point RBLINK in SVRB to PRB
  - Point TCBRB to SVRB

Scheduling a Supervisor Request Block

```
         LH    2,SVOPSW+2          Get Interruption Code from OldPSW
         SLL   2,3                 IC*8 = index of Addr. in SVC
*
         L     7,CVTSVCTA        get Loc(A(SVCMOD)) - R7 (convention)
         LA    7,0(2,7)            Address of this entry in table
         L     6,0(,7)           get     A(SVCMOD)  - R6 (convention)
*
*   STEP 9 : Determine SVC as Type 1 or Type 'Other'
*
         ICM   8,8,4(7)          Get SVC Type code
         BM    SVCTYPEO          If byte 0 is 1, then not Type 1
*
```

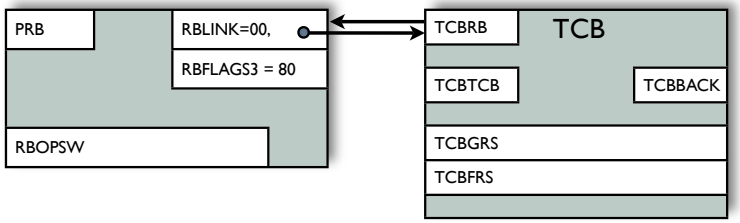
Scheduling a Supervisor Request Block

```
* TYPE 'OTHER'   TYPE 'OTHER'   TYPE 'OTHER'   TYPE 'OTHER'   ---------
*
*   STEP 14: Otherwise, do;
*              Schedule a new SVRB
*
SVCTYPEO DS    0H                       SVC requested is Type 2,3 or 4
         USING SVCTYPEO,8               Get addressability on this area
*
         GETCB100
*                               10 <-- A(SVRB)
         USING RB,10                                  Initialize SVRB:

         XC    0(CB100LTH,10),0(10)     Set all 256 bytes of SVRB to 0
*
         MVC   RBTYPE(4),NEWSVRB                 CL4'SVRB' --> RBTYPE
         MVI   RBFLGS3,X'00'                Indicate RBLINK is NOT TCB
         ST    4,RBTCB                                 A(TCB) --> RBTCB
*
         ST    5,RBLINK                      Set RBLINK to current PRB
         XC    RBLINK(1),RBLINK            Clear the high byte of RBLINK
*
         DROP  10                  End domain of R10
*
         LR    5,10                             Set current PRB to SVRB
         ST    5,TCBRB                             Chain SVRB into TCB
```

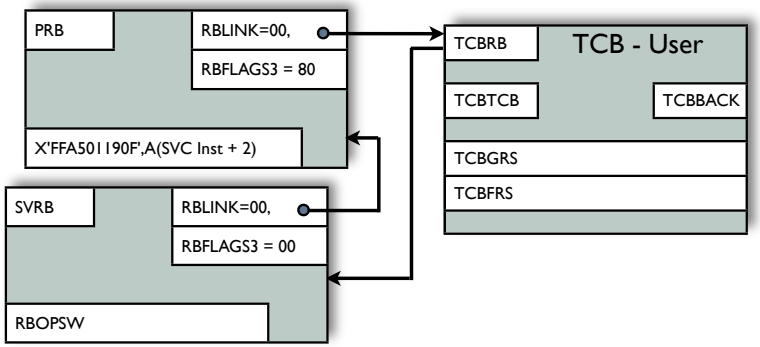

Scheduling a Supervisor Request Block

Before Acquiring SVRB - TCB & PRB



Scheduling a Supervisor Request Block

After Acquiring SVRB - TCB, PRB & SVRB



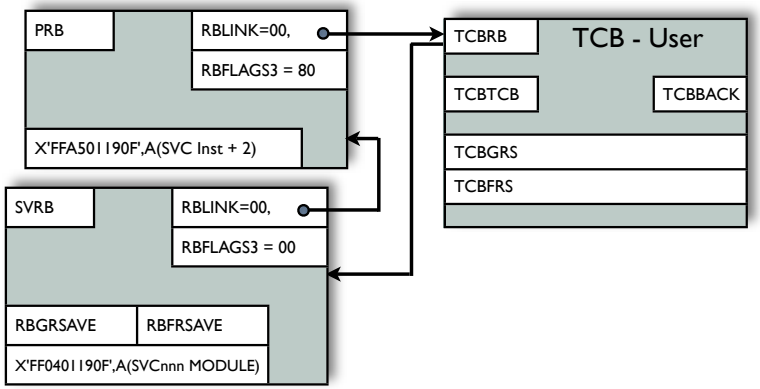
Scheduling a Supervisor Request Block

Logic of SVC

- Copy TCBGRS and TCBFRS from TCB into SVRB
- Set PSW in SVRB
  - I/O Enabled
  - Key 0
  - Supervisor State
  - Pointing to SVC Module

Scheduling a Supervisor Request Block

After Acquiring SVRB - TCB, PRB & SVRB



Scheduling a Supervisor Request Block

Logic of SVC

- Store 'standard' SVC FLIH registers (3, 4, 5, and 6) into TCBGRS
- Store A(CVTEXIT) into TCBGRS+(14*4).
- Branch to the Dispatcher!

Scheduling a Supervisor Request Block

Logic of SVC

- The SVCnnn Module will run enabled for Interrupts, in key 0, and in Supervisor State
- The Base register of all SVC Modules is R6
- All SVCs exit on R14 with return parms in R15, R0, and R1

Scheduling a Supervisor Request Block

Logic of SVC

- For SVCs that are dispatched with SVRBs, a different restoration process is necessary:

1. Return param registers to TCBGRS locations
2. 'Kill' the SVRB & return to dynamic CB pool

- We need the Type 1 SVC 3 (RB Killer) to accomplish this
- Let's examine four Type 1 SVCs

Some Type 1 SVCs

SVC 0 - EXCP/XDAP

- Input parameters:
- R1 A(Input/Output Block)
- SVC 0 uses Assist-V XREAD & XPRNT in explicit format:

XREAD 0(0,Rn),(Rm)
XPRINT 0(0,Rn),(Rm)
E0x0n000m000

Rn - A(Buffer)
Rm - Buffer Length
x = 0 for XREAD
x = 2 for XPRINT

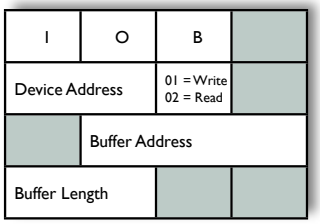
Some Type 1 SVCs

SVC 0 - EXCP/XDAP

- Perform three tests before proceeding to the XREAD/XPRNT:
  - Verify that the first 3 bytes of the IOB are C'IOB'.
  - Verify that the 2-byte Device Address is zero.
  - Verify that the IOBOPCDE is either X'01' or X'02'.

Some Type 1 SVCs

SVC 0 - EXCP/XDAP



Some Type 1 SVCs

SVC 0 - EXCP/XDAP

```
SVC0MOD  DS    0H                  Begin SVC0 Module
         USING SVC0MOD,6           Get addressability of SVC0 Module
         USING IOB,1               Get addressability of IOB
         USING CVT,3               Get addressability on CVT
*
DOEZIO   SR    15,15               Start with RC = 0
         L     7,IOBUFADD          R7 - A(IOB buffer)
         LH    8,IOBUFLEN          R8 - length of IOB buffer
*
*                                 Check the IOB operation code
         CLI   IOBOPCDE,X'01'       If X'01',
         BE    SVC0PRNT               write is requested.
         CLI   IOBOPCDE,X'02'       If X'02',
         BE    SVC0READ               read is requested.
         B     SVC00119            No other operation code is allowed
```

Some Type 1 SVCs

SVC 0 - EXCP/XDAP

```
*
* STEP 2 : If write is requested, perform XPRNT(v1.1) and set RC
*
SVC0PRNT XPRNT 0(0,7),(8)         Print the IOB record
         BR    14                 Return to SVCFLIH
*
```


Some Type 1 SVCs

SVC 0 - EXCP/XDAP

```
*
* STEP 3 : If read is requested, perform XREAD(v1.1)
*           and set RC accordingly
*
SVC0READ XREAD 0(0,7),(8)         Read the next card
*
         BM    SVC0EOF            If CC = 4 (EOF), set RC in R15 to 4
*                                    and go to end of SVC 0 module.
*                                Otherwise, (CC = 0)
         CLC   SVC0EOF9(8),0(7)      Test IOB buffer for 8C'9' (EOF)
         BE    SVC0EOF               If 8C'9' in IOB buffer,
*                                       treat as EOF (same as above).
*                                   Otherwise, valid read performed.
*
* STEP 4 : Return to SVFLIH
*
         BR    14                  Return to SVCFLIH (v1.1)
```


Some Type 1 SVCs

SVC 0 - EXCP/XDAP

- Assist-V's XREAD CC
  - 0 - successful read
  - 1 - End Of File encountered
- SVC 0 provides Return Code (0 or 4) in R15

Some Type 1 SVCs

SVC 1 & 2 - Wait & Post

- Input Parameters, SVC 1:
  - R0 - Number of events
  - R1 - A(Event Control Block)
- Input Parameters, SVC 2:
  - R0 - Post Code in bits 2-31
  - R1 - A(Event Control Block)

Some Type 1 SVCs

SVC 1 & 2 - Wait & Post
ECBCC(1) ECBRBA(3)

Some Type 1 SVCs

SVC 1 & 2 - Wait & Post

SVC 1			SVC 2		
W	P	Action	W	P	Action
0	0	Change WP to '10' A(RB) from R5 ST AL3(RB) in ECB+1 RBWCF = RBWCF + 1	0	0	Change WP to '01' No change to RBWCF Go to set Post Code in bits 2-31 of ECB
0	1	No-Op	0	1	No-Op
1	0	S301 Abend	1	0	Change WP to '01' L Rn from ECB & Using RB, Rn RBWCF = RBWCF - 1 Go to set Post Code in bits 2-31 of ECB
1	1	No-Op	1	1	No-Op

Some Type 1 SVCs

SVC 1 - Wait

```
SVC1MOD  DS    0H                Begin SVC1 Module
         USING SVC1MOD,6         Get addressability on SVC1MOD
         USING RB,5              Get addressability on RB
*
         C     0,SVC1ONE         If R0 is not = 1
         BNE   SVC10119             Abend (v1.1)
...
```


Some Type 1 SVCs

SVC 1 - Wait

```
...
*                                W P
*                                A O
*                                I S
*                                T T  bit tests:
*                           CASE;
         TM    0(1),X'40'        0 1  or      (not non-dispatchable)
         BOR   14                1 1: NoOp, return to SVFLIH
         TM    0(1),X'80'        1 0: S301 error,
         BO    SVC10119                      force local abend
         IC    15,RBWCF          0 0:
         LA    15,1(,15)         Make this task non-dispatchable:
         STC   15,RBWCF               Increment MSWCF
         STCM  5,7,1(1)               Put A(RB) in ECB+1
         OI    0(1),X'80'             turn ON WAIT bit
         BR    14                     return to SVFLIH
```

Some Type 1 SVCs

SVC 2 - Post

```
SVC2MOD  DS    0H
         USING SVC2MOD,6          Get addressability on the SVC
*
*                                 W P
*                                 A O
*                                 I S
*                                 T T  bit tests:
*                           CASE;
         TM    0(1),X'40'        0 1  or      (not non-dispatchable)
         BOR   14                1 1: NoOp, return to Caller
*
         ICM   15,7,1(1)               Get A(RB) from ECB
*
...
```


Some Type 1 SVCs

SVC 2 - Post

```
...
         TM    0(1),X'80'        0 0 or 1 0:
         ST    0,0(,1)                Store the post code
*                                     (bit 0 and 1 are overwritten)
         BZ    SVC2POST          0 0: Only perform post next
*
*                                1 0:
         USING RB,15                  Get addressability on this RB
         IC    0,RBWCF                Get RBWCF
         BCTR  0,0                       decrement it by 1
         STC   0,RBWCF                      and save the new value
         DROP  15                     End addressability on this RB
*
*
SVC2POST OI    0(1),X'C0'             Turn ON both W-P
         XI    0(1),X'80'             turn OFF WAIT bit
         BR    14                     return to Caller
```


Some Type 1 SVCs

SVC 3 - Exit, RB Killer

- No input or outut parameters
- SVC 3 will "Kill" the current RB by disconnecting its CB from the TCB and returning that CB to the dynamic memory pool.
- The TCBRB+1 field is updated with the contents of the RBLINK+1.

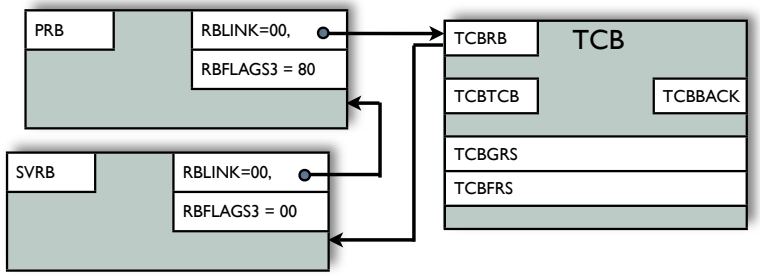
Some Type 1 SVCs

SVC 3 - Exit, RB Killer

- If there are multiple RBs chained off of the TCB via the TCBRB pointer, the current RB will be "Killed", its CB will be returned to the dynamic memory pool, and the next RB in the chain will now be pointed to via the TCBRB field

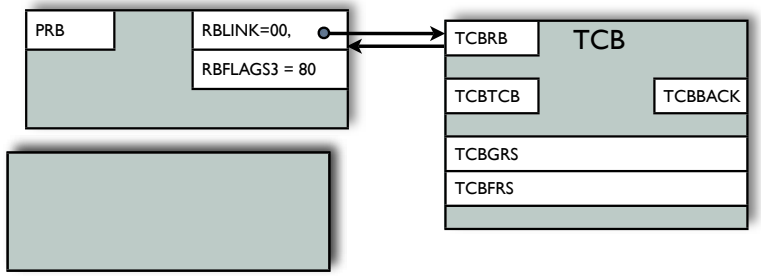
Some Type 1 SVCs

SVC 3 - Exit, RB Killer
Multiple RBs



Some Type 1 SVCs

SVC 3 - Exit, RB Killer
Multiple RBs



Some Type 1 SVCs

SVC 3 - Exit, RB Killer

- If the RB is the only one chained off of the TCB via the TCBRB pointer, the TCBRB field will be zeroed and when an attempt is made to dispatch this TCB, the dispatcher Job End Code will process the termination of this job

Some Type 1 SVCs

SVC 3 - Exit, RB Killer

```
SVC3MOD  DS    0H
         USING CVT,3             Base Reg. for Comm. Vector Table
         USING TCB,4             Base Reg. for Task Control Block
         USING RB,5              Base Reg. for Request Block
         USING SVC3,6            Base Reg. for SVC 3
*
         STM   0,1,RBGRSAVE            Save parameter registers
         ST    15,RBGRSAVE+15*4        in RB.
*
         MVC   TCBGRS(16*4),RBGRSAVE   Move all RB GPR's and
         MVC   TCBFRS(8*4),RBFRSAVE    and FPR's into TCB
*
         MVC   TCBRB+1(3),RBLINK+1  Move A(prev RB) into TCBRB
*
...
```

Some Type 1 SVCs

SVC 3 - Exit, RB Killer

```
...
         CLM   4,7,TCBRB+1          If the RB just unchained was
         BNE   SVC3FG0                 the last RB in the chain
*
         TM    RBFLGS3,X'80'        and the flag bit is one
         BNO   SVC30119                (force local abend otherwise)
*
         XC    TCBRB(4),TCBRB       Then set the TCBRB to 0
         B     LOADGPR5             To reload GPR5 code
*
SVC3FGO  TM    RBFLGS3,X'80'        test to confirm flag = 0
         BO    SVC30119             (force local abend otherwise)
*
LOADGPR5 L     5,TCBRB              GPR5 = 0, or next RB in line
         BR    14                   return from SVC 3
*
SVC30119 XOPC  25    = X'0119'      ASSIST-V:terminate simulation
```


Some Type 1 SVCs

Why do you kill an RB?

- Remember:
  When the dispatcher finds a TCB with its TCBRB set to zero the dispatcher goes to its 'Job End' code, ends the processing of that job and prepares to process the next job.

Some Type 1 SVCs

SVC 3 - Exit, RB Killer

- You may be surprised to learn how all of your jobs have ended
- How do programmers end programs?
  - BR 14
  - R14 - a value received on entry
- Now you know how to find A(THECVT)
  - R14 - A(MYCVT+X'50'), What is there?

SVC 3

SVC 8 - (Type 2) - Loader-Lite (LL)

- The output of an Assembler produces at least four types of "Object Module (O/M)" records: ESD, TXT, RLD, and END
- These records possess the pertinent information that was contained in the Assembly source program.
- A linkage editor or a loader can use the O/M to create an executable version of the program in the computer memory.

SVC 8 - (Type 2) - Loader-Lite (LL)

- SVC-8 Input:
  - R0 = Program Load Address
  - R1 = Available Memory
- SVC-8 Output:
  - R0 = Entry Point Address
  - R1 = Size of Loaded Program
  - R15 = Return Code (RC), 0=OK, >0=Not!

SVC 8 - (Type 2) - Loader-Lite (LL)

- LL typically executes a five-step program.

1. Use HLASM to assemble SOS SVC-8
2. Copy the SVC-8 O/M to have two O/Ms
3. Use AMBLIST to format & display O/M
4. Use IEBPTPCH to view all bytes of O/M
5. Run Assist-V. Supplied SVC-8 loads SOS SVC-8, that SVC-8 loads 2nd SOS SVC-8, that SVC-8 loads all future test programs.

SVC 13 (Type 4) - Abend

- Entry Parameters:
  - R1

    1. Flag Byte - If Bit 0 = 1 - Dump request
    2. System ABEND Code (3 hex digits)
    3. User ABEND Code (~ 4 decimal digits)
- Exit Parameters: None

SVC 13 (Type 4) - Abend

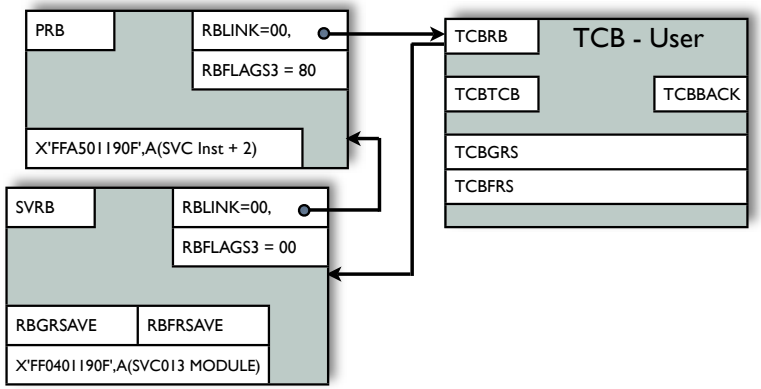
- SVC 13 formats and writes out (USING SVC 0) the following:
  - interrupt code (ALL of R1)
  - the PSW
  - twelve bytes of memory "centered" on the A in the PSW
  - address of the first of the 12 bytes
  - GPRs and FPRs from SVRB

SVC 13 (Type 4) - Abend

- Finally, chain through all RBs and point PSWs at the CVTEXIT
- Exit SVC 13 normally with BR R14

SVC 13 (Type 4) - Abend

Location of PSW & Regs at Abend



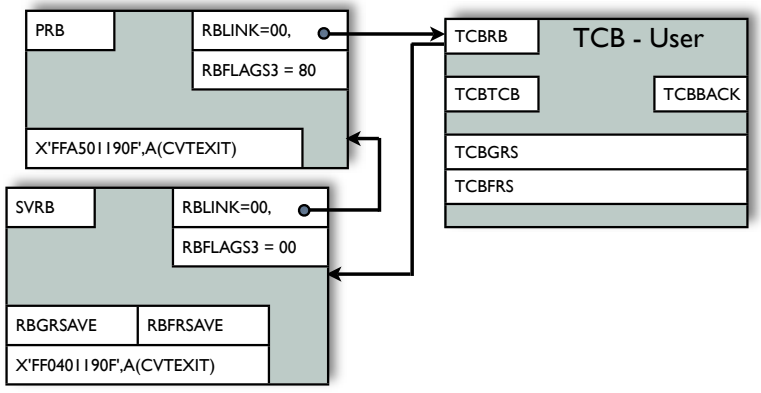
SVC 13 (Type 4) - Abend

Sample Indicative Dump (a REAL one)

```
IEA995I SYMPTOM DUMP OUTPUT
SYSTEM COMPLETION CODE=0C1 REASON CODE=00000001 TIME=14.51.21 SEQ=00168 CPU=0000 ASID=002F
     PSW AT TIME OF ERROR  078D0000 0000E012   ILC 2   INTC  01  ACTIVE LOAD MODULE=**GO
ADDRESS=0000E000
                                                          OFFSET=00000012
     DATA AT PSW 0000E00C - 00000000   00010203   04050607
     GPR 0-3     00005F40   00005F60   80005F64   0000E010
     GPR 4-7     00005FE8   FFFFFF13   00005F88   00000012
     GPR 8-11    00000000   00005EA8   00005FD8   0000E000
     GPR 12-15   00006D4C   00005E58   80FD3408   0000E010
END OF SYMPTOM DUMP
+IEW1991 ERROR - USER PROGRAM HAS ABNORMALLY TERMINATED
IEF450I RRHABEND GO-ABEND=S0C1 U0000 REASON=00000001
```


SVC 13 (Type 4) - Abend

A (CVTEXIT) in all PSWs to finish Abend



Sneak Preview

Part 4: Program Interrupts (You want an exit with that?)

- Program Mask: Friend or Foe?
- Program Check Abend with SVC 13: From Inside SVC-FLIH?
- SPIE or NOT?
- SVC 14 in SOS: Tougher than SPIE?
- Safe Return From Exits? Only with "Portia"!

Questions

rrannie@cs.niu.edu
m-kozomara@ti.com
www.cs.niu.edu/~rrannie