Mainframe Operating Systems “Boot Camp”
Program Interrupts (You Want An Exit With That?)

Part 4

Session #2898
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”

- The Program Mask (PM)
- Program Check FLIH (PC-FLIH)
- SVC 14 (Type 3) - SPIE
- Rules for Operation in a SPIE Exit Routine
- 'Portia' Modifications to SVC 3

The Program Mask (PM)

Program-Interruption Codes Under Program Mask Control

- SOC8 - 0008 Fixed-point overflow exception
- SOCA - 000A Decimal-overflow exception
- SOCD - 000D Exponent-underflow exception
- SOCE - 000E Significance exception

Mask bit = 0: Exception will Not cause an Interrupt
Mask bit = 1: Exception will cause an Interrupt

<table>
<thead>
<tr>
<th>Channel Masks</th>
<th>E</th>
<th>PSW key</th>
<th>c</th>
<th>m</th>
<th>w</th>
<th>p</th>
<th>Interruption Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>34</td>
<td>36</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td>Instruction Address</td>
</tr>
</tbody>
</table>

| Mask bit = 0: | Exception will Not cause an Interrupt |
| Mask bit = 1: | Exception will cause an Interrupt   |
The Program Mask (PM)

More About Program Mask

- To determine current state of PM
  - BAL/R
  - IPM (in MVS)
- To set PM (from Problem state)
  - SPM

Program Check FLIH (PC - FLIH)

No SPIE Exists (or Applies)

Use SVC 13 to abend program causing PC

- Provide input parameter in R1
  - X'800Cn000'
  - n - (hex) type of Program Interrupt
- Load A(BRABEND) from CVTBRABN into R15
- BR R15

Program Check FLIH (PC - FLIH)

No SPIE Exists (or Applies)

Note: The BRABEND code need not be a separate module as is done in SOS. This code may simply be a part of the PC-FLIH.

- BRABEND is equivalent to scheduling and dispatching an SVRB for the execution of the SVC 13.
- Rather than duplicating the code that you wrote in the SVC FLIH in order to accomplish this, you should do the following:
Program Check FLIH (PC - FLIH)

No SPIE Exists (or Applies)

- In BRABEND:
- Load A(SVC 13 module) from A(SVCTABLE) +13*8 into R6
- Get A(‘Type Other’ code in SVC-FLIH) from CVTSSVRB
- With R6 and R1 (interrupt code X’800Cn000’) set, branch into ‘Type Other’ code in SVC-FLIH.

Program Check FLIH (PC - FLIH)

SPIE environment Exists

| TCB+4 | Points to the SCA (TCB & SCA in memory protected from user) |
| SCA+0 | Points to the PIE (In user accessible memory) |
| PIE+0 | Points to the PICA (In user accessible memory) |

Program Check FLIH (PC - FLIH)

- If no SPIE environment exists, TCB+4 (TCBPIE) will be zero
- If there is a SPIE environment, there will be three Control Blocks
  - SCA - Spie Control Area (4 byte in SOS)
  - PIE - Program Interrupt Element (8 Full Words)
  - PICA - Program Interrupt Control Area (6 bytes)
SPIE environment Exists

- TCB and PICA exist for duration of “task”
- SVC 14, when called to create a SPIE environment, dynamically acquires an SCA (protected memory) and a PIE (user accessible memory)
- If the SPIE (and all requisite CBs) exists and is applicable to the type of interrupt that occurred, the PC-FLIH will dispatch the program into its Exit Routine.

SPIE environment Exists - PC-FLIH Program Logic

- If there is a SPIE, the chain of control blocks must exist and contain correct values.
- If an error is encountered at any point ABEND the program via BRABEND.

SPIE environment Exists - PC-FLIH Program Logic, Step 1

- Get TCBPIE from TCB of interrupted program
- If TCBPIE is zero, ABEND
- Get A(SCA) from TCBPIE
- If SCA+0 is zero, OR
- if InEx Flag = 1, ABEND
- Get A(PIE) from SCA
Program Check FLIH (PC - FLIH)

SPIE environment Exists - PC-FLIH Program Logic, Step 1

- If PIE+0 is zero, OR if PIE InEx Flag = 1, ABEND
- Get A(PICA) from PIE
- If A(Exit Routine) is zero, OR if ‘S0C FLGs’ is zero, ABEND

PICA

0  PM  AL3( Exit Routine)  SOC FLGs

SPIE environment Exists - PC-FLIH Program Logic, Step 2

- At this point, the CB chain has been validated.
- Test the type of PC interrupt that occurred for SPIE support:
  - Supported S0Cs are indicated by B’1’ in corresponding field in PICA
- If this interrupt type is not specified by the PICA, ABEND

SPIE environment Exists - PC-FLIH Program Logic, Step 3

- To send the program to its Exit Routine
  - Move the contents of R14 - R2 from TCBGRS to PIE+12
  - Move the 8 bytes of PSW from RBOPSW into PIE+4
  - Store A(Portia SVC 3) into TCBGRS+14*4

- To send the program to its Exit Routine
  - Store A(Exit Routine) into TCBGRS+15*4
  - Store AL3(Exit Routine) into RBOPSW+5
  - Store A(PIE) into TCBGRS+1*4
  - Set InEx Flags at SCA+0 and PIE+0 to B’1’
Program Check FLIH (PC - FLIH)

SPIE environment Exists - PC-FLIH Program Logic, Step 4

- Branch to the dispatcher

SVC 14 (Type 3) - SPIE

Overview

- Entry: \( R1 = A(\text{PICA}) \)
- Exit: \( R1 = \text{Address of previous PICA (or zero if there was none)} \)
- SPIE performs one of three options
  - Create a SPIE Environment
  - Modify a SPIE Environment
  - Cancel a SPIE Environment

SVC 14 (Type 3) - SPIE

Overview

- Test for zero in three locations
  - \( A(\text{PICA}) \) in \( R1 \)
  - \( A(\text{Exit Routine}) \) in Pica
  - 15 Flags for \( \text{S0C1-S0CF} \) in PICA

- If ANY of these are zero, that constitutes a request to
  - Cancel a SPIE Environment (if one exists)
  - Ignore the SPIE request (if no Environment exists)
SVC 14 (Type 3) - SPIE

Overview

- The first step for any of the three SPIE options:
  - If TCBPIE is zero, no SPIE Environment exists
  - If it is non-zero, a SPIE Environment (may) exist

Create a SPIE environment - Program Logic

- Obtain an SCA (SOS on TCB at TCBDMSCA)
- Store A(SCA) into TCBPIE + 1
- Store A(PIE) into SCA+1
- Set SCA+0 to X'00'
- Store A(PICA) into PIE+1
- Set PIE+0 to X'00'

Modify a SPIE environment - Program Logic

- Confirm that a valid SPIE Environment exists by checking the chain of CBs
  - If any items are incorrect, XOPC 25 (force abend)
  - NON-zero TCBPIE contains A(SCA)
  - NON-zero SCA contains A(PIE), and bit 0 is zero
  - NON-zero PIE contains A(PICA), and bit 0 is zero
SVC 14 (Type 3) - SPIE

Modify a SPIE environment - Program Logic

• Save A(OLD! PICA) from PIE+1
  Move A(NEW! PICA) into PIE+1
• Set PIE+0 to 0
• Move PM from right hit of NEW! PICA+0 into
  PM field of PRB PSW
• Put the address of the previous saved A(OLD!
  PICA) into Reg. 1 for return to the program
  that issued SVC 14

Note: A request to Modify a SPIE Environment
does NOT call for you to change the (original)
PSW PM that is stored in the TCBPIE+0

SVC 14 (Type 3) - SPIE

Cancel a SPIE environment - Program Logic

• When one of the “three circumstances” signals
  that a SPIE Environment is to be canceled, the
  first test is of the TCBPIE to see if a SPIE
  Environment exists
• If it does not, return a zero in R1

• If a SPIE Environment does exist
  • Move the saved PSW PM from TCBPIE+0
    into the PM field of the PRB PSW
  • Obtain A(SCA) and clear the TCBPIE to
    zero
  • Obtain A(PIE) and clear the SCA
    (TCBDMSCA) to zero
SVC 14 (Type 3) - SPIE

Cancel a SPIE environment - Program Logic

- If a SPIE Environment does exist
  - Save A(PICA) from PIE+1 and zero the first four bytes of the PIE
  - Put A(PICA) into R1 for return from SVC 14

Rules for Operation in a SPIE Exit Routine

Overview

- On entry to an Exit Routine (ER)
  - R15 is base register
  - R14 is A(Porta SVC 3)
  - R1 is A(PIE)

- On entry to the ER you may STM R0,R15,EXITSAVE
- If that is done you must LM R3,R14,EXITSAVE +3*4 before BR R14
- Changes to R14 - R2 that you want to be in effect on return from the ER should be made to the register areas in the PIE
- Make any desired changes to R3 - R13 at the appropriate location in EXITSAVE
'Portia' Modifications to SVC 3

Overview
- Create a 'Portia' SVC 3 located IMMEDIATELY before the entry point of the PC-FLIH
- Modify SVC 3 to for the 'Portia' concept
  - On entry to SVC 3, test whether this is a 'Portia' entry or whether this is a regular entry
  - To do this, determine if the SVC 3 instruction was the one located immediately before the PC-FLIH

Program Logic
- If SCA InEx bit = 0, XOPC 25
  Otherwise, set it to 0 and set the PIE InEx bit to 0
- R3-R13 are now in TCBGRS as they were at the time of the BR R14 from ER

- If this is not a 'Portia' entry
  - Proceed with standard SVC 3 code
- If this is a 'Portia' entry
  - Run the CB chain from the TCB to get A(SCA) and A(PIE)
  - Do NOT assume that R1 points to the PIE!
'Portia' Modifications to SVC 3

- Remember what happens in the SVC-FLIH immediately upon return from a Type-1 SVC module? The SVC-FLIH moves R15-R1 into the corresponding TCBGRS.
- It is therefore, IMPERATIVE, that on exit from SVC 3 the contents of R15-R1 are the SAME as the contents of those register locations in TCBGRS!

Actual Program Logic

- Move R14 - R2 from PIE to TCBGRS
- Move the second half of the PSW from the PIE to the RBOPSW+4
- Load R15 and R0 - R1 from TCBGRS
- Exit the SVC 3 module via BR R14 and return to the SVC-FLIH

Sneak Preview

Part 5: FLIH: I/O Interrupts

- Priority: It's Just Swiss & American
- So that's why we need a Wait TCB/RB
- I/O, All it is: Get it going; Get it finished
- But, It's asynchronous: Can you catch it?
- Partial solution: More (and better) CBs

Questions

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