17555: Lab: z/OS Debugging IPCS Hands-on Lab

Wednesday, August 12
12:30 PM

John C. Shebey III
IBM Poughkeepsie
jshebey@us.ibm.com

Patricia Little
IBM Poughkeepsie
plittle@us.ibm.com

Lab Agenda

How the Lab is Presented

- **Self-Paced**
  - Be warned there is a lot of material in this lab. You may not finish in the time allotted.
  - '**' on page title indicates exercises on that page.
    - Each exercise details commands to be entered.
- **This lab session lets you experiment with some of the IPCS commands and displays discussed in lecture session 17907. Please refer to lecture session 17907 for sample command display output.**

Downloading Dumps via FTP

- ALL 4 data sets used in today's lab are available for FTP download.
- Download from TESTCASE.BOULDER.IBM.COM, login as ANONYMOUS, and use your complete E-MAIL address as the password.
- Files are in BINARY format in the ‘fromibm/mvs’ directory.
- Filenames are ‘share.s17555.dumpN.trs.bin’ (N is 1 - 4; filenames are lower case).
- All files are in tersed format, so will need to be untersed after download.
- Additional information on using TRSMAIN or FTPing files can be found at the following web site:
  - http://techsupport.services.ibm.com/390/tcprocs.html

Related Sessions

- The following session may be helpful in the diagnosis of problems. You may wish to refer to this session on the SHARE proceedings if you are unable to attend:
  - S17907: z/OS Debugging: Diagnosing Loops and Hangs
    Thursday, August 13, 1:45pm

**Lab Setup Instructions

- On the first panel, only specify Application: **TSO**

```
Enter Your Userid:
Password:                                New password:
Application: TSO
Application Required. No Installation Default
```

- Log onto **SHARAnn** where nn is the # on the front of this handout. The password you specify will be **FIRSTPW**. Specify TSO Logon Procedure: **SHARE**
- Once in ISPF on the command line, type: **IPCS**

- At the IPCS primary options panel, choose option 0 (zero) for DEFAULTS.
- Fill in the default screen to look like this:
  You may change any of the defaults listed below. Note that we're changing the Scope to BOTH so both the LOCAL and GLOBAL defaults get updated.

```
Scope   ==> BOTH    (LOCAL, GLOBAL, or BOTH)

If you change the Source default, IPCS will display the current
default Address Space for the new source and will ignore any data
entered in the Address Space field.

Source  ==> DSNAME('SHARE.S17555.DUMP1')
Address Space   ==>
Message Routing ==> NOPRINT TERMINAL
Message Control ==> CONFIRM VERIFY FLAG(WARNING)
Display Content ==> NOMACHINE REMARK REQUEST NOSTORAGE SYMBOL
```

- Hit <ENTER> and go to IPCS Option 6 (Commands Menu) by typing '**=6**' on the Command line. Proceed with the lab exercises.
  - When the lab refers to <ENTER>, it's really the <CTRL> key.
  - Press <ENTER> after each IPCS command.
  - The <PF3> key exits an IPCS report and goes back to the previous IPCS screen.
  - When viewing IPCS output or in BROWSE mode, scroll up via PF7 and down via PF8. Scroll left via PF10 and right via PF11.

AFTER SETTING UP THE SOURCE DSNAME AS ABOVE, DO NOT CHANGE AGAIN UNTIL DIRECTED TO DO SO IN THE LAB EXERCISE INSTRUCTIONS

**Lab 1: Dump of Hung Job

- Issue command: **ST SYSTEM**
  - This will result in dump initialization. Press <ENTER> after you see '***' when the dump has completed initialization:

```
BLS18123I 14,369 blocks, 59,775,040 bytes, in DSNAME('SHARE.S17555.DUMP1')
IKJ56650I TIME-11:19:00 PM. CPU-00:00:00 SERVICE-69174 SESSION-00:57:56 JULY 29,2015
BLS18224I Dump of z/OS 02.01.00-0 - level same as IPCS level
***
```

  - What type of dump is this? _____
    (Hint: Program Producing Dump)
- Now issue '**IP L TITLE**' on the Command line
  - What is the dump title? _____
    - System generated dumps typically have a COMPID= and other system generated information depending on the recovery routine that takes the dump.
    - Console dumps will have a title of whatever the user puts in COMM= as the dump title.
    - Dumps taken as a result of a slip trap have SLIP trap id in them.
    - Any program can issue an SDUMP macro and generate a title of its choosing
  - Based on the title of this dump, make a guess as to which type of dump this is:
    - STANDALONE DUMP
    - A CONSOLE DUMP
    - SLIP TRAP GENERATED DUMP
    - SYSTEM GENERATED DUMP
  - Does the dump title give a possible indication about which job is hung? _____
    - If so, which job? _____
- Issue command: **IP CBF CVT**
  - What is the z/OS release level on which this dump was taken? This is useful because some of the IPCS commands require the z/OS release level of IPCS to match the z/OS release level of the dump.
    (Hint: Issue command: **F PRODI**)
    - HBB7780 (z/OS V1R13)
    - HBB7790 (z/OS V2R1)
    - HBB77A0 (z/OS V2R2)
- On the Command line, issue '**IP CBF RTCT**' to identify which address spaces were dumped. Then, issue command '**F SDAS**'. The ASIDs dumped are listed under the SDAS heading.
  - Which ASIDs were dumped? _____
- Now, issue '**IP SELECT ALL**' to correlate the ASIDs identified above with jobnames.
  - What jobname is associated with ASID x'29'? _____
  - What jobname is associated with ASID x'2A'? _____

**Lab 1: Dump of Hung Job (cont)

- On the Command line, issue command: **IP ST WORKSHEET**
  - Notice that the Dump Title is also listed in this report. This is followed by the Date _____ and Time _____ of the dump and the Original dump dataset _____.
  - Identify the name of the system. _____
    (Hint: Issue command: **F SNAME**)
  - Issue command '**F CSD**' to get to CPU information. How many active CPUs? ____
    (Hint: Look at #CPUs for Alive Mask)
  - Note from the bit masks that the active CPU is a general CPU (CP), not a specialty zIIP or zAAP CPU. From the PROCESSOR RELATED DATA, the CPU number is ____.
- Now, let's determine which SDATA options were requested with the dump. This may be important to verify that the storage required to diagnose a problem was requested. Issue the command '**IP CBF RTCT+9C? STR(SDUMP) VIEW(FLAGS)**'. Then issue '**F SDUSDATA'** to get to SDUSDATA flags.
  - Was LSQA requested on the dump? _____
  - Was RGN requested (shown as rgn-private area)? _____
- The SDUEXIT flags in the output above indicate whether certain component exits received control for this dump. Issue '**F SDUEXIT**' to get to these flags. Was GRSQ specified? _____
- Recall from the dump title that job SHRENQ2 is hung. One possible reason for a job to be hung is resource contention. Issue command '**IP ANALYZE RESOURCE**' to display resource contention. Then, issue command '**F SHRENQ2**' to check whether job SHRENQ2 is waiting for a resource? _____
  - What is the MAJOR name of the ENQ? _____
  - What is the MINOR name of the ENQ? _____
  - What is the SCOPE of the ENQ? _____
  - Which job holds this ENQ? _____
- Now, the question becomes why job SHRENQ1 is not releasing this ENQ, which is required by job SHRENQ2. Issue command '**IP VERBX MTRACE**' to display SYSLOG messages just before the dump was taken. Issue command '**F SHRENQ1**' to look for any messages pertaining to SHRENQ1.
  - What time was the SHRENQ1 job started (S SHRENQ1)? _____
  - Check for subsequent messages to determine what the SHRENQ1 job did after it started.
    - Is there anything that could be done on the console to relieve this problem?
      _____
      _____

**Lab 2: Dump of Looping Job

- Switch the Source DSNAME by typing '**=0**' (zero) on the IPCS Command Line.
  - Change the DSNAME to 'SHARE.S17555.DUMP2' and the SCOPE to BOTH, and then hit <ENTER>.
- Issue command '**IP ST SYSTEM**'
  - This will result in dump initialization. Reply '**Y**' to the following prompt and hit <ENTER>:

```
BLS18160D May summary dump data be used by dump access?
Enter Y to use, N to bypass.
```

  -
  - Press <ENTER> after you see '***' when the dump has completed initialization:

```
BLS18123I 14,119 blocks, 58,735,040 bytes, in DSNAME('SHARE.S17555.DUMP2')
IKJ56650I TIME-11:00:16 PM. CPU-00:00:00 SERVICE-58813 SESSION-00:39:11 JULY 29,2015
BLS18224I Dump of z/OS 02.01.00-0 - level same as IPCS level
***
```

  - Note the dump type: _____

- Then, issue command **IP ST WORKSHEET**
  - Note the dump title: _____
  - How many CPUs are active on the system? _____
    - Which CPU # is active? ___
- Issue '**IP CBF RTCT**' to identify which address spaces were dumped. Then, issue command '**F SDAS**'. The ASIDs dumped are listed under the SDAS heading.
  - Which ASIDs were dumped? _____
- To correlate the ASID identified above with its jobname, issue: **IP SELECT ASID(x'1A')**
  - What is the jobname? _____
- Issue '**IP SYSTRACE ALL TI(LO)**' and check for any processor WAIT trace entries. Do you find any? ____
  (Hint: Issue command: **F WAIT 21**)
  (Refer to page 29 in Appendix for more info about fields in SYSTRACE report)
  - The absence of any processor WAIT entries is indicative that the system is busy running at 100% CPU.
- Since this dump was supposedly taken of a looping job, it's reasonable to assume that the looping job is ASID(x'1A'). Issue command: **IP SYSTRACE ASID(x'1A') TI(LO)**
  - Do you see any pattern with the trace entries? _____
  - What pattern do you see?
    (Note: The PC # is in the UNIQUE-2 field of the PC trace entry)
    _____
    _____

**Lab 2: Dump of Looping Job (cont)

- What is the PSW address of the PC 30B (Storage Obtain)? _____
  - Issue '**IP WHERE 25900040**' to determine the issuing module (and hex offset)
    _____
    - The PSW address actually points after the Storage Obtain PC. To determine the true address of the PC, we need to subtract the length of a PC instruction (4 bytes) from the PSW address: x'2590003C' (LOOPER+3C)
- What is the PSW address of the PC 311 (Storage Release)? _____
  - Issue '**IP WHERE 2590006E**' to determine the issuing module (and hex offset)
    _____
    - The PSW address actually points after the Storage Release PC. To determine the true address of the PC, we need to subtract the length of a PC instruction (4 bytes) from the PSW address: x'2590006A' (LOOPER+6A)

- It looks like the LOOPER module is in a loop on TCB(x'8F8238') obtaining and releasing storage. The SSRV 132 trace entry provides more details about the Storage Obtain, and the SSRV 133 trace entry provides more details about the Storage Release. (Note: the contents of the SSRV trace entries are described in z/OS MVS Diagnosis: Tools and Service Aids, Chapter 8).

```
0001 001A 008F8238  SSRV   132    00000000  00000672 00005FB8 25901000
                                             001A0000
0001 001A 008F8238  SSRV   133    00000000  00000603 00005FB8 25901000
                                             001A0000
```

  - The UNIQUE-3 field (in bold above) on the SSRV 132 and SSRV 133 trace entries contains the storage address being obtained/released.
  - Examine the SYSTRACE entries more closely for ASID(x'1A'). Is the code obtaining and releasing different storage areas or the same storage area?
    _____

- Based on what we've determined, the LOOPER code appears to be issuing a Storage Obtain at offset x'3C' and then looping back after it issues the Storage Release at offset x'6A'.
  - Issue the following command to review the Storage Release PC and subsequent instructions: **IP LIST 2590006A LEN(x'20') I**
    (The 'I' operand interprets the storage as Assembler instructions. The Assembler instructions are displayed to the right, with the associated machine code displayed to the left.)
    - Which instruction is at address x'2590006E'? _____
  - Thus, LOOPER is branching backwards (Branch Relative on Condition) x'54' bytes right after the Storage Release. Hence, the loop!

**Lab 3: ABEND0C1 Dump

- As described previously, switch the Source DSNAME to: SHARE.17555.DUMP3
- Issue command '**IP ST SYSTEM**' to determine the dump type: _____
  - This will result in dump initialization. Reply '**Y**' to the summary dump prompt.
- Determine the reason for this dump by issuing the '**IP LIST TITLE**' command. From the output, is there any indication that this dump was the result of an ABEND? If so, what ABEND? _____.
  - An ABEND0C1 occurs following a PIC 1 (Program Interrupt Code 1) when the CPU attempts to execute an instruction with an invalid operation code (opcode).
- Issue '**IP SYSTRACE ALL TI(LO)**', and then go to the bottom of the report by typing '**M**' on the command line followed by **<PF8>** key. Once at the bottom, issue command: **F *PGM PREV**

```
 PR   ASID WU-Addr- Ident  CD/D PSW----- Address- Unique-1 Unique-2

0001 001A 008FF1C0 *PGM    001 00000000 259000B4 00060001 00000000
                               07850000 80000000           7F6DA400
```

(Note that PGM 001 represents a PIC 1).

- Record the failing ASID _____ and TCB (WU_Addr) _____.
  - Use the '**IP SELECT ALL**' command to determine the jobname: _____
- At which PSW address did the PGM 001 occur? _____
  (Note: The PGM entry displays the 128-bit (4 word) PSW on 2 lines, but not in the order of bit 0 to bit 127. The instruction address (2 words) is displayed first, followed on the second line by the first half (2 words) of the PSW).
    - Issue '**IP WHERE 259000B4**' to determine the failing module (and hex offset): _____
- The first half (left 2 bytes) of 'Unique-1' contains the failing instruction length (ILC).
  - Note the ILC value: _____
- The second half (right 2 bytes) of 'Unique-1' contains the failing PIC:
  - Note the PIC value: _____

- Fill in the ABEND code in the *RCVY entry below, based on the *RCVY entry that immediately follows the *PGM 001 entry in SYSTRACE:
  - *RCVY PROG 94 __ __ __ 000 (fill in the 3 missing characters)
- Use the '**IP ST FAILDATA**' command to get the relevant information about the ABEND0C1. Issue **F 'TIME OF ERROR'** to get to the relevant section.
  - Record the following:
    - PSW address (right 2 words) _____
    - Instruction length (ILC) _____ and Interrupt code _____
    - Failing instruction text _____
    - Breaking event address (BEAR) _____
  - Following the 'Breaking event address' are the registers at time of error. The registers are formatted as AR (Access Register) / GR (General Register) pairs. Note that each AR is 1 word (32 bits) in size, and each GR is 2 words (64-bits) in size.
    - What are the values of GR 14 _____ and GR 15 _____?

**Lab 3: ABEND0C1 Dump (cont)

- Below the registers are the Home, Primary, and Secondary ASIDs.
  - Note the Home ASID: _____
- Then the report formats “This Task's ASID/TCB:”
  - Record the ASID _____ and TCB _____.
- Take notice that the information we found from ST FAILDATA matches the information we found from SYSTRACE. This information is built by RTM (Recovery Termination Manager) and gets stored in an SDWA (System Diagnostic Work Area) that gets passed to recovery routines.

- Now that we have information from the time of the ABEND0C1, we need to determine why we encountered the PIC 1. The most common reasons for a PIC 1 are a wild branch or overlaid code.
  - Let’s examine the “Failing instruction text” more closely. The PSW address always points 6 bytes into the failing instruction text. In other words, this text contains 6 bytes before the PSW, and 6 bytes after. The specific PIC determines whether the PSW points “at” or “after” the failing instruction.
    - In the case of a PIC 1 (ABEND0C1), the PSW points “after” the failing instruction. We need to back up by the number of bytes of the Instruction length (ILC) that we found earlier. Record the failing instruction: _____
    - This looks like EBCDIC data! Let’s confirm by LISTing the storage. We get the failing instruction address by taking the PSW address (x’259000B4’) and subtracting 6 bytes (ILC value): **IP LIST 259000AE LEN(6)**
      - Record the EBCDIC translation you see between the bars | | on the right: _____
  - We found above that we tried to execute EBCDIC code at address x’259000AE’. So, how did we get here?
    - A common branching technique is a BALR 14,15 where a program branches from the code pointed to by GR 14 to the code pointed to by GR 15.
    - Look at the value of GR 15 that we recorded earlier (from ST FAILDATA). Does it point to our failing instruction address? _____
    - Now look at the value of GR 14 that we recorded earlier. A BALR is a 2-byte instruction. If a BALR was done, GR 14 would point to the instruction after the BALR. So, if we subtract 2 bytes from the GR 14 address, and turn off the high-order addressing mode bit, we end up with an address of _____.
    - Now, the Breaking Event Address (BEAR) that we recorded earlier from ST FAILDATA comes into play. BEAR is a 64-bit register (filled in by hardware following a program interrupt) containing the address of the last instruction causing a break in sequential execution of code (last branch). Does this match the address we derived above from GR 14? _____
      - As you can see, the BEAR is very useful for debugging ABEND0C1’s, especially those due to a wild branch. It can help eliminate guesswork from the picture.

**Lab 3: ABEND0C1 Dump (cont)

- Lastly, let's confirm that there really is a BALR instruction (2 bytes) at address x'259000A4'. Issue '**IP LIST 259000A4 LEN(2)**
  - Record the 2 bytes seen: _____
  - Issue command '**IP OPCODE 05EF**' to determine the opcode (mnemonic) associated with this instruction? Is it a BALR? _____
- Issue '**IP WHERE 259000A4**' to determine who issued this BALR to an invalid instruction. _____

o The next step would be to review the Assembler code listing for this ABEND0C1 program to determine why it branched to an invalid instruction at offset x'A4'.

**Lab 4: ABEND0C4 Dump

- As described previously, switch the Source DSNAME to: SHARE.S17555.DUMP4
- Issue command '**IP ST SYSTEM**' to determine the dump type: _____
  - This will result in dump initialization. Reply '**Y**' to the summary dump prompt.
- Issue the '**IP LIST TITLE**' command. The title for a SLIP dump contains the ID of the SLIP trap that resulted in the dump. Record the title: _____
- We can review the SLIP trap by issuing the '**IP L SLIPTRAP**' command.
  - The 'A=SVCD' parm is what told SLIP to take an SVC DUMP. 'A' is short for ACTION.
  - The 'C=xxx' parm tells SLIP to trap on the ABEND with system completion code xxx. 'C' is short for COMP. Which ABEND code was this SLIP set for? _____
- An ABEND0C4 can be issued for a variety of reasons, depending on the PIC (Program Interrupt Code). We can determine the PIC associated with this ABEND0C4 from SYSTRACE.
  - Using the '**IP SYSTRACE ALL**' command, go to the bottom of the report, and issue command: **F *RCVY PREV**

```
PR   ASID WU-Addr- Ident  CD/D PSW----- Address- Unique-1 Unique-2

0001 001A 008FF1C0 *RCVY  PROG          940C4000 00000004 00000000
```

  - Fill in the ABEND code below based on this *RCVY entry:
    - *RCVY PROG 94 __ __ __ 000 (fill in the 3 missing characters)
  - The PIC associated with the ABEND0C4 is in 'Unique-1'. Record the PIC: _____
  - Record the failing ASID _____ and TCB (WU_Addr) _____.
    - Use the '**IP SELECT ALL**' command to determine the jobname: _____
  - On which processor (PR) did this ABEND0C4 occur? _____
- Next, look for the previous PGM trace entry in SYSTRACE for this same processor, ASID, and TCB:

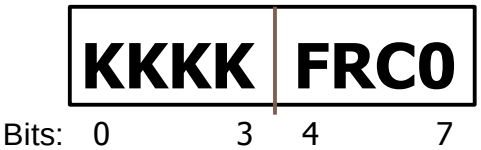
```
PR   ASID WU-Addr- Ident  CD/D PSW----- Address- Unique-1 Unique-2

0001 001A 008FF1C0  PGM    004 00000000_259000AE 00040004 00000000
                               07850000 80000000          00FBA408
```

(Note that PGM 004 represents a PIC 4).

  - At which PSW address did the PGM 004 occur? _____
    - Issue '**IP WHERE 259000AE**' to determine the failing module (and hex offset): _____
  - The first half (left 2 bytes) of 'Unique-1' contains the failing instruction length (ILC).
    - Note the ILC value: _____
  - The second half (right 2 bytes) of 'Unique-1' contains the failing PIC:
    - Note the PIC value: _____

**Lab 4: ABEND0C4 Dump (cont)

- An ABEND0C4 PIC4 occurs when there is a protection exception most commonly due to:
  - Key-controlled protection. Every page of storage is associated with a key. The PSW key must match the storage key (or PSW key = 0) when writing to storage or reading from fetch-protected storage.
  - Low address protection. The PSA (Prefix Save Area) is a processor-related save area that begins at address 0 for a length of 2 pages (x'2000' bytes). PSA storage addresses in the range 0 – x'1FF' and x'1000' – x'11FF' are write-protected.
- We determined above (from SYSTRACE) that the PSW address associated with the PIC4 was x'259000AE'. In the case of a PIC 4 (ABEND0C4), the PSW points "after" the failing instruction. To get the failing instruction address, we need to back up from the PSW address by the number of bytes of the Instruction length (ILC) that we found earlier.
  - What is the address of the failing instruction? _____
- Now, issue '**IP LIST 259000AA LEN(4) I**' to display the failing instruction at this address. Record the failing instruction: _____
  - The instruction is attempting to STORE the contents of GR 3 into the storage pointed to by GR 4 plus a displacement and index of 0.
    - Are we trying to "read" from or "write" to storage? _____
  - When writing to storage, our PSW key must either be 0 (zero) or must be equal to the key of the storage we are writing to.
- Now, issue '**IP ST REGS**' to display the registers and PSW information at the time of the failure. Note that the ST REGS report also breaks down some of the PSW fields, including the PSW key. (Refer to page 28 in Appendix for more info about fields in ST REGS)
  - Note that the failing PSW address, failing ASID, and failing TCB match what we found from SYSTRACE earlier.
  - Below the PSW is a breakdown of some of the key PSW fields.
    - What is the PSW key? _____
  - Review the General Purpose Registers (GRs) to determine the contents of GR 3 and GR 4. Record them here.
    - GR 3 _____ GR 4 _____
  - Recall that we were trying to STORE the contents of GR 3 into the storage pointed to by GR 4.
    - Since we are running PSW key 8, the storage pointed to by GR 4 must be key 8.
  - The key associated with a page of storage is mapped as follows:



KKKK = Key
F = Fetch-Protection Bit
R = Reference Bit
C = Change Bit
0 = Reserved

**Lab 4: ABEND0C4 Dump (cont)

- We can display the storage key by using the IP LIST command with the DISPLAY parm. Issue '**IP LIST 00FBA000 DISPLAY**' to list the storage pointed to by GR 4 with its key.

```
LIST FBA000. ASID(X'001A') LENGTH(X'04') AREA
ASID(X'001A') ADDRESS(FBA000.) KEY(00)
00FBA000. C1E2C3C2                              |ASCB            |
```

  - Note the KEY of storage at address x'FBA000' is displayed as KEY(00).
    - What is the key of this storage? _____
    - Is this storage fetch-protected? _____
- Based on what we found, the reason for the ABEND0C4 PIC 4 (protection exception) is now understood. ABDPROG+x'AA' was running key 8 and tried to store data into key 0 storage that contained an eyecatcher of 'ASCB'. An ASCB is a system control block that represents an address space. The next step would be to review the ABDPROG code to determine how GR 4 gets set and why it points to an ASCB.

ANSWERS

Lab 1: Dump of Hung Job
Answers

- Issue command: **ST SYSTEM**
  - What type of dump is this? ___**SVCDUMP**_____
    (Hint: Program Producing Dump)
- Now issue '**IP L TITLE**' on the Command line
  - What is the dump title? ___**SHRENQ2 JOB HUNG**___
    - System generated dumps typically have a COMPID= and other system generated information depending on the recovery routine that takes the dump.
    - Console dumps will have a title of whatever the user puts in COMM= as the dump title.
    - Dumps taken as a result of a slip trap have SLIP trap id in them.
    - Any program can issue an SDUMP macro and generate a title of its choosing
  - Based on the title of this dump, make a guess as to which type of dump this is:
    - STANDALONE DUMP
    - **A CONSOLE DUMP**
    - SLIP TRAP GENERATED DUMP
    - SYSTEM GENERATED DUMP
  - Does the dump title give a possible indication about which job is hung? __**YES**___
    - If so, which job? __**SHRENQ2**___
- Issue command: **IP CBF CVT**
  - What is the z/OS release level on which this dump was taken? This is useful because some of the IPCS commands require the z/OS release level of IPCS to match the z/OS release level of the dump.
    (Hint: Issue command: **F PRODI**)
    - HBB7780 (z/OS V1R13)
    - **HBB7790 (z/OS V2R1)**
    - HBB77A0 (z/OS V2R2)
- On the Command line, issue '**IP CBF RTCT**' to identify which address spaces were dumped. Then, issue command '**F SDAS**'. The ASIDs dumped are listed under the SDAS heading.
  - Which ASIDs were dumped? __**x'29, x'2A'**____
- Now, issue '**IP SELECT ALL**' to correlate the ASIDs identified above with jobnames.
  - What jobname is associated with ASID x'29'? ___**SHRENQ1**____
  - What jobname is associated with ASID x'2A'? ___**SHRENQ2**____

Lab 1: Dump of Hung Job (cont)
Answers

- On the Command line, issue command: **IP ST WORKSHEET**
  - Notice that the Dump Title is also listed in this report. This is followed by the Date ____**07/25/2015**_____ and Time __ **10:55:57.490798 Local**___ of the dump and the Original dump dataset _____**SYS1.DUMP14**_____.
  - Identify the name of the system. ___**SY1**_____
    (Hint: Issue command: **F SNAME**)
  - Issue command '**F CSD**' to get to CPU information. How many active CPUs? _**1**_
    (Hint: Look at #CPUs for Alive Mask)
  - Note from the bit masks that the active CPU is a general CPU (CP), not a specialty zIIP or zAAP CPU. From the PROCESSOR RELATED DATA, the CPU number is _**0001**_.
- Now, let's determine which SDATA options were requested with the dump. This may be important to verify that the storage required to diagnose a problem was requested. Issue the command '**IP CBF RTCT+9C? STR(SDUMP) VIEW(FLAGS)**'. Then issue '**F SDUSDATA'** to get to SDUSDATA flags.
  - Was LSQA requested on the dump? __**YES**____
  - Was RGN requested (shown as rgn-private area)? __**YES**____
- The SDUEXIT flags in the output above indicate whether certain component exits received control for this dump. Issue '**F SDUEXIT**' to get to these flags. Was GRSQ specified? _**YES**_
- Recall from the dump title that job SHRENQ2 is hung. One possible reason for a job to be hung is resource contention. Issue command '**IP ANALYZE RESOURCE**' to display resource contention. Then, issue command '**F SHRENQ2**' to check whether job SHRENQ2 is waiting for a resource? __**YES**____
  - What is the MAJOR name of the ENQ? __**SYSZTEST**___
  - What is the MINOR name of the ENQ? ___**S17555NQ**_____
  - What is the SCOPE of the ENQ? ___**SYSTEMS**_____
  - Which job holds this ENQ? __**SHRENQ1**_____
- Now, the question becomes why job SHRENQ1 is not releasing this ENQ, which is required by job SHRENQ2. Issue command '**IP VERBX MTRACE**' to display SYSLOG messages just before the dump was taken. Issue command '**F SHRENQ1**' to look for any messages pertaining to SHRENQ1.
  - What time was the SHRENQ1 job started (S SHRENQ1)? ___**10:53:55.16**____
  - Check for subsequent messages to determine what the SHRENQ1 job did after it started.
    - Is there anything that could be done on the console to relieve this problem?
      SHRENQ1 issued a WTOR indicating it is stuck waiting for a reply. It is reasonable to assume that a reply would eliminate the hang and release the ENQ.

      ***03 COMPLETED ENQ. REPLY TO CONTINUE.**

Lab 2: Dump of Looping Job
Answers

- Switch the Source DSNAME by typing '**=0**' (zero) on the IPCS Command Line.
  - Change the DSNAME to 'SHARE.S17555.DUMP2' and the SCOPE to BOTH, and then hit <ENTER>.
- Issue command '**IP ST SYSTEM**'
  - Note the dump type: ___**SVCDUMP**_____
- Then, issue command **IP ST WORKSHEET**
  - Note the dump title: _____**LOOPING JOB**_____
  - How many CPUs are active on the system? __**1**__
    - Which CPU # is active? __**0001**___
- Issue '**IP CBF RTCT**' to identify which address spaces were dumped. Then, issue command '**F SDAS**'. The ASIDs dumped are listed under the SDAS heading.
  - Which ASIDs were dumped? ____**1A**_____
- To correlate the ASID identified above with its jobname, issue: **IP SELECT ASID(x'1A')**
  - What is the jobname? _____**LOOPER**_____
- Issue '**IP SYSTRACE ALL TI(LO)**' and check for any processor WAIT trace entries. Do you find any? _**NO**_
  (Hint: Issue command: **F WAIT 21**)
  (Refer to page 29 in Appendix for more info about fields in SYSTRACE report)
  - The absence of any processor WAIT entries is indicative that the system is busy running at 100% CPU.
- Since this dump was supposedly taken of a looping job, it's reasonable to assume that the looping job is ASID(x'1A'). Issue command: **IP SYSTRACE ASID(x'1A') TI(LO)**
  - Do you see any pattern with the trace entries? ___**YES**___
  - What pattern do you see?
    (Note: The PC # is in the UNIQUE-2 field of the PC trace entry)
    _**some code is repeatedly issuing PC 30B Storage Obtain and PC 311 Storage**___
    _**Release requests.**_____

Lab 2: Dump of Looping Job (cont)
Answers

- What is the PSW address of the PC 30B (Storage Obtain)? ___**25900040**_____
  - Issue '**IP WHERE 25900040**' to determine the issuing module (and hex offset) __**LOOPER+40**_____
    - The PSW address actually points after the Storage Obtain PC. To determine the true address of the PC, we need to subtract the length of a PC instruction (4 bytes) from the PSW address: x'2590003C' (LOOPER+3C)
- What is the PSW address of the PC 311 (Storage Release)? ___**2590006E**_____
  - Issue '**IP WHERE 2590006E**' to determine the issuing module (and hex offset) __**LOOPER+6E**_____
    - The PSW address actually points after the Storage Release PC. To determine the true address of the PC, we need to subtract the length of a PC instruction (4 bytes) from the PSW address: x'2590006A' (LOOPER+6A)

- It looks like the LOOPER module is in a loop on TCB(x'8F8238') obtaining and releasing storage. The SSRV 132 trace entry provides more details about the Storage Obtain, and the SSRV 133 trace entry provides more details about the Storage Release. (Note: the contents of the SSRV trace entries are described in z/OS MVS Diagnosis: Tools and Service Aids, Chapter 8).

```
0001 001A 008F8238  SSRV   132    00000000  00000672 00005FB8 25901000
                                             001A0000
0001 001A 008F8238  SSRV   133    00000000  00000603 00005FB8 25901000
                                             001A0000
```

  - The UNIQUE-3 field (in bold above) on the SSRV 132 and SSRV 133 trace entries contains the storage address being obtained/released.
  - Examine the SYSTRACE entries more closely for ASID(x'1A'). Is the code obtaining and releasing different storage areas or the same storage area?
    ____**the code is obtaining and releasing the same storage address x'25901000'**____

- Based on what we've determined, the LOOPER code appears to be issuing a Storage Obtain at offset x'3C' and then looping back after it issues the Storage Release at offset x'6A'.
  - Issue the following command to review the Storage Release PC and subsequent instructions: **IP LIST 2590006A LEN(x'20') I**
    (The 'I' operand interprets the storage as Assembler instructions. The Assembler instructions are displayed to the right, with the associated machine code displayed to the left.)
    - Which instruction is at address x'2590006E'? ___**A7F4 FFD6 BRC X'F',*-X'54'**___
  - Thus, LOOPER is branching backwards (Branch Relative on Condition) x'54' bytes right after the Storage Release. Hence, the loop!

Lab 3: ABEND0C1 Dump
Answers

- As described previously, switch the Source DSNAME to: SHARE.17555.DUMP3
- Issue command '**IP ST SYSTEM**' to determine the dump type: ___**SYSMDUMP**____
  - This will result in dump initialization. Reply '**Y**' to the summary dump prompt.
- Determine the reason for this dump by issuing the '**IP LIST TITLE**' command. From the output, is there any indication that this dump was the result of an ABEND? If so, what ABEND? __**0C1**___.
  - An ABEND0C1 occurs following a PIC 1 (Program Interrupt Code 1) when the CPU attempts to execute an instruction with an invalid operation code (opcode).
- Issue '**IP SYSTRACE ALL TI(LO)**', and then go to the bottom of the report by typing '**M**' on the command line followed by **<PF8>** key. Once at the bottom, issue command: **F *PGM PREV**

```
 PR   ASID WU-Addr- Ident  CD/D PSW----- Address- Unique-1 Unique-2

0001 001A 008FF1C0 *PGM    001 00000000 259000B4 00060001 00000000
                               07850000 80000000           7F6DA400
```

(Note that PGM 001 represents a PIC 1).

- Record the failing ASID __**1A**___ and TCB (WU_Addr) ___**8FF1C0**_____.
  - Use the '**IP SELECT ALL**' command to determine the jobname: _**BADPROG1**_
- At which PSW address did the PGM 001 occur? _____**259000B4**_____
  (Note: The PGM entry displays the 128-bit (4 word) PSW on 2 lines, but not in the order of bit 0 to bit 127. The instruction address (2 words) is displayed first, followed on the second line by the first half (2 words) of the PSW).
    - Issue '**IP WHERE 259000B4**' to determine the failing module (and hex offset): _____**ABEND0C1+B4**_____
- The first half (left 2 bytes) of 'Unique-1' contains the failing instruction length (ILC).
  - Note the ILC value: __**0006**_
- The second half (right 2 bytes) of 'Unique-1' contains the failing PIC:
  - Note the PIC value: _**0001**__

- Fill in the ABEND code in the *RCVY entry below, based on the *RCVY entry that immediately follows the *PGM 001 entry in SYSTRACE:
  - *RCVY PROG 94 _**0**_ _**C**_ _**1**_ 000 (fill in the 3 missing characters)
- Use the '**IP ST FAILDATA**' command to get the relevant information about the ABEND0C1. Issue **F 'TIME OF ERROR'** to get to the relevant section.
  - Record the following:
    - PSW address (right 2 words) ____**00000000 259000B4**____
    - Instruction length (ILC) __**06**___ and Interrupt code _**0001**__
    - Failing instruction text ____**C9D5E5C1 D3C9C440 C9D5E2E3**_____
    - Breaking event address (BEAR) _____**00000000_259000B4**_____

Lab 3: ABEND0C1 Dump (cont)
Answers

- Following the 'Breaking event address' are the registers at time of error. The registers are formatted as AR (Access Register) / GR (General Register) pairs. Note that each AR is 1 word (32 bits) in size, and each GR is 2 words (64-bits) in size.
  - What are the values of GR 14 __**A59000A6**__ and GR 15 ___**259000AE**___?
- Below the registers are the Home, Primary, and Secondary ASIDs.
  - Note the Home ASID: ____**1A**_____
- Then, the report formats "This Task's ASID/TCB:"
  - Record the ASID __**1A**____ and TCB ___**8FF1C0**_____.
- Take notice that the information we found from ST FAILDATA matches the information we found from SYSTRACE. This information is built by RTM (Recovery Termination Manage) and gets stored in an SDWA (System Diagnostic Work Area) that gets passed to recovery routines.

- Now, that we have information from the time of the ABEND0C1, we need to determine why we encountered the PIC 1. The most common reasons for a PIC 1 are a wild branch or overlaid code.
  - Let's examine the "Failing instruction text" more closely. The PSW address always points 6 bytes into the failing instruction text. In other words, this text contains 6 bytes before the PSW, and 6 bytes after. The specific PIC determines whether the PSW points "at" or "after" the failing instruction.
    - In the case of a PIC 1 (ABEND0C1), the PSW points "after" the failing instruction. We need to back up by the number of bytes of the Instruction length (ILC) that we found earlier. Record the failing instruction: _____**C9D5E5C1D3C9**_____
    - This looks like EBCDIC data! Let's confirm by LISTing the storage. We get the failing instruction address by taking the PSW address (x'259000B4') and subtracting 6 bytes (ILC value): **IP LIST 259000AE LEN(6)**
      - Record the EBCDIC translation you see between the bars | | on the right: _____**INVALI**_____
  - We found above that we tried to execute EBCDIC code at address x'259000AE'. So, how did we get here?
    - A common branching technique is a BALR 14,15 where a program branches from the code pointed to by GR 14 to the code pointed to by GR 15.
    - Look at the value of GR 15 that we recorded earlier (from ST FAILDATA). Does it point to our failing instruction address? __**YES**___
    - Now look at the value of GR 14 that we recorded earlier. A BALR is a 2-byte instruction. If a BALR was done, GR 14 would point to the instruction after the BALR. So, if we subtract 2 bytes from the GR 14 address, and turn off the high-order addressing mode bit, we end up with an address of ____**259000A4**_____.

Lab 3: ABEND0C1 Dump (cont)
Answers

- Now, the Breaking Event Address (BEAR) that we recorded earlier from ST FAILDATA comes into play. BEAR is a 64-bit register (filled in by hardware following a program interrupt) containing the address of the last instruction causing a break in sequential execution of code (last branch). Does this match the address we derived above from GR 14? __**YES**___
  - As you can see, the BEAR is very useful for debugging ABEND0C1's, especially those due to a wild branch. It can help eliminate guesswork from the picture.
- Lastly, let's confirm that there really is a BALR instruction (2 bytes) at address x'259000A4'. Issue '**IP LIST 259000A4 LEN(2)**
  - Record the 2 bytes seen: __**05EF**___
  - Issue command '**IP OPCODE 05EF**' to determine the opcode (mnemonic) associated with this instruction? Is it a BALR? ___**YES**____
- Issue '**IP WHERE 259000A4**' to determine who issued this BALR to an invalid instruction. ____**ABEND0C1+A4**_____

o The next step would be to review the Assembler code listing for this ABEND0C1 program to determine why it branched to an invalid instruction at offset x'A4'.

Lab 4: ABEND0C4 Dump
Answers

- As described previously, switch the Source DSNAME to: SHARE.S17555.DUMP4
- Issue command '**IP ST SYSTEM**' to determine the dump type: __**SLIPDUMP**___
  - This will result in dump initialization. Reply '**Y**' to the summary dump prompt.
- Issue the '**IP LIST TITLE**' command. The title for a SLIP dump contains the ID of the SLIP trap that resulted in the dump. Record the title: _____**SLIP DUMP ID=0002**_____
- We can review the SLIP trap by issuing the '**IP L SLIPTRAP**' command.
  - The 'A=SVCD' parm is what told SLIP to take an SVC DUMP. 'A' is short for ACTION.
  - The 'C=xxx' parm tells SLIP to trap on the ABEND with system completion code xxx. 'C' is short for COMP. Which ABEND code was this SLIP set for? __**0C4**___
- An ABEND0C4 can be issued for a variety of reasons, depending on the PIC (Program Interrupt Code). We can determine the PIC associated with this ABEND0C4 from SYSTRACE.
  - Using the '**IP SYSTRACE ALL**' command, go to the bottom of the report, and issue command: **F *RCVY PREV**

```
PR   ASID WU-Addr- Ident  CD/D PSW----- Address- Unique-1 Unique-2

0001 001A 008FF1C0 *RCVY  PROG          940C4000 00000004 00000000
```

  - Fill in the ABEND code below based on this *RCVY entry:
    - *RCVY PROG 94 _**0**_ _**C**_ _**4**_ 000 (fill in the 3 missing characters)
  - The PIC associated with the ABEND0C4 is in 'Unique-1'. Record the PIC: _**00000004**_
  - Record the failing ASID _**1A**__ and TCB (WU_Addr) __**8FF1C0**___.
    - Use the '**IP SELECT ALL**' command to determine the jobname: __**TESTJOB**__
  - On which processor (PR) did this ABEND0C4 occur? _**0001**___
- Next, look for the previous PGM trace entry in SYSTRACE for this same processor, ASID, and TCB:

```
PR   ASID WU-Addr- Ident  CD/D PSW----- Address- Unique-1 Unique-2

0001 001A 008FF1C0  PGM    004 00000000_259000AE 00040004 00000000
                               07850000 80000000           00FBA408
```

(Note that PGM 004 represents a PIC 4).

  - At which PSW address did the PGM 004 occur? _____**259000AE**_____
    - Issue '**IP WHERE 259000AE**' to determine the failing module (and hex offset): ____**ABDPROG+AE**_____
  - The first half (left 2 bytes) of 'Unique-1' contains the failing instruction length (ILC).
    - Note the ILC value: __**0004**__
  - The second half (right 2 bytes) of 'Unique-1' contains the failing PIC:
    - Note the PIC value: __**0004**___

Lab 4: ABEND0C4 Dump (cont)
Answers

- An ABEND0C4 PIC4 occurs when there is a protection exception most commonly due to:
  - Key-controlled protection. Every page of storage is associated with a key. The PSW key must match the storage key (or PSW key = 0) when writing to storage or reading from fetch-protected storage.
  - Low address protection. The PSA (Prefix Save Area) is a processor-related save area that begins at address 0 for a length of 2 pages (x'2000' bytes). PSA storage addresses in the range 0 – x'1FF' and x'1000' – x'11FF' are write-protected.
- We determined above (from SYSTRACE) that the PSW address associated with the PIC4 was x'259000AE'. In the case of a PIC 4 (ABEND0C4), the PSW points "after" the failing instruction. To get the failing instruction address, we need to back up from the PSW address by the number of bytes of the Instruction length (ILC) that we found earlier.
  - What is the address of the failing instruction? ___**259000AA**___
- Now, issue '**IP LIST 259000AA LEN(4) I**' to display the failing instruction at this address. Record the failing instruction: _____**5034 0000 | ST R3,X'0'(R4)**_____
  - The instruction is attempting to STORE the contents of GR 3 into the storage pointed to by GR 4 plus a displacement and index of 0.
    - Are we trying to "read" from or "write" to storage? __**write**___
  - When writing to storage, our PSW key must either be 0 (zero) or must be equal to the key of the storage we are writing to.
- Now, issue '**IP ST REGS**' to display the registers and PSW information at the time of the failure. Note that the ST REGS report also breaks down some of the PSW fields, including the PSW key. (Refer to page 28 in Appendix for more info about fields in ST REGS)
  - Note that the failing PSW address, failing ASID, and failing TCB match what we found from SYSTRACE earlier.
  - Below the PSW is a breakdown of some of the key PSW fields.
    - What is the PSW key? __**key 8**____
  - Review the General Purpose Registers (GRs) to determine the contents of GR 3 and GR 4. Record them here.
    - GR 3 __**00000000**_ GR 4 ___**00FBA000**___
  - Recall that we were trying to STORE the contents of GR 3 into the storage pointed to by GR 4.
    - Since we are running PSW key 8, the storage pointed to by GR 4 must be key 8.
  - The key associated with a page of storage is mapped as follows:

KKKK = Key
F = Fetch-Protection Bit
R = Reference Bit
C = Change Bit
0 = Reserved

Lab 4: ABEND0C4 Dump (cont)
Answers

- We can display the storage key by using the IP LIST command with the DISPLAY parm. Issue '**IP LIST 00FBA000 DISPLAY**' to list the storage pointed to by GR 4 with its key.

```
LIST FBA000. ASID(X'001A') LENGTH(X'04') AREA
ASID(X'001A') ADDRESS(FBA000.) KEY(00)
00FBA000. C1E2C3C2                              |ASCB            |
```

  - Note the KEY of storage at address x'FBA000' is displayed as KEY(00).
    - What is the key of this storage? __**0**___
    - Is this storage fetch-protected? __**NO**____
- Based on what we found, the reason for the ABEND0C4 PIC 4 (protection exception) is now understood. ABDPROG+x'AA' was running key 8 and tried to store data into key 0 storage that contained an eyecatcher of 'ASCB'. An ASCB is a system control block that represents an address space. The next step would be to review the ABDPROG code to determine how GR 4 gets set and why it points to an ASCB.

APPENDIX

IPCS Option 1 - Browse Mode

- **Browsing Storage:** IPCS Option 1 (=1 on any IPCS command line) brings up the following panel:

```
------------------------ IPCS - ENTRY PANEL  ---------------------

 CURRENT DEFAULTS:
  Source ==> DSNAME('SHARE.S17555.DUMP1')
  Address space ==> ASID(X'0001')

 OVERRIDE DEFAULTS:                (defaults used for blank fields)
  Source ==> DSNAME('SHARE.S17555.DUMP1')
  Address space ==>
  Password      ==>

 POINTER:
  Address       ==>                 (blank to display pointer stack)
  Remark        ==>                                  (optional text)
```

- Hit <ENTER> on screen above to get to pointer stack on next screen:

```
DSNAME('SHARE.S17555.DUMP1') POINTERS
--------------------------------------
ASID(X'0001') is the default address space
PTR   Address   Address space               Data type
s0001 00000000  ASID(X'0001')              AREA
      Remarks:
**************** END OF POINTER STACK ************************
```

- Type an 's' under PTR to select an address you'd like to browse, or fill in the '**s**' as shown above to get to address 0, where you can use the **L** (**LOCATE**) command to display the storage you're interested in, as demonstrated below:

```
ASID(X'0001') ADDRESS(0200.) STORAGE ----------------------------------
Command ===> L 07E00F04. asid(x'65')                    SCROLL ===> CSR
00000000   000A0000  000130E1  00000000  00000000   | ..............  |
00000010   00FD8650  00000000  7FFFF000  7FFFF000   | ..f&....".0.".0. |
00000020   7FFFF000  7FFFF000  7FFFF000  7FFFF000   | ".0.".0.".0.".0. |
00000030   00000000  00000000  7FFFF000  7FFFF000   | ........".0.".0. |
```


IP ST REGS

- Displays registers at time of dump
- This report is not dependent on an SDWA (as ST FAILDATA), and is useful for both SLIP dumps and dumps generated by a system recovery routine.
  - Provides more information on the PSW, but the Failing Instruction Text, Instruction Length, and Interrupt Code are not provided (as with ST FAILDATA), since there is no SDWA.

```
CPU STATUS:
PSW=470C6000 95508DAE
   (Running in AR, key 0, AMODE 31, DAT ON)
    ENABLED FOR PER I/O EXT MCH
  ASID(X'000E') 15508DAE. BPXINPVT+103DAE IN EXTENDED PRIVATE
 ASCB261 at B98100, JOB(AFA46083), for the home ASID

 ASXB261 at 9FDE90 and TCB261G at 9A5E88 for the home ASID

 HOME ASID: 0105 PRIMARY ASID: 000E SECONDARY ASID: 0105

 General purpose register values
   Left halves of all registers contain zeros
    0-3  00000000  00000000  7F665470  81E5A828
    4-7  00000000  00000000  00000000  00000000
   8-11 155096D6  15756F00  16E55F1E  155086D7
  12-15 155076D8  16E57220  95508AFA  00000052

 Access register values
     0-3  00000000  00000000  00000000  00000000
     4-7  00000002  0101002C  00000000  00000000
    8-11 00000000  00000002  00000000  00000000
   12-15 00000000  00000000  00000000  00000000
```

PSW address (drop the high order bit), thus: 9 becomes 1

First word of PSW broken down; AR mode means Access REGs will be used!

TCB address that dump occurred under. Runs under HOME ASID

ASID where PSW address is running; If 'IP WHERE' is used to determine where PSW points, this ASID should be used.
-> IP W 15508DAE ASID(x'E')

The '**IP WHERE**' command is used to identify storage areas in the dump and offset into those areas.

- Load Module (LMOD) name
  - If externally identified member of IEANUC01, then CSECT name.
- Area of storage
  - Common service area (CSA), Fixed link pack area (FLPA), Modified link pack area (MLPA), Pageable link pack area (PLPA), Private, Prefixed save areas (PSA), Read only nucleus, Read/write nucleus, System queue area (SQA) 0

```
   IPCS OUTPUT STREAM ---------------------------- LINE 0 COLS 1 78
COMMAND ===> _                                      SCROLL ===> CSR
 ********************** TOP OF DATA ******************************
      ASID(X'001B') 01D073D0. IGC0004B+03D0 IN EXTENDED PLPA
 ********************** END OF DATA ******************************
```


IP SYSTRACE

- Displays trace of system events at time of dump, including:
  - Dispatches, interrupts, PC/PR, recovery actions.
- **IP SYSTRACE ALL TI(LO)**
  - Formats trace records for all address spaces
- **IP SYSTRACE ASID(x'nn') TI(LO)**
  - Formats trace records only associated with requested ASID
- Entries formatted in chronological order (oldest at top, newest at bottom)

```
PR   ASID WU-Addr- Ident  CD/D PSW----- Address-  Unique-1 Unique-2 Unique-3  PSACLHS-
                                                  Unique-4 Unique-5 Unique-6  PSACLHSE

0001 000A 008F91C0  SVC     2F 00000000_259464A2  25AB44AC 04000000 25AB4708  STimerM
                               07040000 80000000
0001 000A 008F91C0  SSRV    78          815AEE7A  0000F502 00000080 0259B058  Getmain
                                                  000A0000
0001 000A 008F91C0  SVCR    2F 00000000_259464A2  00000000 04000000 25AB4708
                               07040000 80000000
0001 000A 008F91C0  SVC      1 00000000_25945534  808F9120 00000001 DA54B848  Wait
                               07041000 80000000
0001 000A 008F91C0  SVCR     1 00000000_25945534  808F9120 00000001 DA54B848
                               07041000 80000000
0001 0001 00000000  WAIT
```

  - **PR** = logical CPU number
  - **ASID** = HOME ASID (address space identifier)
  - **WU-Addr** (Work Unit Address)
    - TCB mode: TCB address
    - SRB mode: Web address
    - 0's under special cases
  - **Ident** = trace entry identifier
  - **CD/D** = number related to trace entry
  - **PSW----- Address-**
    - PSW or module address
  - **Date and Time** (not displayed in above example) of associated trace entry can be found by scrolling to the right
    - PF11 - scrolls right
    - RIGHT nn - scrolls right nn columns

- Refer to z/OS MVS Diagnosis: Tools and Service Aids manual, Chapter 8, for more details on System Trace entries

