IBM Education Assistance for z/OS V2R1

Item: Zero Address Detection (ZAD)
Element/Component: BCP SLIP

Material is current as of March 2013

Agenda

- Trademarks
- Presentation Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.

Presentation Objectives

Describe Zero Address Detection (ZAD) and the SLIP support that enables it

Overview

- Problem:
  - Z196 machine introduced a powerful new feature called Zero Address Detection. It was not supported by z/OS.
- Solution:
  - z/OS 2.1 SLIP supports ZAD
- Benefit / Value:
  - Improved reliability

What is Zero Address Detection?

- The z196 machine provides a new PER function called "Zero Address Detection".
- A Zero Address Detection (ZAD) event is a PER program interrupt caused by execution of an instruction that accesses (stores or fetches) storage using an operand address formed from a general register containing zero, when the PSW PER bit is on and Zero Address Detection is enabled.
- You enable Zero Address Detection by a SLIP PER trap of type ZAD. A ZAD trap can help to detect errors that involve unintentionally referencing the PSA. All instructions are monitored for ZAD events; you can limit ZAD events only by limiting the address spaces and/or jobs for which PER is active.
- You can, however, limit the ZAD events that match your SLIP traps by using other SLIP filtering keywords.

ZAD Events

ZAD events might be errors and might not. When an instruction is executed and the conditions are met, a ZAD event occurs. It is up to you to decide if this is an error or not (as is the case for any SLIP trap match). It is obviously good to get rid of the errors. It can also be useful to get rid of the non-error "noise" events so that when you are looking for events in your own modules you expect to find none at all so that you know that any event needs to be dealt with.

ZAD non-error events

An example of non-error:

- You are using a data space. The data space starts at page 0. You choose to use that page, and your starting origin is a pointer with a value of 0 (which you use as a base register when accessing the storage so that you can specify the ALET in the corresponding access register).
- Many components choose to avoid using page 0. In fact they consciously cause page 0 to be hidden so that any use of page 0 will program check. IBM recommends this approach for just about any program (details are on following slide).

ZAD non-error events (cont)

- You can use
  IARVSERV ChangeAccess,Target_View=HIDDEN
  to hide page 0. If using this IARVSERV invocation, be sure that you use it only when the origin returned by DSPSERV was 0.
- z/OS 2.1 introduces the HIDEZERO=YES option on DSPSERV CREATE. If you know you are running on z/OS 2.1, it is simpler to use this option than to use the “extra” IARVSERV call.

ZAD non-error events (cont)

Another example of non-error:

- Your program is looping through the PSA's of each processor. You pick up the PSA address to use from PCCAPSAV. But when doing "this CPU" you must use an address of 0. Since you don't dual-path your code, you end up with a pointer with value of 0 referencing the PSA. There are various approaches to resolving this problem which avoid dual-pathing the code, such as placing a value of "n" in the register (e.g., 8) and having your assembler USING be to "PSA+n", and for the other processors use "PCCAPSAV+n" if you want to share that USING.

ZAD non-error events (cont)

Another example of non-error:

- In assembler, you put your base and index registers "backwards", so that the base register is really the "array offset" (and might be 0) and the index register is the address of the array. The machine's address calculation will work fine, but this will produce a ZAD event when the array offset is 0. So that your own runs are "clean" within your own modules, you probably want to clean this up.

Usage & Invocation – How should you use ZAD

- We suggest that you look for ZAD events within your own products' code when you are doing testing. You are welcome to help others if you find errors in their code, but code owners might not appreciate getting flooded with information about non-error events. Thus, if you see a ZAD event outside of your scope, and you don't have good reason to believe it is an error, probably don't do anything about it.
- You should not report a ZAD event to IBM unless you have reason to believe it is an error (or if you are providing a ZAD report to help diagnosis a problem on which you are working with IBM)
- To use ZAD, you must be running on a z196. SLIP will reject the ZAD operand if it does not find ZAD available (this can happen even on a z196 if running under zVM in some cases)
- You probably will want to run this on a test system to avoid unnecessarily impacting your production workload

How should you use ZAD?

- START IEAVTSZR (this is a PROC that you will have to put into proclib; a sample proc is shown later). This time it initializes an area to record the info to be gathered
- SETPROG LPA,ADD, DSN=SYS1.LINKLIB,MOD=IEAVTSZE,FIXED (you could have used the FLPA system parameter with an IEAFIXxx parmlib member to have done this at IPL time if you prefer)
- SLIP SET,ZAD,A=AEXIT,AEXIT=IEAVTSZE,ID=ZAD1,PL=50,OK,E (AEXIT is SLIP functionality that is not supported for other use. You must use the syntax A=AEXIT,AEXIT=IEAVTSZE)
- Run your programs

How should you use ZAD? (cont)

- START IEAVTSZR (this time it writes to SYSPRINT a report about all the ZAD events and resets things, to continue)
- more iterations of running and START IEAVTSZR as appropriate
- START IEAVTSZR,OP=FREE when you're done (no need if you're re-IPLing)
- Look at the report(s) for ZAD events within the programs that you own

IEAVSZR report

- The report identifies, to the extent it can:
  - where the event occurred (both by address and by module name if available)
  - how many times it occurred
  - what instruction was issued
- You might add ASIDSA=SA to the SLIP trap if you don't want to get hits for data space stores (the ZAD event will still occur for a data space store but SLIP will filter it out). The report does flag data space stores with a "D" so you can easily ignore them even if they're in the report.

ZAD on SLIP command

- You can add further modifiers to your SLIP trap if you don't want events other than yours to appear (you can use LPAMOD, NUCMOD, ASID, ADDRESS, etc). The ZAD events will still occur, but SLIP will filter them.
- You can specify ZAD similarly where you use SA. All of the other SLIP keywords work the same as if you had specified SA but had not specified the range of the storage alteration, with the exception of the address of the instruction causing the event (hence do not use RANGE for a ZAD trap, use ADDRESS if you want to filter based on the address of the event-causing instruction).

IEAVTSZR proc

A sample IEAVTSZR proc is:

```
//*************************************************************
//** SIZE MAY BE NK, NNK, NNNK, NM, NNM, NNNM. IT IS THE AMOUNT TO *
//** BE USED FOR INDIVIDUAL INSTRUCTION DATA. IT IS ADDED TO THE   *
//** AMOUNT NEEDED TO CAPTURE DATA FOR EVERY ADDRESS SPACE.        *
//**                                                              *
//** OP=FREE TO INDICATE THAT YOU'RE DONE, OR ANYTHING            *
//** ELSE TO INDICATE                                            *
//** - ALLOCATE IF NOT YET ALLOCATED                             *
//** - PRODUCE A REPORT BASED ON THE CURRENT DATA                *
//** - AFTER PRODUCING A REPORT, CLEAR THE DATA                  *
//*************************************************************
//IEAVTSZR PROC SIZE=1M,OP=DATA,STATS=YES,SYSOUT=*
//IEAVTSZR EXEC PGM=IEAVTSZR,TIME=1440,REGION=0M,
// PARM='OP=&OP,SIZE=&SIZE,STATS=&STATS'
//SYSPRINT DD SYSOUT=&SYSOUT
```

ZAD report information

- Information provided regarding a ZAD event
  - Jobname
  - ASID in hex
  - Address of the instruction
  - Number of occurrences
  - Module name (as best can be determined); not CSECT within module
  - Offset within module (you can use AMBLIST to help locate which CSECT within the module)
  - Whether the event was in a data space or not
  - Instruction Text (hex) – e.g., `BF3F2000BF3F`
  - Decoded Instruction Text – e.g., `ICM R3,15,0(R2)`

Sample ZAD report

```
SLIP Action Exit IEAVTSZR Event Report  08/28/2012 09:10:44.70


IEAVTSZR SLIP Trap ID=ZAD1 Duration: 0000005F_6095B7E5  Seconds: 100
CP   AddrSp #: 00000000_00000004  DataSp #: 00000000_00000000
zAAP AddrSp #: 00000000_00000000  DataSp #: 00000000_00000000
zIIP AddrSp #: 00000000_00000000  DataSp #: 00000000_00000000
PerSecPerCPU: 00000000
PerSecPerCPU_CP: 00000000
PerSecPerCPU_zAAP: 00000000
PerSecPerCPU_zIIP: 00000000
Max PRCNTLIM: 0

TC       (ASID 001C):  CP AddrSp #: 00000000_00000004  DataSp #: 00000000_00000000
                      zAAP AddrSp #: 00000000_00000000  DataSp #: 00000000_00000000
                      zIIP AddrSp #: 00000000_00000000  DataSp #: 00000000_00000000

Pvt Search - Ser/Ok:             2 Ser/Fail:             0 UnSer/Ok:             0 UnSer/Fail:  0
Used Entries:             2 (  0%)  Longest Chain:             1


JobName  ASID  Address      Count  ModName    Offset  Dsp InstrucText   Decoded Instruction Text
TC       001C  00007054         2  TEST      00000054     BF3F2000BF3F  ICM     R3,15,0(R2)
TC       001C  00007058         2  TEST      00000058     BF3F2004181D  ICM     R3,15,4(R2)
```

Interactions & Dependencies

- Software Dependencies
  - None
- Hardware Dependencies
  - ZAD is available on z196 and newer machines. It is not available on z10 or older machines
- Exploiters
  - None

Appendix

- Publications:
  - MVS System Commands (SLIP) SA38-0666-00