Taking the Cross out of Cross Memory
Address Spaces, Data Spaces & Access Registers (oh my!)

NaSPA May 14th, 2019

Patty Little plittle@us.ibm.com
John Shebey jshebey@us.ibm.com

Trademarks

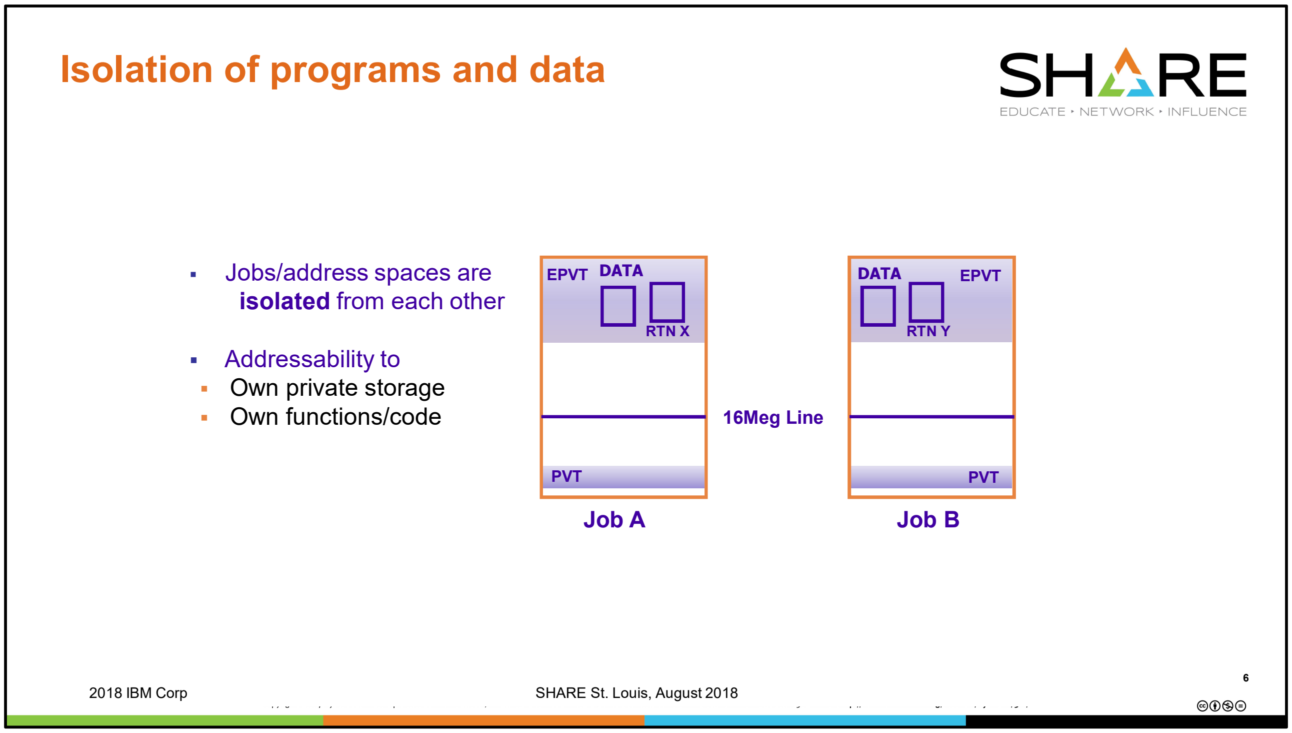
Table of Contents

- WHY BE CROSS?
  - Isolation of programs and data 6
  - Asynchronous address space communication 8
- CROSS MEMORY
  - What is cross memory? 10
  - Cross memory concepts 11
  - Simple example 13
  - Exploiting cross memory 16
  - Putting it together 17
  - Another example 18
  - Identify cross memory in dump 20
  - PC routines 21
  - PC table 22
  - PC and cross memory in systrace 24
  - Linkage Stacks 26
  - Applied example 29

Table of Contents (continued)

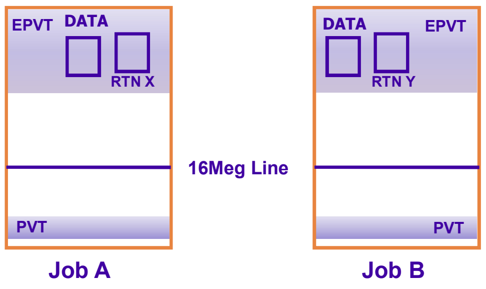
- EXTENDED ADDRESSABILITY
  - Access registers and data spaces 36
  - AR mode 37
  - Dramatization via chat 38
  - Access lists 43
  - ALETs 44
  - AR mode in dump 47
  - Access registers in dump 48
  - Translating an ALET 49
  - Data space storage in dump 50
  - Applied example 51
- APPENDIX
  - Putting it together, with control registers 58
  - Including data spaces in dump 59
  - Displaying data space storage 60
  - Identifying data spaces owned by ASID 62

WHY BE CROSS?



Isolation of programs and data

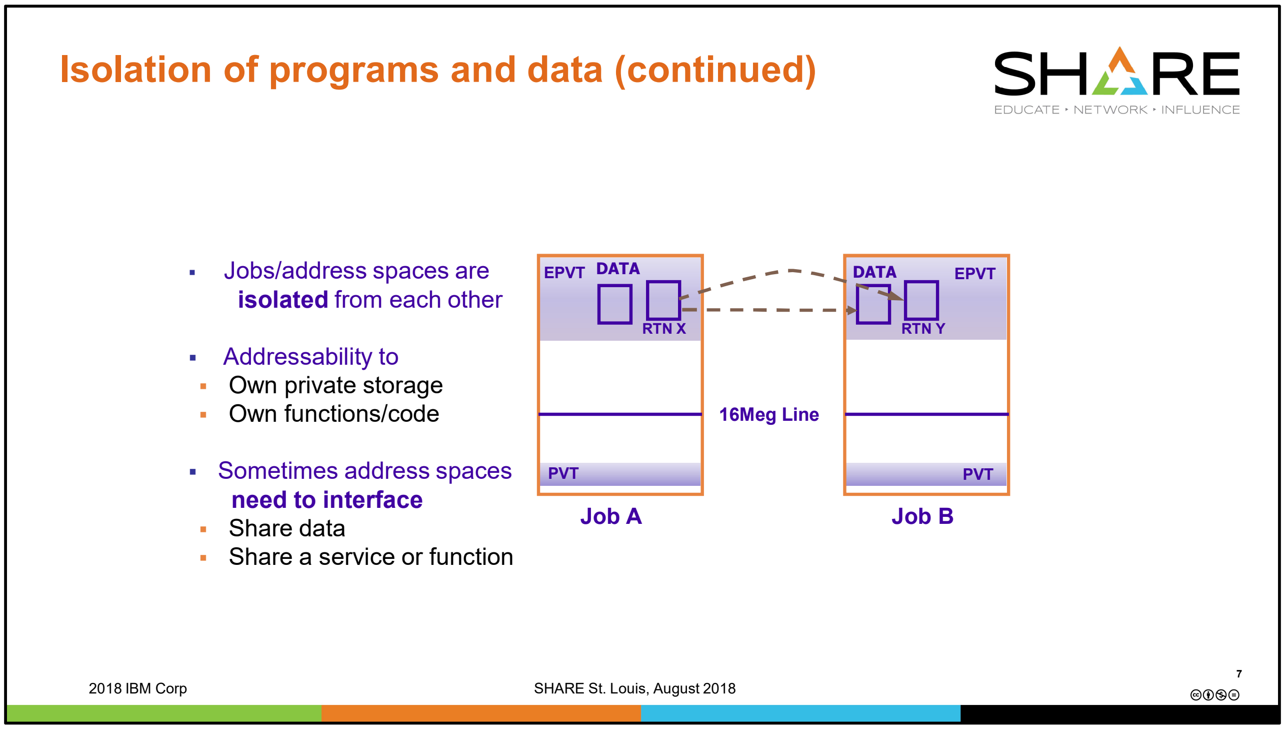
- Jobs/address spaces are **isolated** from each other
- Addressability to
  - Own private storage
  - Own functions/code



* Built into the design of z/OS is the protection that comes from isolating one address space from another. A job can only directly and easily access its own programs and data located within its private storage or in common (Nucleus and LPA, SQA and CSA). .

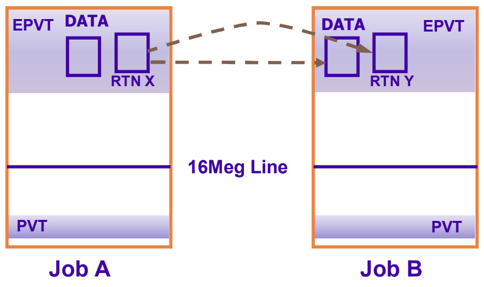
* An address space represents a function or application. Each function/application has its own data and its own code.

* The operating system intentionally isolates address spaces from each other. This is to provide protection of the address space's code and storage.



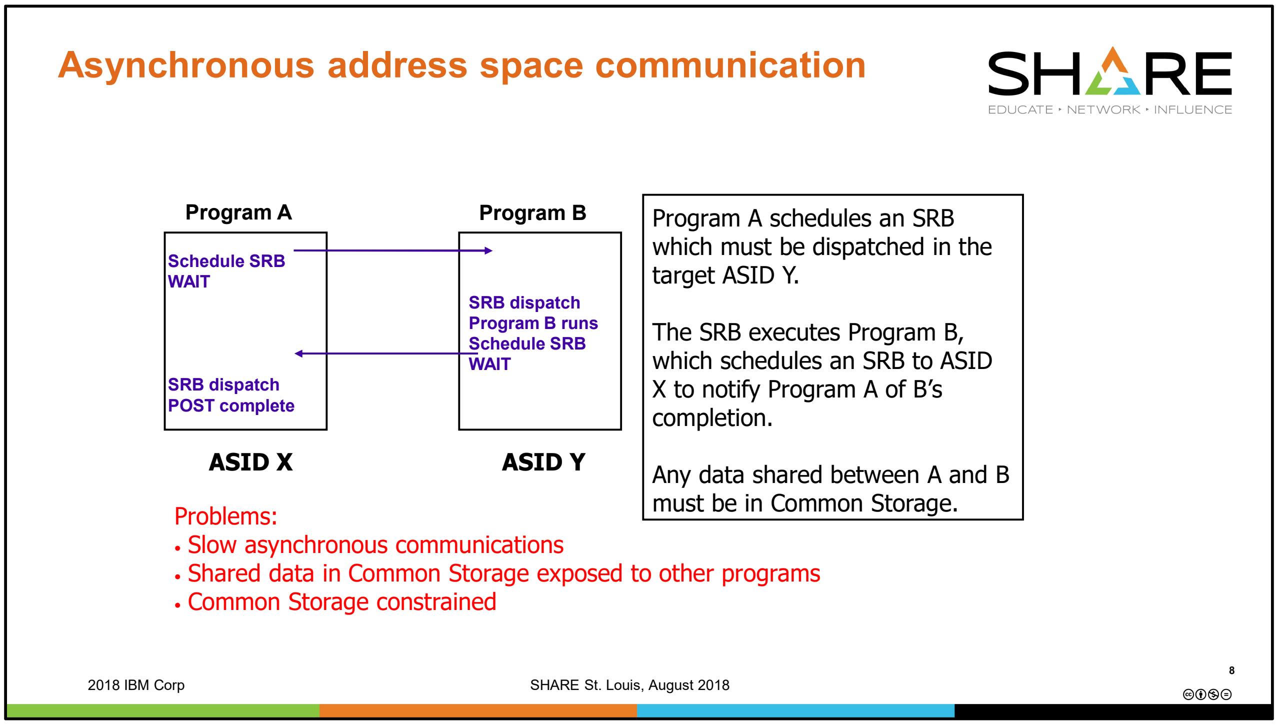
Isolation of programs and data (continued)

- Jobs/address spaces are **isolated** from each other
- Addressability to
  - Own private storage
  - Own functions/code
- Sometimes address spaces **need to interface**
  - Share data
  - Share a service or function

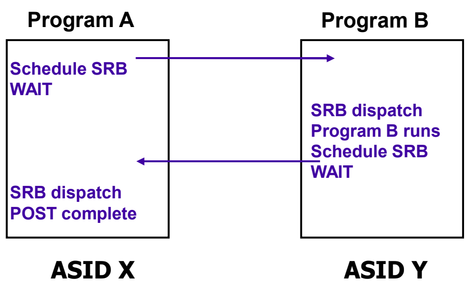


Cross memory services and instructions can be used to allow one function to access the data or code in another function. Various authorization checks must be passed before one address space can establish cross memory access to another address space.

The MVS Extended Addressability Guide is an excellent reference for discussion and examples of cross memory concepts.



Asynchronous address space communication



Program A schedules an SRB which must be dispatched in the target ASID Y.

The SRB executes Program B, which schedules an SRB to ASID X to notify Program A of B's completion.

Any data shared between A and B must be in Common Storage.

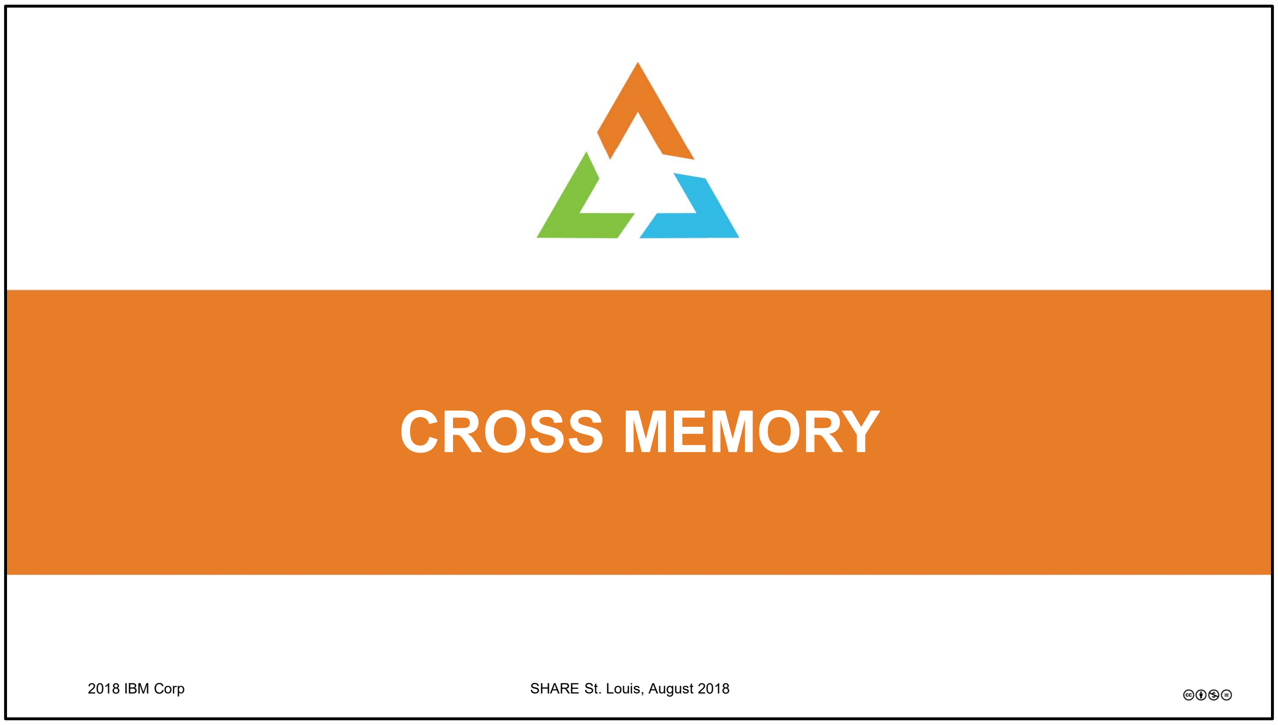
Problems:
- Slow asynchronous communications
- Shared data in Common Storage exposed to other programs
- Common Storage constrained

2018 IBM Corp SHARE St. Louis, August 2018 8

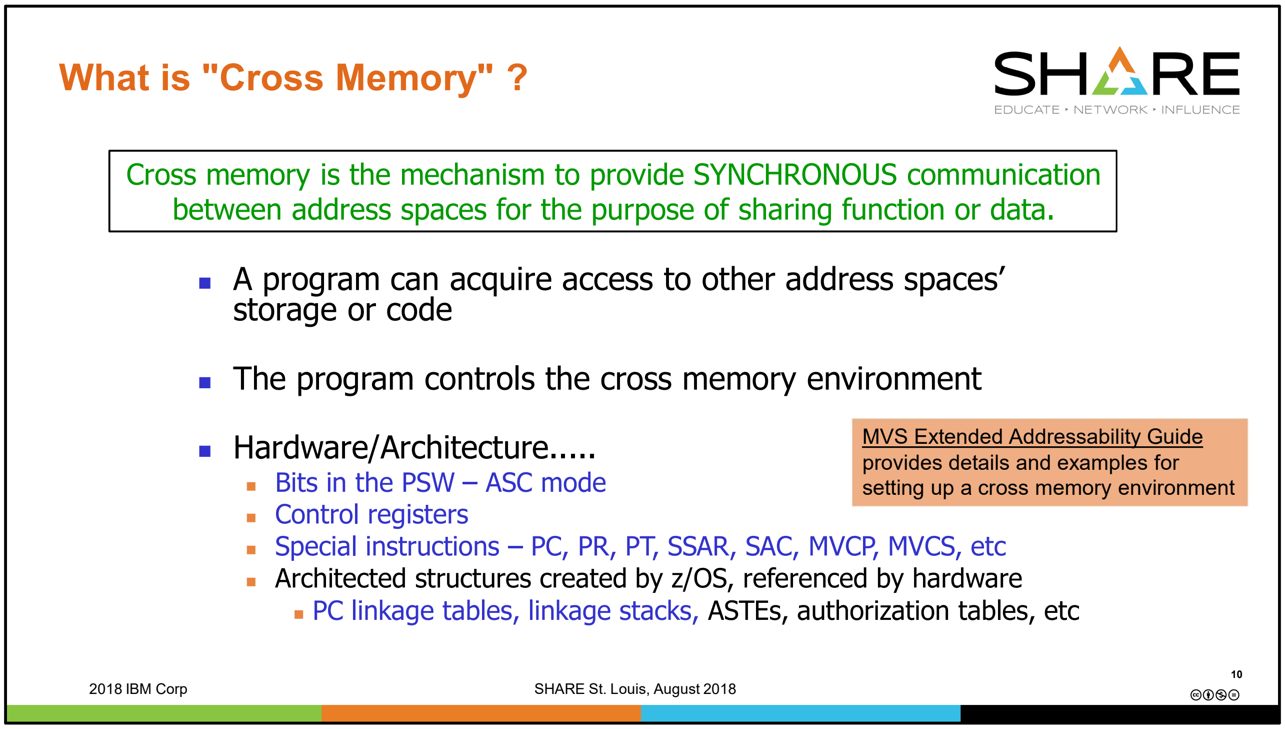
In the early days of MVS, a program could communicate with a program in another address space, but only through a cumbersome asynchronous method. If Program A, running in ASID X, needed a service or function that was performed by Program B in ASID Y, there was no way to directly call a program that resided in a different address space.

Program A solved this by scheduling an SRB to the target address space (Y) and then waiting to be notified of completion. The SRB would be queued and eventually dispatched in the target address space, ASID Y. The SRB would run program B, possibly also asynchronously by posting a task for Program B in ASID Y to run. Once Program B dispatched and ran, notification needed to be sent back to Program A. Another SRB was scheduled back to ASID X. This SRB would be queued and then dispatched, and would post Program A which was waiting for the notification. Finally, Program A, running under its task, would be dispatched and continue.

This asynchronous method required several trips through the MVS dispatcher and a wait/post mechanism to notify for completion. In addition, any data shared between the programs must be in Common Storage (Nucleus, SQA, CSA) to be accessible to both programs. In the early days of MVS especially, when only 24-bit addressing existed, Common Storage was very constrained.



CROSS MEMORY



What is "Cross Memory" ?

Cross memory is the mechanism to provide SYNCHRONOUS communication between address spaces for the purpose of sharing function or data.

- A program can acquire access to other address spaces' storage or code
- The program controls the cross memory environment
- Hardware/Architecture.....
  - Bits in the PSW – ASC mode
  - Control registers
  - Special instructions – PC, PR, PT, SSAR, SAC, MVCP, MVCS, etc
  - Architected structures created by z/OS, referenced by hardware
    - PC linkage tables, linkage stacks, ASTEs, authorization tables, etc

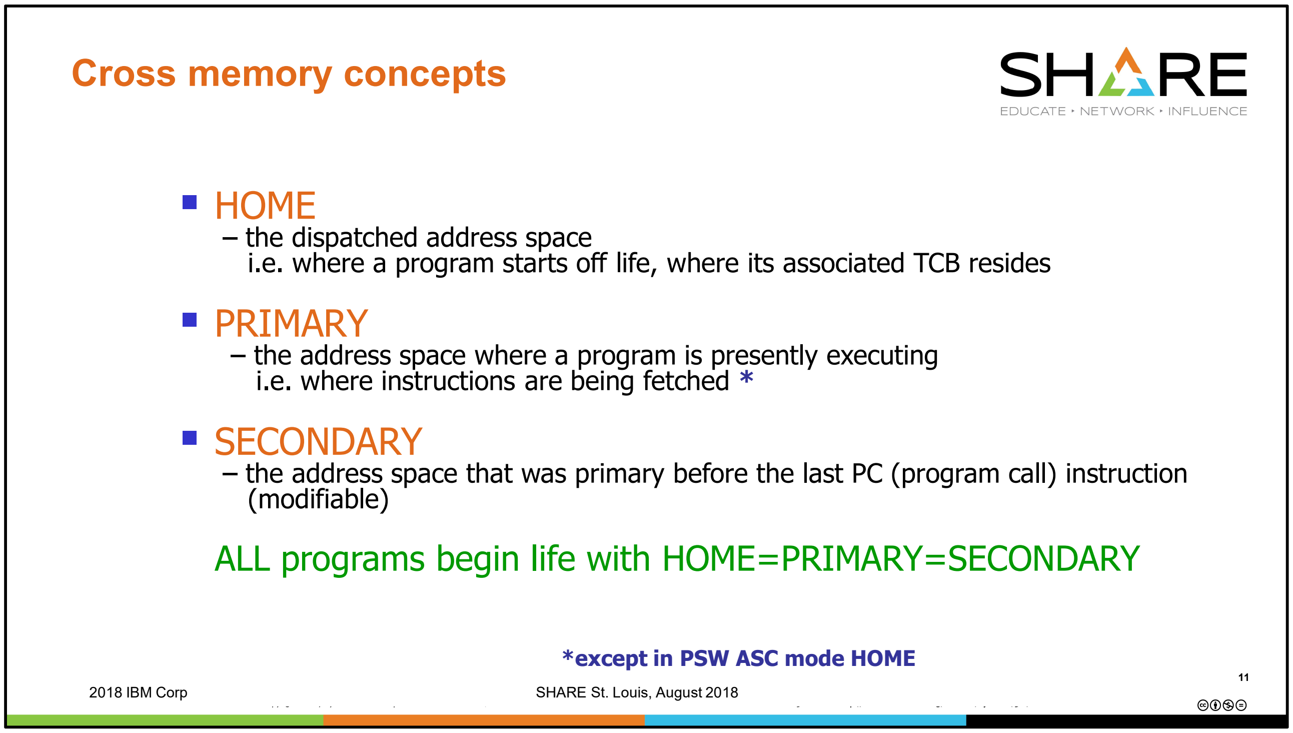
MVS Extended Addressability Guide provides details and examples for setting up a cross memory environment

2018 IBM Corp SHARE St. Louis, August 2018 10

Providing a SYNCHRONOUS mechanism to communicate between address spaces improves performance by avoiding the schedule/wait/schedule/post asynchronous communications, and allows direct access to private storage, eliminating the need to keep shared data in common storage. Data is isolated to only the programs that have access, and the common storage constraint is relieved.

The **MVS Extended Addressability Guide** is an excellent reference for discussion and examples of cross memory concepts.

Items highlighted in blue will be discussed later in the presentation.



Cross memory concepts

- HOME
  – the dispatched address space
  i.e. where a program starts off life, where its associated TCB resides
- PRIMARY
  – the address space where a program is presently executing
  i.e. where instructions are being fetched *
- SECONDARY
  – the address space that was primary before the last PC (program call) instruction (modifiable)

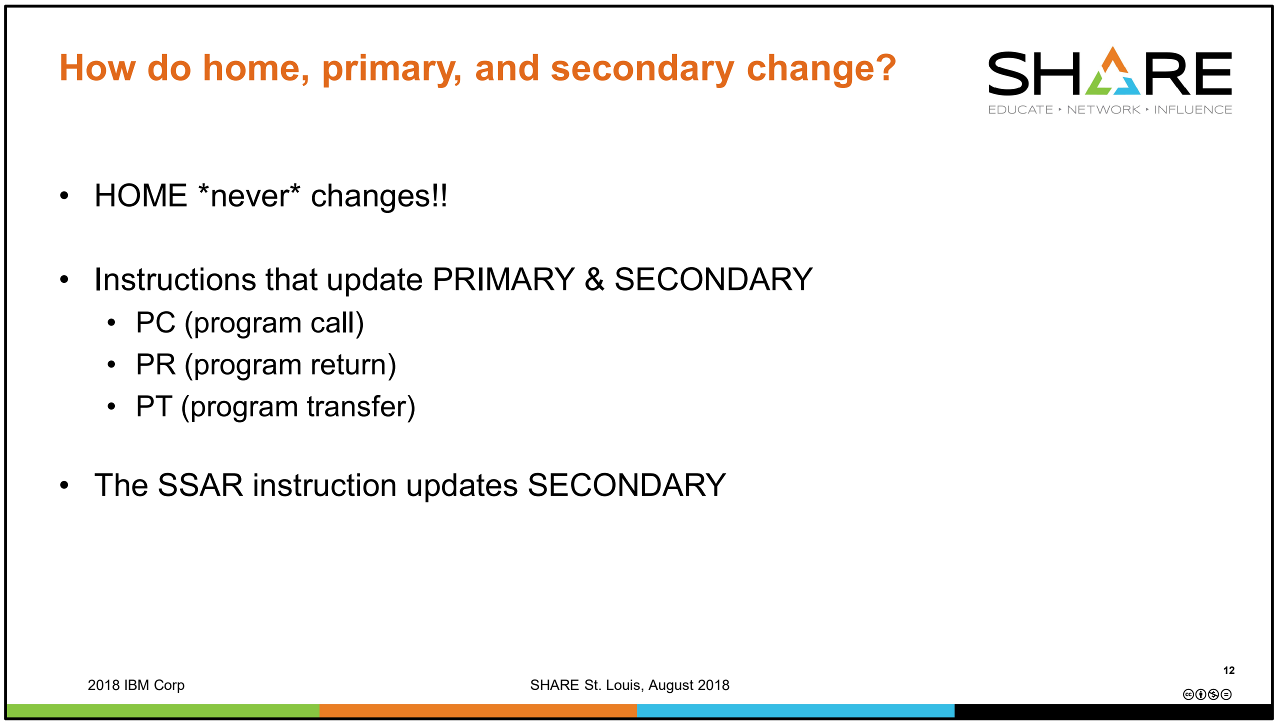
ALL programs begin life with HOME=PRIMARY=SECONDARY

***except in PSW ASC mode HOME**

2018 IBM Corp SHARE St. Louis, August 2018 11

The whole purpose of cross memory addressability is to extend a program's "reach". A program running in an environment where Home, Primary, and Secondary are all different has ready access to the code and data in 3 different address spaces. An example of this will be provided shortly.

To be discussed later, access registers offer an alternate and more powerful way of extending a program's addressability. Access registers can be used to gain access to data spaces, as well as to easily address data in other address spaces.



How do home, primary, and secondary change?

- HOME *never* changes!!
- Instructions that update PRIMARY & SECONDARY
  - PC (program call)
  - PR (program return)
  - PT (program transfer)
- The SSAR instruction updates SECONDARY

2018 IBM Corp SHARE St. Louis, August 2018 12

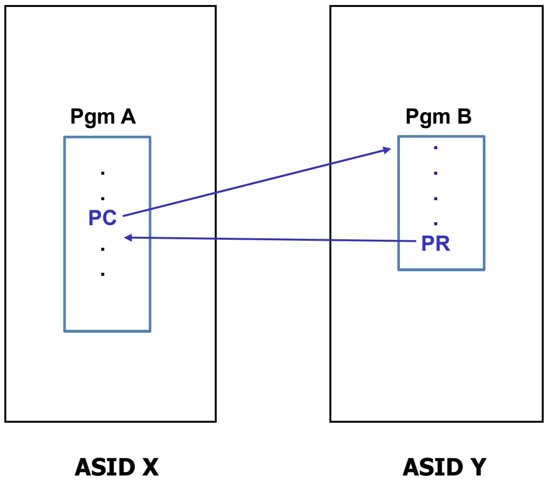
Reminder:
- Home is where a program starts life.
- Primary is where a program is presently executing.
- Secondary is typically where a program was executing prior to its last PC.
    - SSAR can be used to set secondary to something else.

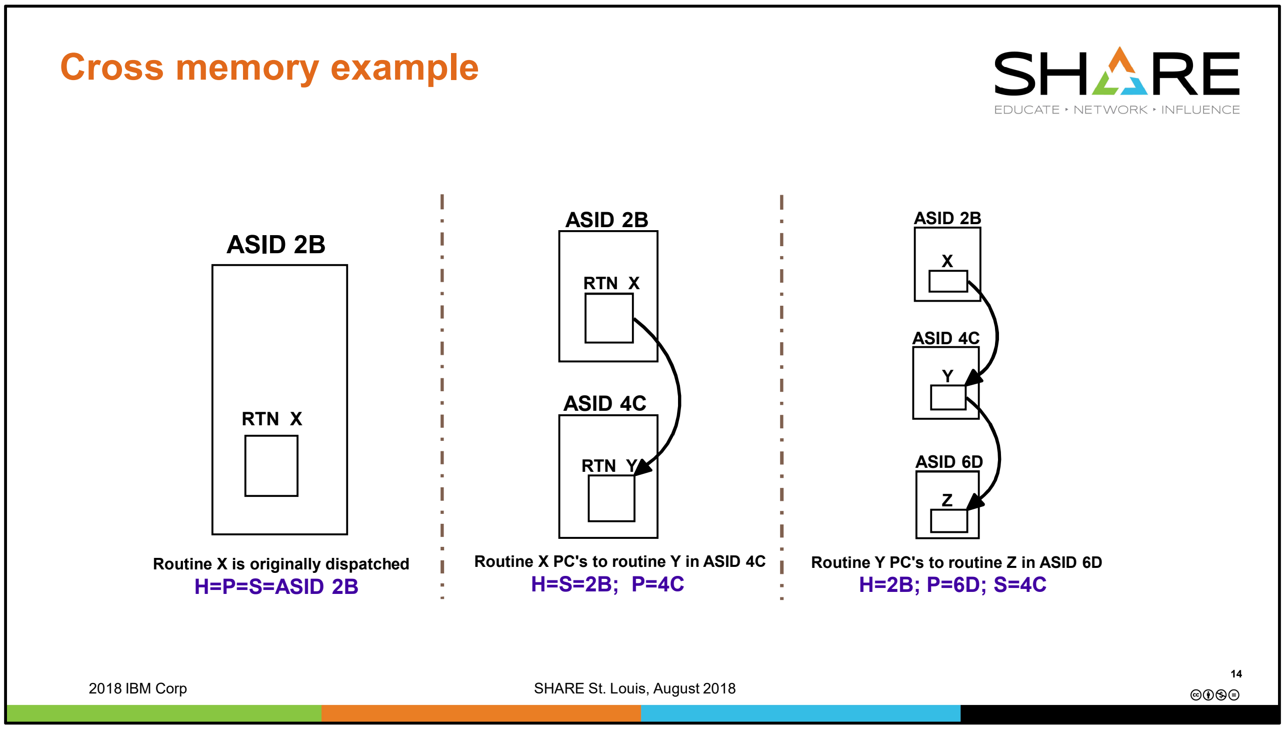
A simple example

- Pgm A begins life with H=P=S=ASID X
- Pgm A issues a PC

- The PC causes a synchronous “jump” to Pgm B in ASID Y
  - Execution ASID becomes new PRIMARY
    - PRIMARY is now ASID Y
  - Previous PRIMARY becomes new SECONDARY
    - SECONDARY is now ASID X

- Pgm B completes and issues a PR
  - PRIMARY restored to time of PC
  - SECONDARY restored to time of PC
  - Processing continues at instruction after PC





Cross memory example

Routine X is originally dispatched
H=P=S=ASID 2B

Routine X PC's to routine Y in ASID 4C
H=S=2B; P=4C

Routine Y PC's to routine Z in ASID 6D
H=2B; P=6D; S=4C

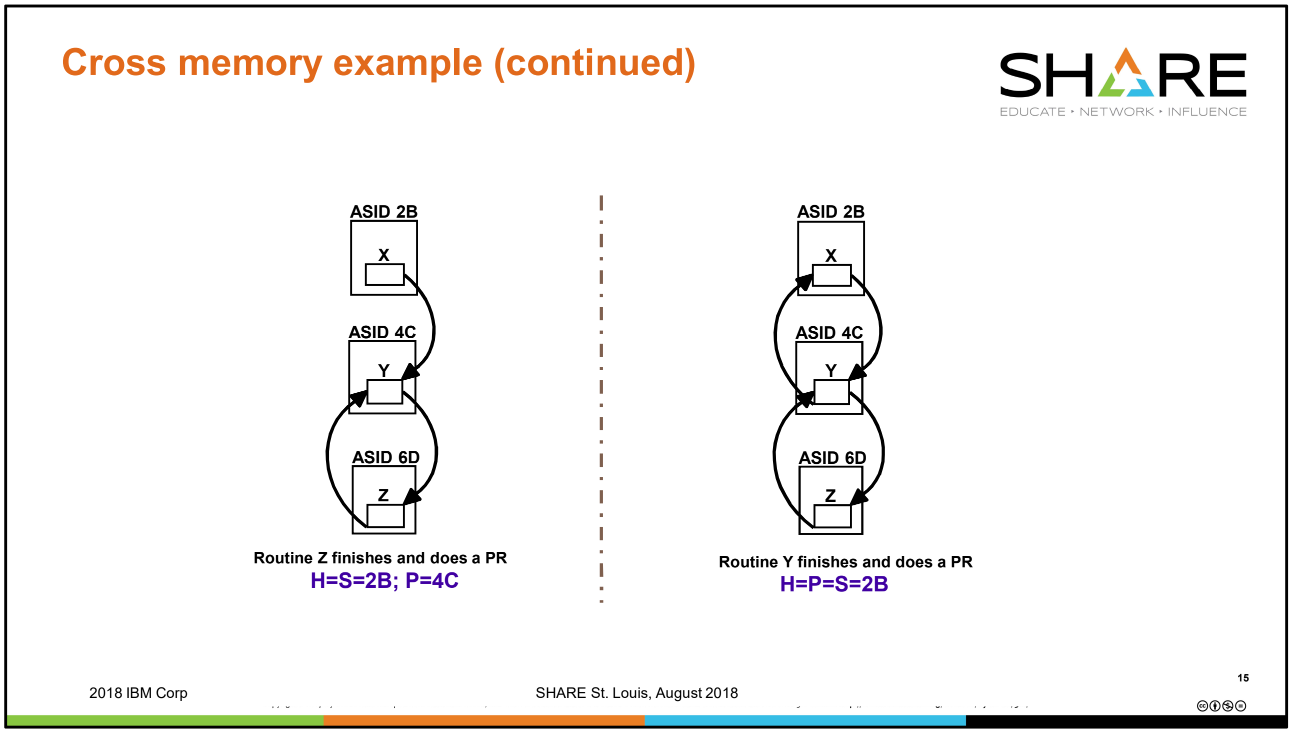
2018 IBM Corp SHARE St. Louis, August 2018 14

A job X begins in ASID 2B. Home, primary, and second address spaces are all the same at this point.

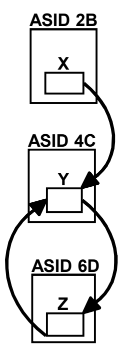
Routine X PCs to Routine Y in ASID 4C. ASID 4C is now the new PRIMARY. ASID 2B, the previous PRIMARY, is now the new SECONDARY. HOME is ASID 2B, the dispatched address space, and will not change.

Routine Y PCs to ROUTINE Z in ASID 6D. ASID 6D becomes the new PRIMARY, ASID 4C becomes the new SECONDARY, and ASID 2B remains HOME.

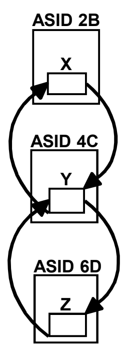
At this point, we have ready addressability to 3 different address spaces.



Cross memory example (continued)



Routine Z finishes and does a PR
H=S=2B; P=4C

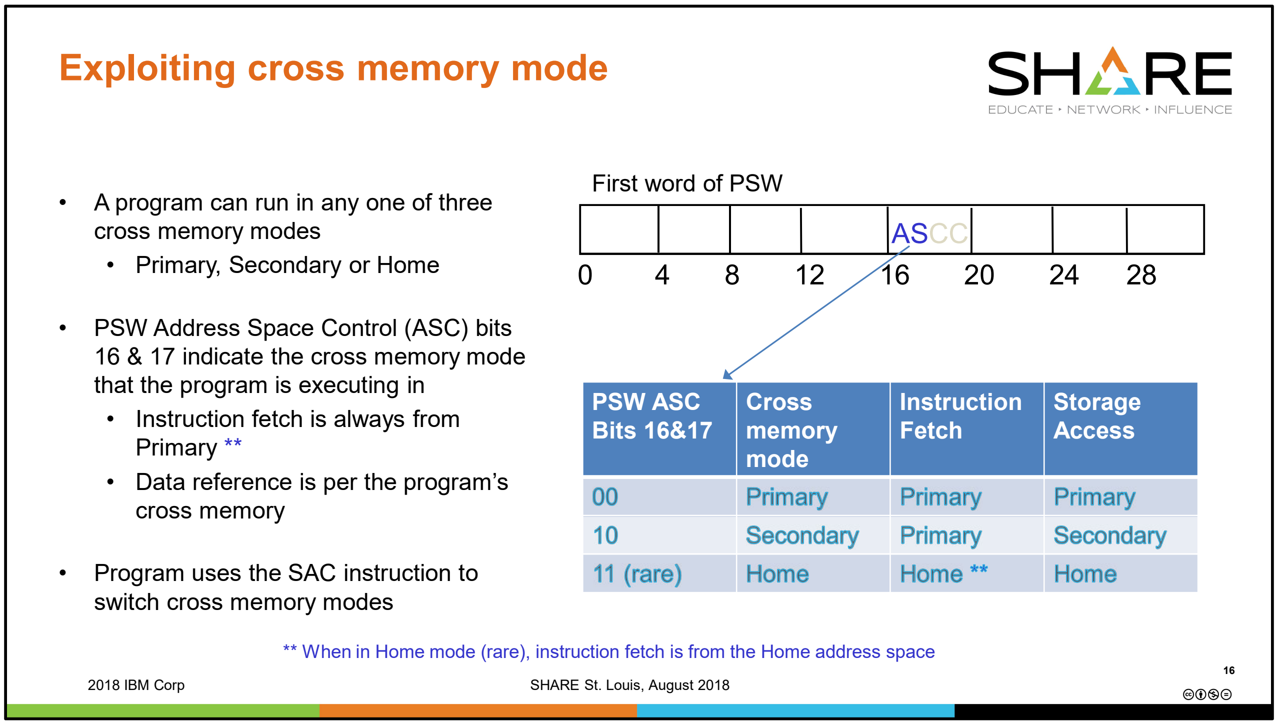


Routine Y finishes and does a PR
H=P=S=2B

2018 IBM Corp SHARE St. Louis, August 2018 15

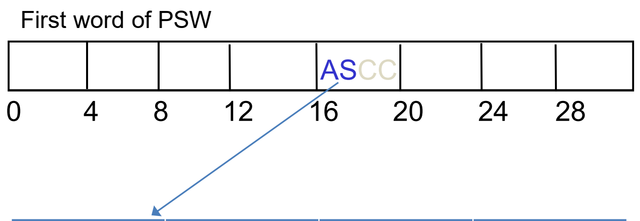
When Routine Z finishes, it returns to Y in ASID 4C via PR. ASID 4C becomes the new PRIMARY, ASID 2B becomes the new SECONDARY (as it was before Y PC'd to Z), and ASID 2B is still HOME.

Routine Y finishes, and returns to X via PR. PRIMARY is again ASID 2B, SECONDARY is again ASID 2B. HOME is, and always has been, ASID 2B. At this point, we are back to H=P=S=2B.



Exploiting cross memory mode

- A program can run in any one of three cross memory modes
  - Primary, Secondary or Home
- PSW Address Space Control (ASC) bits 16 & 17 indicate the cross memory mode that the program is executing in
  - Instruction fetch is always from Primary **
  - Data reference is per the program's cross memory
- Program uses the SAC instruction to switch cross memory modes



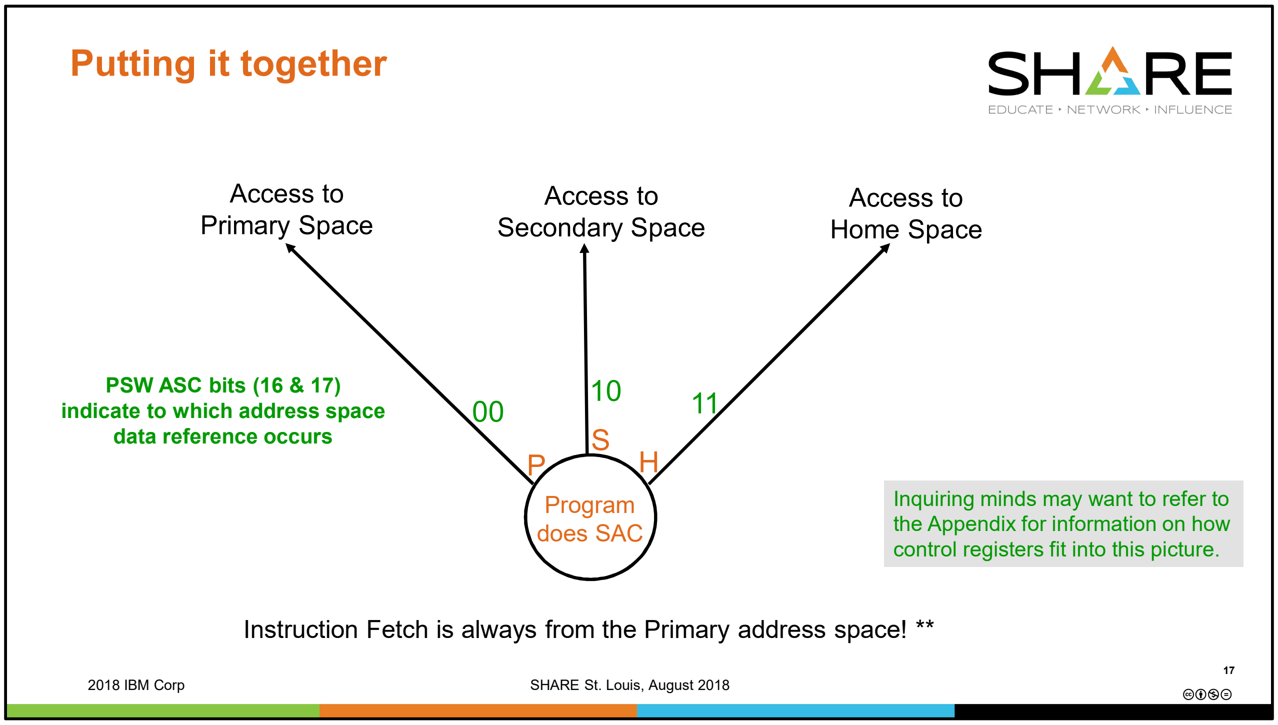
PSW ASC Bits 16&17	Cross memory mode	Instruction Fetch	Storage Access
00	Primary	Primary	Primary
10	Secondary	Primary	Secondary
11 (rare)	Home	Home **	Home

** When in Home mode (rare), instruction fetch is from the Home address space

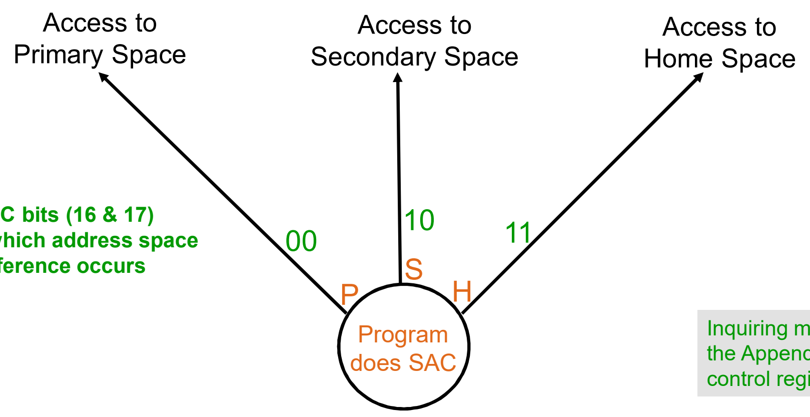
2018 IBM Corp SHARE St. Louis, August 2018 16

An executing program controls the cross memory mode that it wants to execute in. Typically it will run in primary mode where it has ready access to the primary address space's data. However it can easily SAC to secondary to gain access to the data in the secondary address space instead, or it can SAC to home to gain access to the data in the home address space. When a SAC instruction is executed, it updates the PSW ASC bits (16 and 17) in the PSW. This is what tells the hardware whether to access storage in the primary, secondary or home address space when machine instructions are executed.

The PSW ASC mode does not influence instruction fetch (except in the unusual case of running in home mode). Except in the case of home mode, instruction fetch always is from the primary address space.



Putting it together

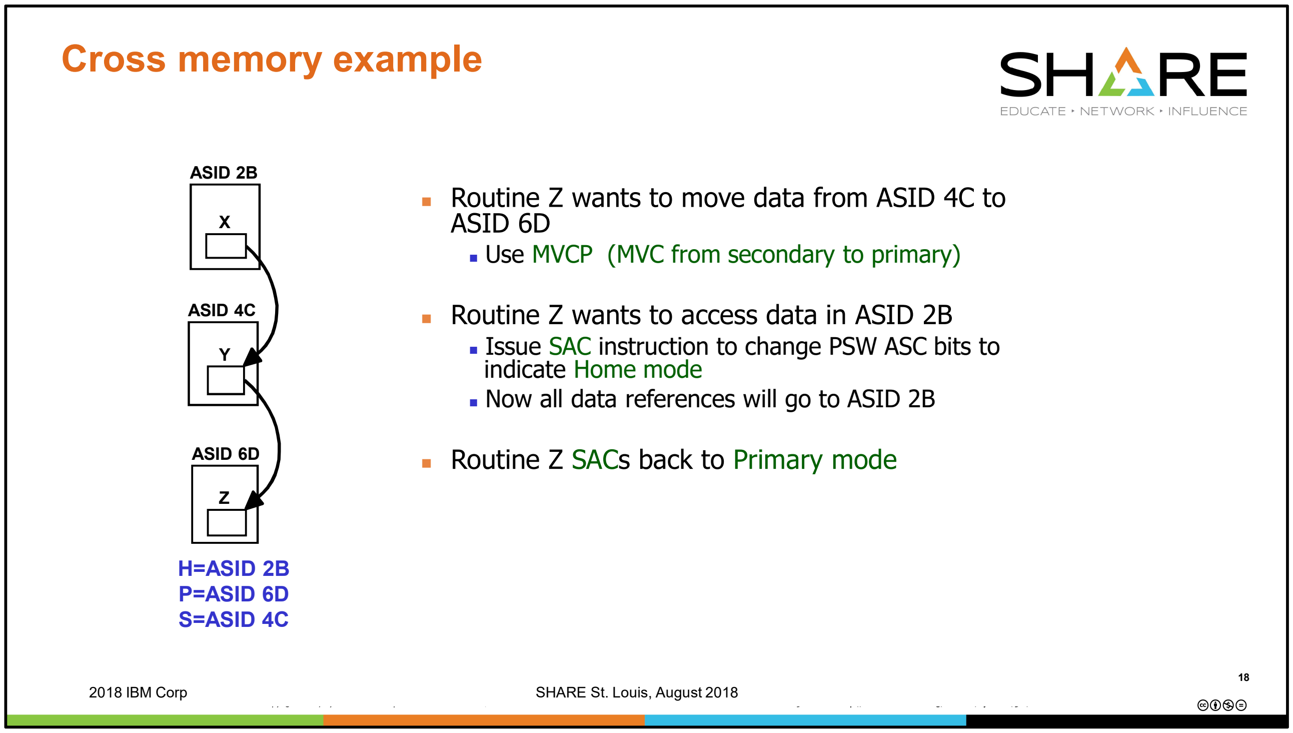


PSW ASC bits (16 & 17) indicate to which address space data reference occurs

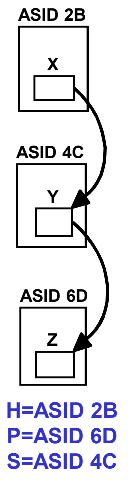
Inquiring minds may want to refer to the Appendix for information on how control registers fit into this picture.

Instruction Fetch is always from the Primary address space! **

The SAC instruction acts like a dial that a program can turn back and forth to control which address space's storage (primary, secondary, or home) it wants to access.



Cross memory example

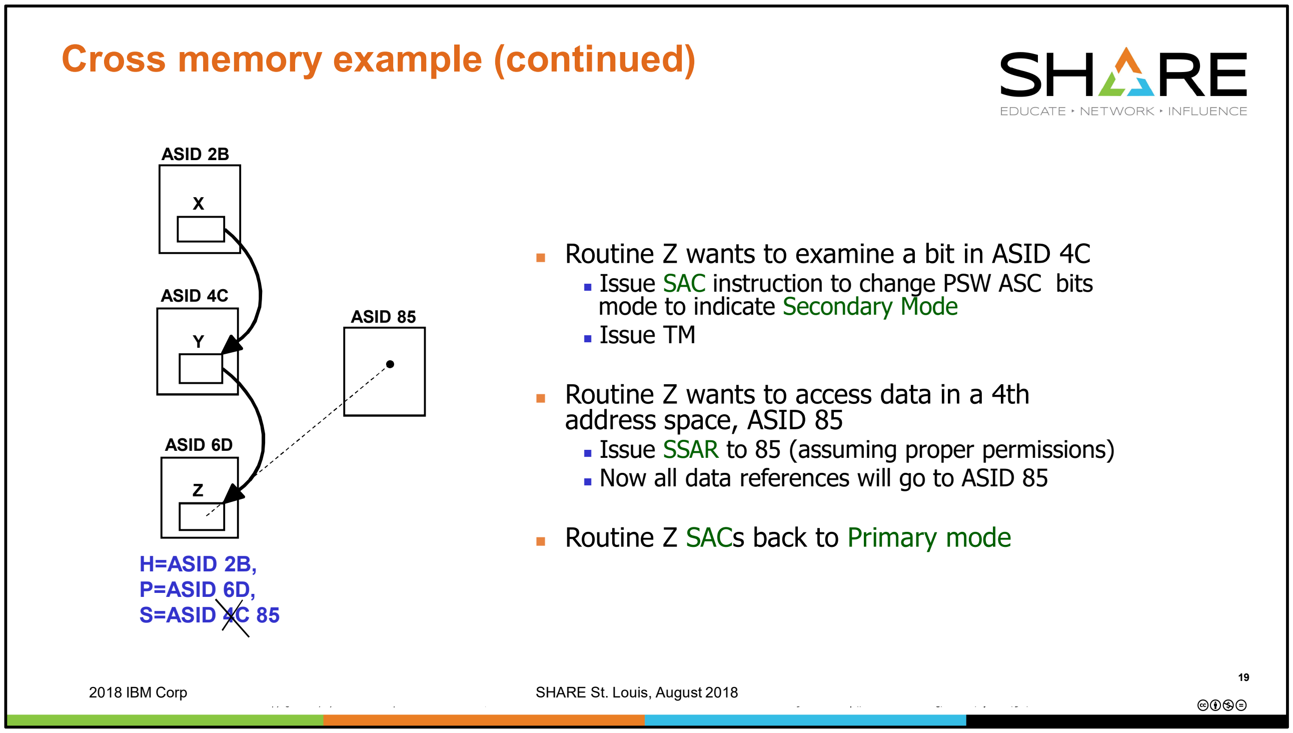


- Routine Z wants to move data from ASID 4C to ASID 6D
  - Use MVCP (MVC from secondary to primary)
- Routine Z wants to access data in ASID 2B
  - Issue SAC instruction to change PSW ASC bits to indicate Home mode
  - Now all data references will go to ASID 2B
- Routine Z SACs back to Primary mode

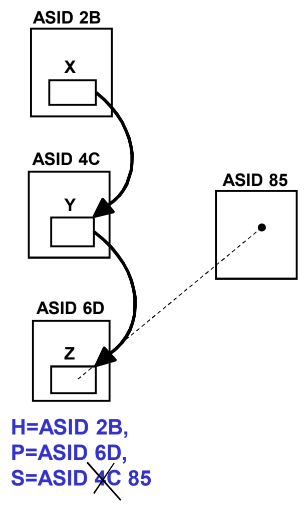
2018 IBM Corp SHARE St. Louis, August 2018 18

When Routine Z is PC'd to, we assume Routine Z has not yet issued SAC to change PSW ASC mode, so assume PSW ASC mode is PRIMARY, which means that all data references will be made to current PRIMARY, 6D. If Z wants to move data from ASID 4C (SECONDARY) to ASID 6D (PRIMARY), it can use MVCP.

To access data in ASID 2B, Z can issue SAC to HOME. Now all data references (LOAD, STORE, MVC, etc.) will occur in ASID 2B. Remember that when in HOME mode, instructions are also fetched from the HOME address space. This means that Routine Z had better live in common (global) storage if its instructions are to be successfully accessed both from the HOME space and from the PRIMARY space! In our diagram, a PC to routine Z results in a space switch to PRIMARY=ASID 6D. Note that it is perfectly okay for a space-switching PC routine to live in global storage.



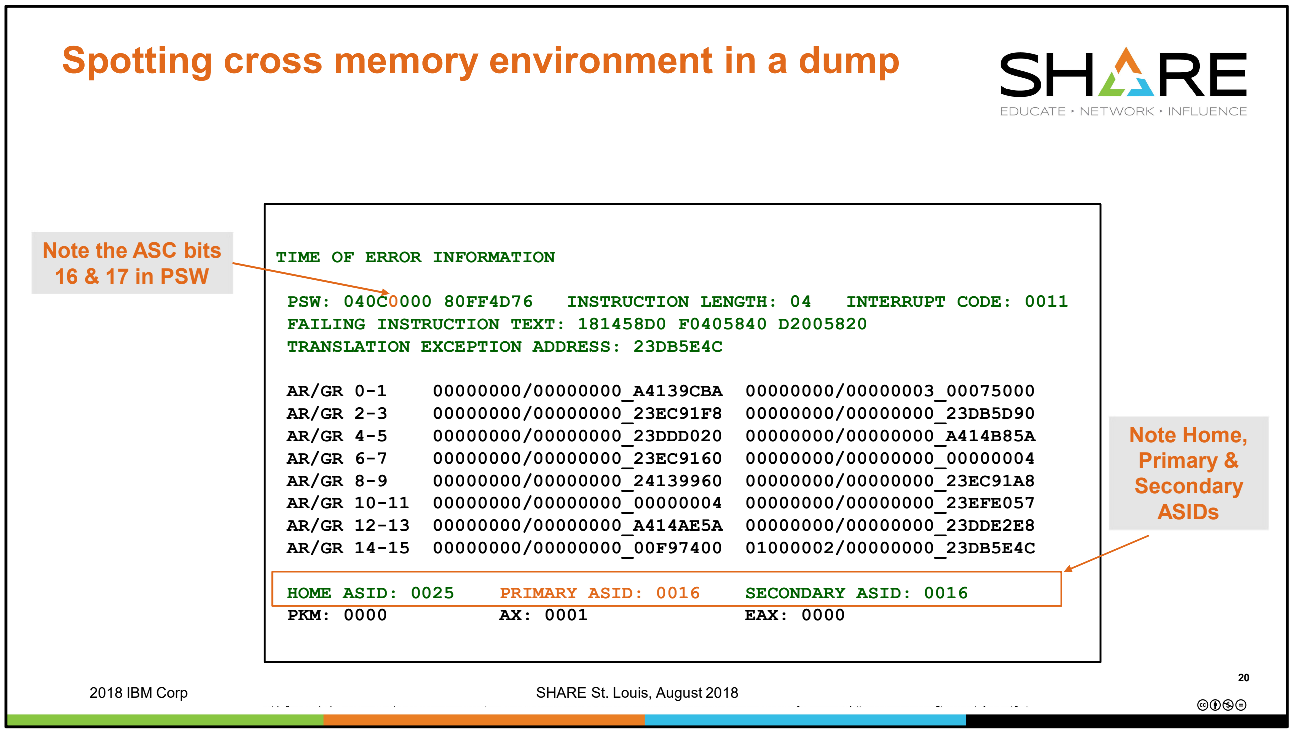
Cross memory example (continued)



- Routine Z wants to examine a bit in ASID 4C
  - Issue SAC instruction to change PSW ASC bits mode to indicate Secondary Mode
  - Issue TM
- Routine Z wants to access data in a 4th address space, ASID 85
  - Issue SSAR to 85 (assuming proper permissions)
  - Now all data references will go to ASID 85
- Routine Z SACs back to Primary mode

We are using the same example as before, except now Routine Z wants to access data in a 4th address space, not one that it PC'd through.

Assuming the authority has been granted, Z can SSAR to ASID 85 to designate ASID 85 as the new SECONDARY. Then a SAC to SECONDARY allows all data references to occur in ASID 85 until Z SACs back to PRIMARY.



Spotting cross memory environment in a dump

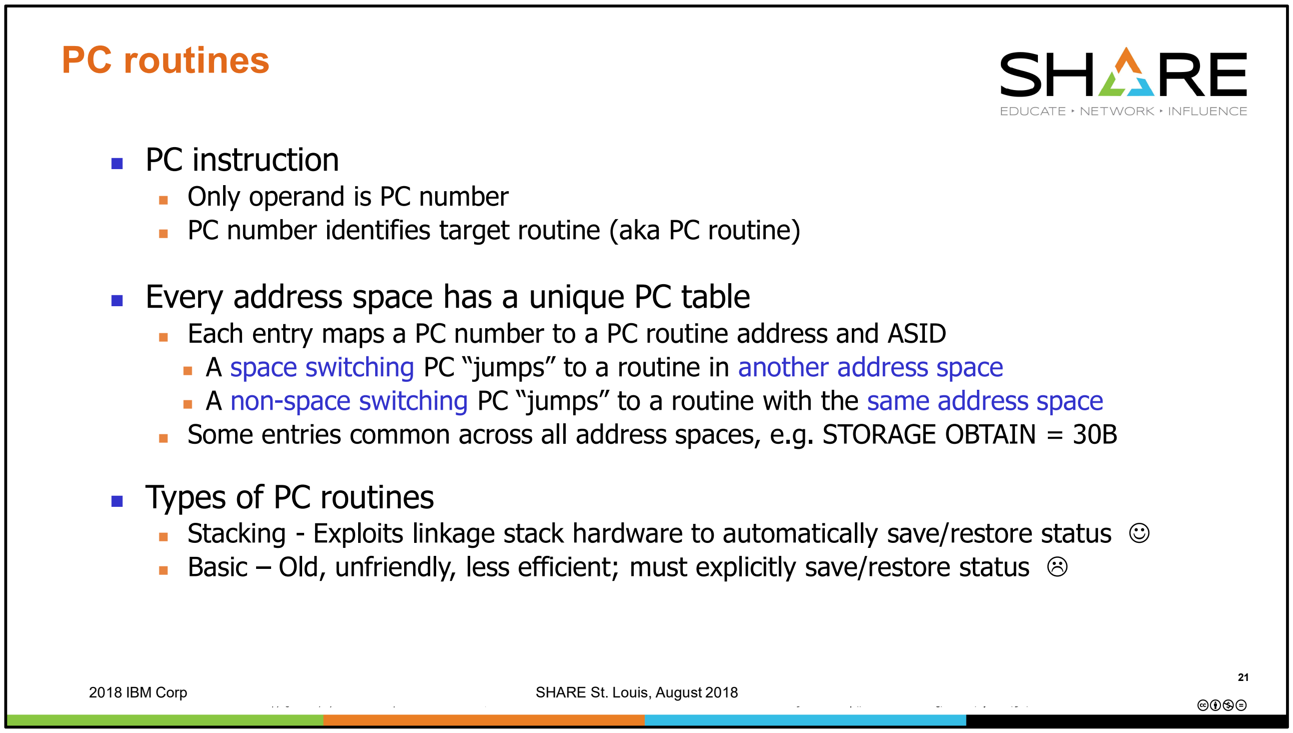
Note the ASC bits 16 & 17 in PSW

```
TIME OF ERROR INFORMATION

 PSW: 040C0000 80FF4D76   INSTRUCTION LENGTH: 04   INTERRUPT CODE: 0011
 FAILING INSTRUCTION TEXT: 181458D0 F0405840 D2005820
 TRANSLATION EXCEPTION ADDRESS: 23DB5E4C

 AR/GR 0-1     00000000/00000000_A4139CBA  00000000/00000003_00075000
 AR/GR 2-3     00000000/00000000_23EC91F8  00000000/00000000_23DB5D90
 AR/GR 4-5     00000000/00000000_23DDD020  00000000/00000000_A414B85A
 AR/GR 6-7     00000000/00000000_23EC9160  00000000/00000000_00000004
 AR/GR 8-9     00000000/00000000_24139960  00000000/00000000_23EC91A8
 AR/GR 10-11   00000000/00000000_00000004  00000000/00000000_23EFE057
 AR/GR 12-13   00000000/00000000_A414AE5A  00000000/00000000_23DDE2E8
 AR/GR 14-15   00000000/00000000_00F97400  01000002/00000000_23DB5E4C

 HOME ASID: 0025     PRIMARY ASID: 0016     SECONDARY ASID: 0016
 PKM: 0000           AX: 0001               EAX: 0000
```

Note Home, Primary & Secondary ASIDs

2018 IBM Corp   SHARE St. Louis, August 2018   20

The highlighted zero in the PSW shows ASC mode bits 16-17 are both zero – Primary mode. The dispatched address space is ASID x'25', but execution is in PRIMARY ASID x'16'. Data reference is also in ASID x'16'.



PC routines

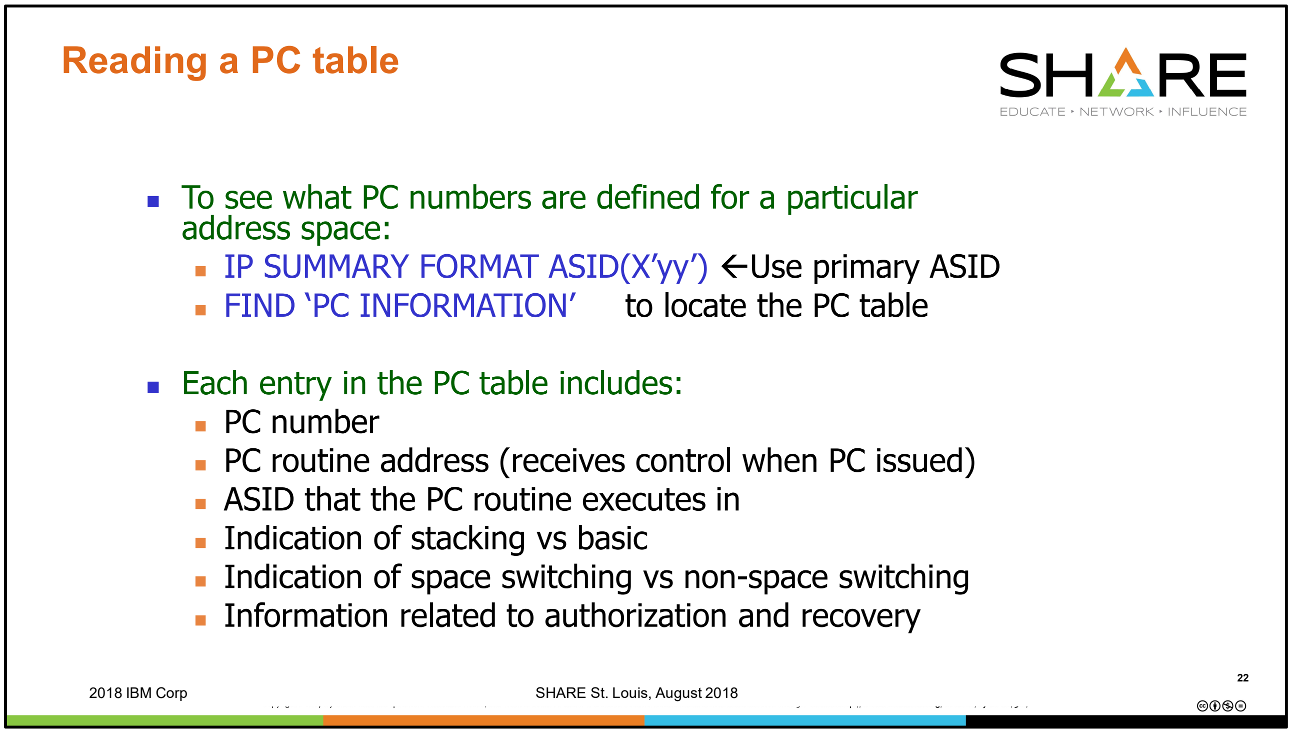
- PC instruction
  - Only operand is PC number
  - PC number identifies target routine (aka PC routine)
- Every address space has a unique PC table
  - Each entry maps a PC number to a PC routine address and ASID
    - A space switching PC "jumps" to a routine in another address space
    - A non-space switching PC "jumps" to a routine with the same address space
  - Some entries common across all address spaces, e.g. STORAGE OBTAIN = 30B
- Types of PC routines
  - Stacking - Exploits linkage stack hardware to automatically save/restore status ☺
  - Basic – Old, unfriendly, less efficient; must explicitly save/restore status ☹

2018 IBM Corp SHARE St. Louis, August 2018 21

The PC number uniquely identifies a target routine which may or may not reside in a different address space. The PC number is actually comprised of 2 concatenated indices that hardware uses to index into tables, ultimately mapping a PC number to a target routine.

Some PC numbers are system-defined. These belong to system services invoked via the PC instruction.

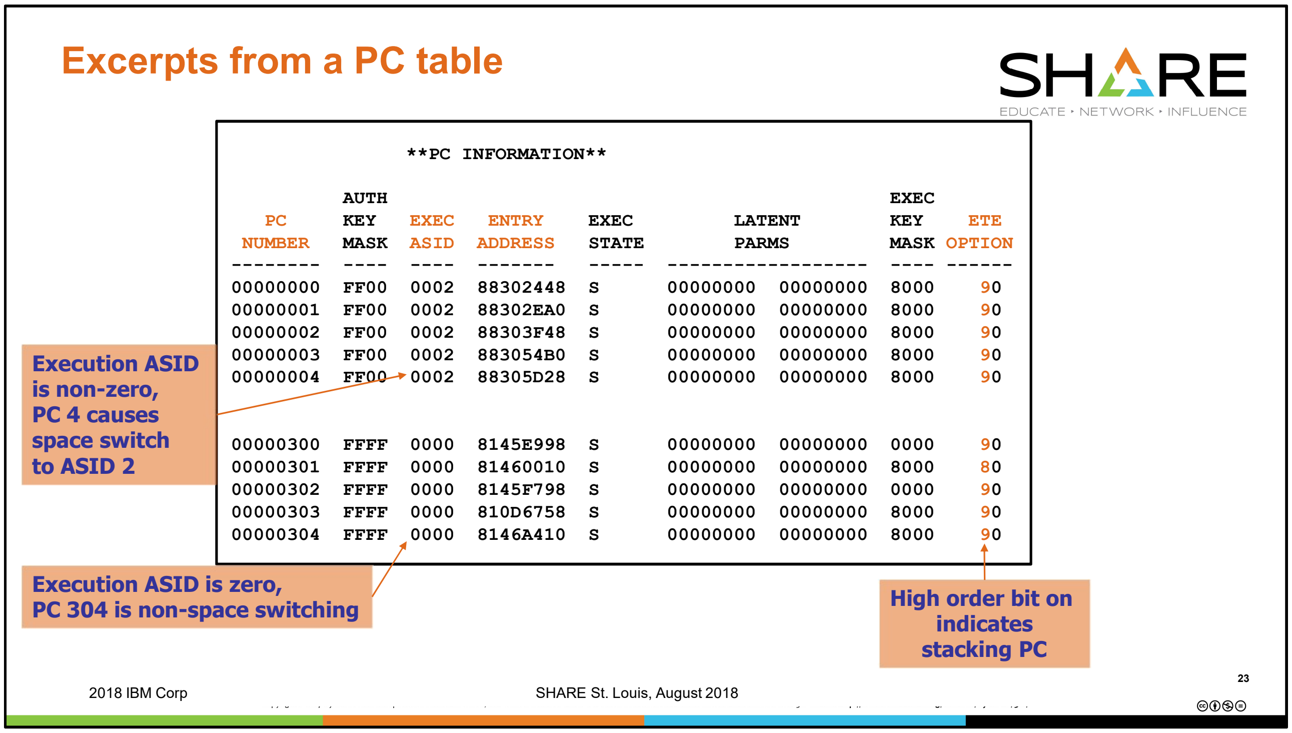
For more information on system-defined PCs, see MVS Diagnosis: Reference, Chapter 5: Program Call (PC) Services in System Function Table.



Reading a PC table

- To see what PC numbers are defined for a particular address space:
  - IP SUMMARY FORMAT ASID(X'yy') ←Use primary ASID
  - FIND 'PC INFORMATION' to locate the PC table
- Each entry in the PC table includes:
  - PC number
  - PC routine address (receives control when PC issued)
  - ASID that the PC routine executes in
  - Indication of stacking vs basic
  - Indication of space switching vs non-space switching
  - Information related to authorization and recovery

The PC Information table is located near the top of IP SUMMARY FORMAT, just under the address space-related control blocks and before the TCBs. Note that applications can create their own unique, address-space-specific PC's, so it is very important to review IP SUMMARY FORMAT for the PRIMARY address space at the time the PC was issued to obtain accurate information about the target PC routine.



Excerpts from a PC table

```
                **PC INFORMATION**

           AUTH                                                EXEC
   PC      KEY   EXEC   ENTRY     EXEC          LATENT         KEY    ETE
 NUMBER    MASK  ASID  ADDRESS    STATE         PARMS          MASK  OPTION
--------   ----  ----  -------    -----   -----------------    ----  ------
00000000   FF00  0002  88302448   S       00000000  00000000   8000     90
00000001   FF00  0002  88302EA0   S       00000000  00000000   8000     90
00000002   FF00  0002  88303F48   S       00000000  00000000   8000     90
00000003   FF00  0002  883054B0   S       00000000  00000000   8000     90
00000004   FF00  0002  88305D28   S       00000000  00000000   8000     90


00000300   FFFF  0000  8145E998   S       00000000  00000000   0000     90
00000301   FFFF  0000  81460010   S       00000000  00000000   8000     80
00000302   FFFF  0000  8145F798   S       00000000  00000000   0000     90
00000303   FFFF  0000  810D6758   S       00000000  00000000   8000     90
00000304   FFFF  0000  8146A410   S       00000000  00000000   8000     90
```

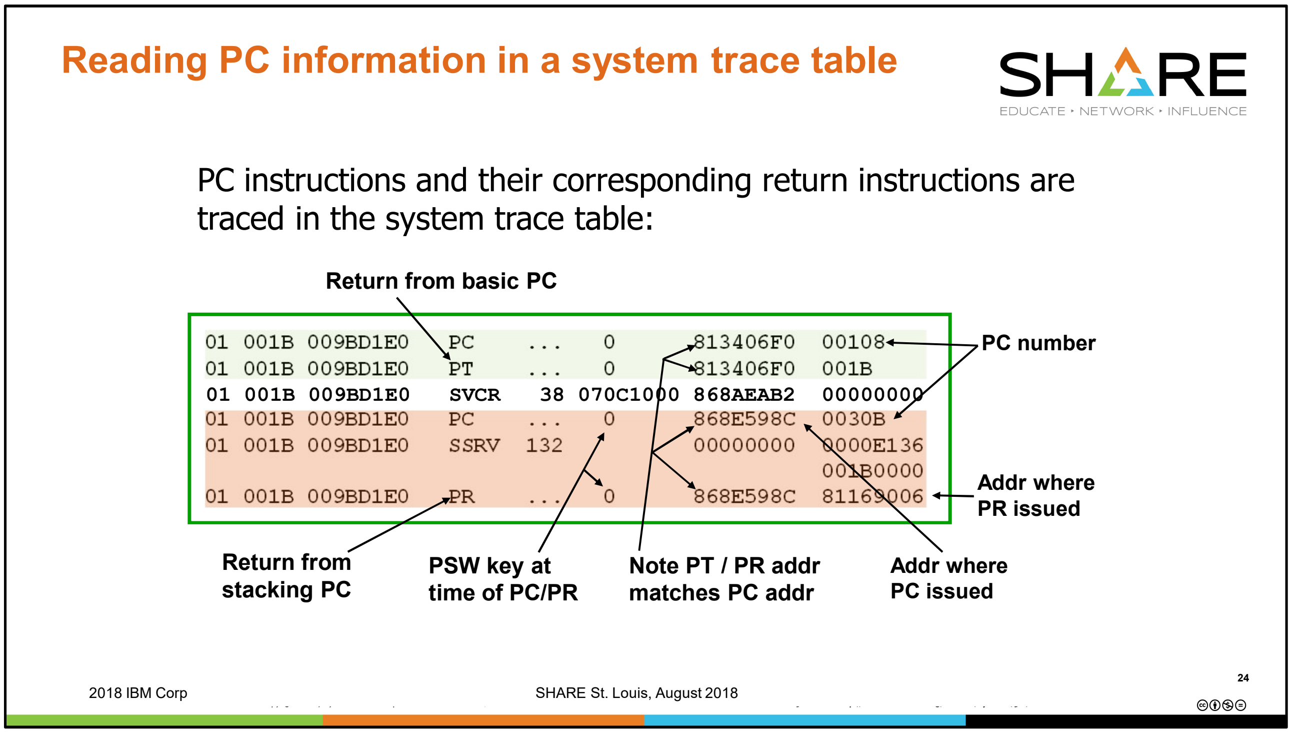
Execution ASID is non-zero, PC 4 causes space switch to ASID 2

Execution ASID is zero, PC 304 is non-space switching

High order bit on indicates stacking PC

2018 IBM Corp SHARE St. Louis, August 2018 23

Use IPCS WHERE on the Entry Address qualified with the EXEC ASID (example: IP W 08302448 ASID(2)), or browse storage in the EXEC ASID to identify the routine that will receive control for a given PC.



Reading PC information in a system trace table

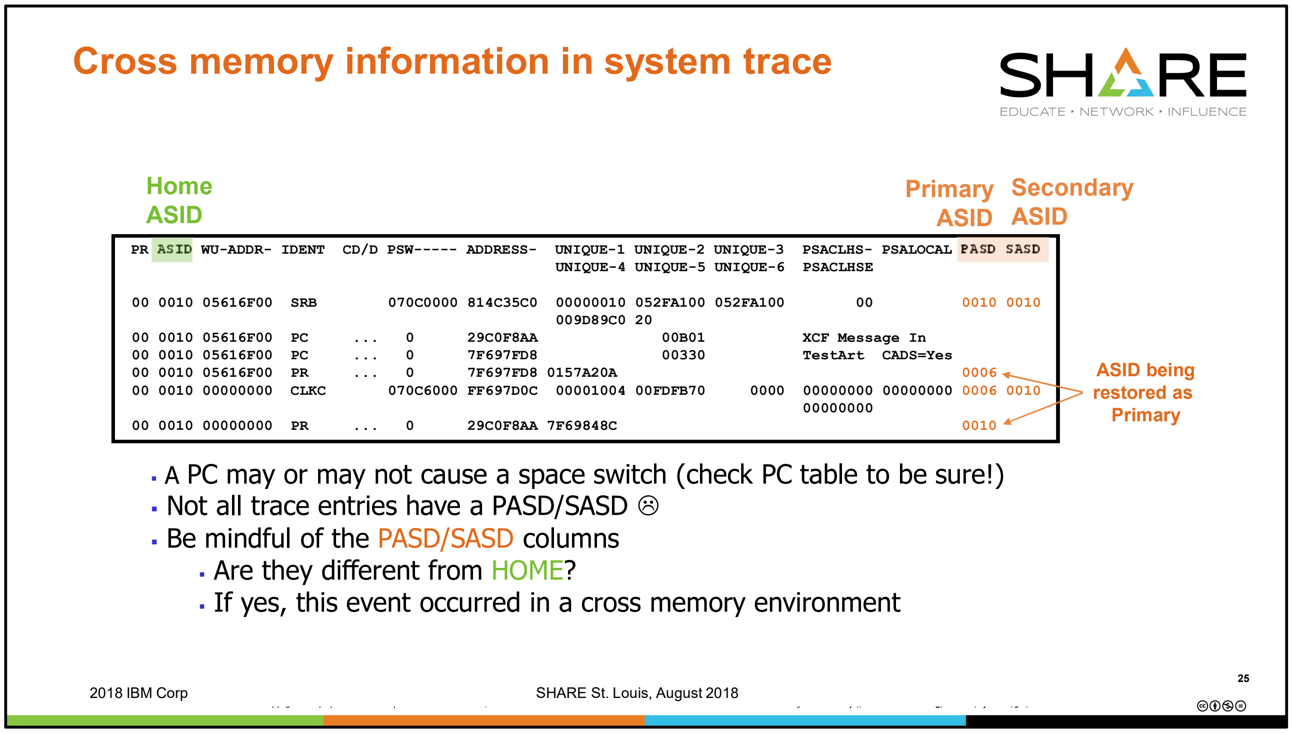
PC instructions and their corresponding return instructions are traced in the system trace table:

```
01  001B  009BD1E0   PC     ...   0      813406F0   00108
01  001B  009BD1E0   PT     ...   0      813406F0   001B
01  001B  009BD1E0   SVCR    38  070C1000 868AEAB2  00000000
01  001B  009BD1E0   PC     ...   0      868E598C   0030B
01  001B  009BD1E0   SSRV   132          00000000   0000E136
                                                    001B0000
01  001B  009BD1E0   PR     ...   0      868E598C   81169006
```

A PC entry will have the same PSW address as its corresponding PT or PR instruction.

PC / PR or PC / PT entries can be nested. When looking at these entries in a system trace table, match them up using the PSW address, similar to matching up SVC / SVCR entries in a system trace table or DO / END statements in a computer program.

You cannot tell if a PC instruction is basic (and so will be matched with a PT) or stacking (and so will be matched with a PR) directly from the system trace table. You could take the PC number and look it up in the SUMM FORMAT PC Information Table, but its generally easier to just match up PSW addresses in order to locate the PT / PR.



Cross memory information in system trace

Home ASID

Primary ASID

Secondary ASID

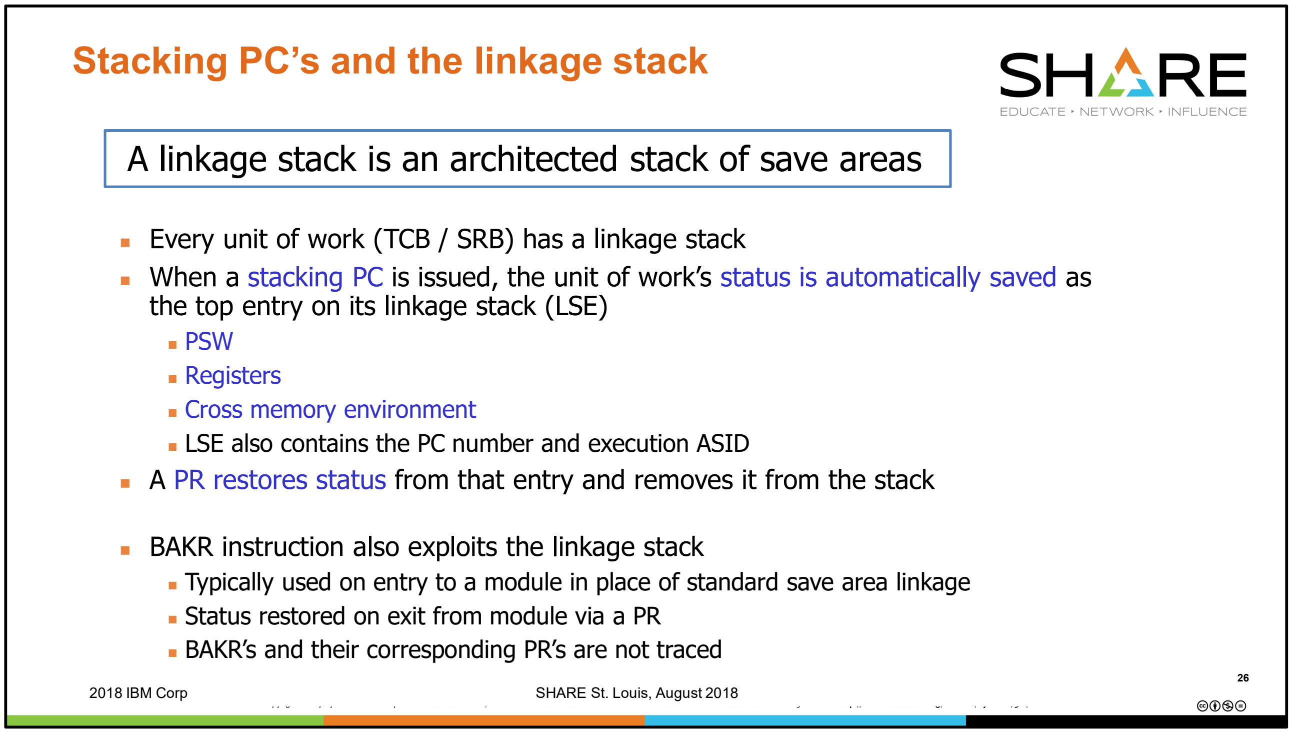
```
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD
                                                UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE

00 0010 05616F00  SRB       070C0000 814C35C0  00000010 052FA100 052FA100        00          0010 0010
                                                009D89C0 20
00 0010 05616F00  PC     ...   0      29C0F8AA           00B01              XCF Message In
00 0010 05616F00  PC     ...   0      7F697FD8           00330              TestArt  CADS=Yes
00 0010 05616F00  PR     ...   0      7F697FD8 0157A20A                                      0006
00 0010 00000000  CLKC      070C6000 FF697D0C  00001004 00FDFB70      0000  00000000 00000000 0006 0010
                                                                            00000000
00 0010 00000000  PR     ...   0      29C0F8AA 7F69848C                                      0010
```

ASID being restored as Primary

- A PC may or may not cause a space switch (check PC table to be sure!)
- Not all trace entries have a PASD/SASD ☹
- Be mindful of the PASD/SASD columns
  - Are they different from HOME?
  - If yes, this event occurred in a cross memory environment

2018 IBM Corp SHARE St. Louis, August 2018 25

When analyzing the System Trace Table, be alert to possibility of a cross memory environment. On the left is the dispatched address space (HOME), but on the right (you may need to adjust your screen or shift right) you will see 2 columns, PASD and SASD (PRIMARY ASID and SECONDARY ASID). If these are different than HOME, that trace entry represents an event that occurred in a cross memory environment. It is good practice to pay close attention to these two columns.



Stacking PC's and the linkage stack

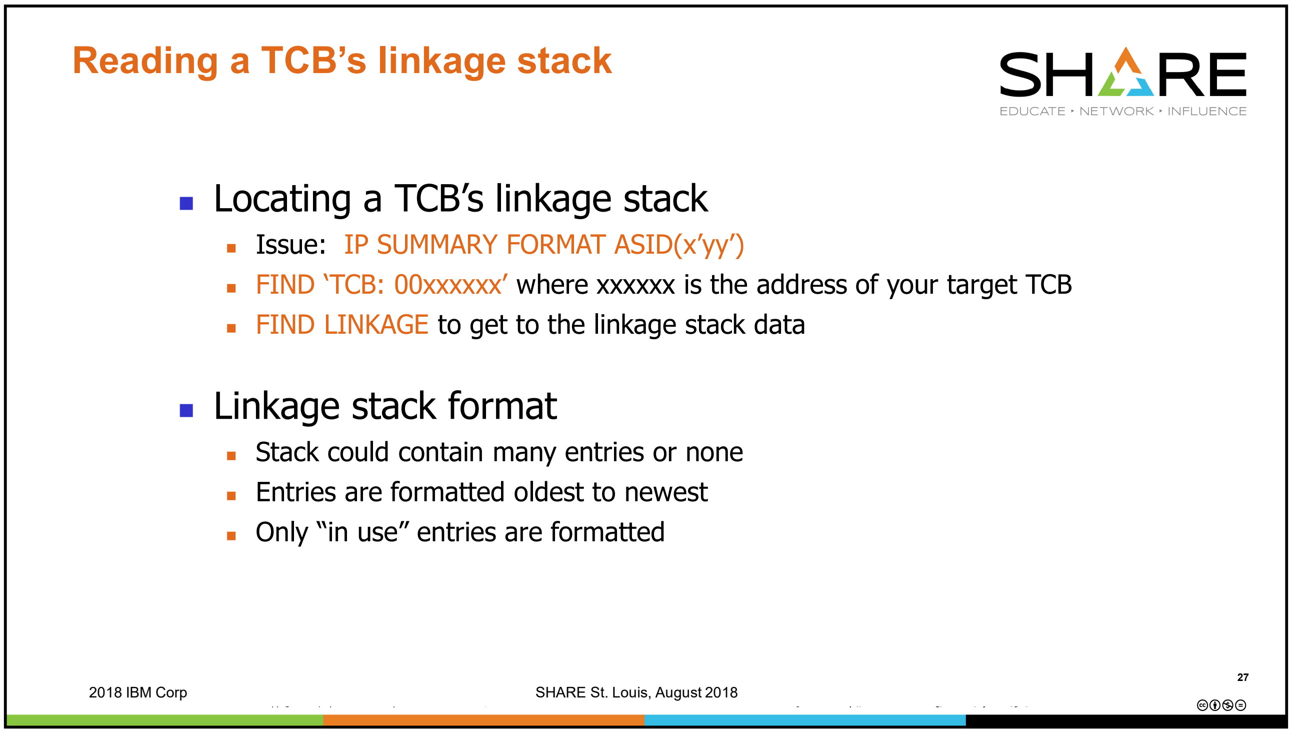
A linkage stack is an architected stack of save areas

- Every unit of work (TCB / SRB) has a linkage stack
- When a stacking PC is issued, the unit of work's status is automatically saved as the top entry on its linkage stack (LSE)
  - PSW
  - Registers
  - Cross memory environment
  - LSE also contains the PC number and execution ASID
- A PR restores status from that entry and removes it from the stack

- BAKR instruction also exploits the linkage stack
  - Typically used on entry to a module in place of standard save area linkage
  - Status restored on exit from module via a PR
  - BAKR's and their corresponding PR's are not traced

2018 IBM Corp SHARE St. Louis, August 2018 26

The exploitation of linkage stacks by stacking PCs provides an easy and efficient way to save program status as program A PC's to program B. For this reason a BAKR instruction was also added with the linkage stack architecture so that code that is not a PC routine can exploit the convenience of the linkage stack. A BAKR (branch and stack register) instruction with R0 specified as the target of the branch (meaning don't branch) can be used on program entry to save the caller's status on the linkage stack. This technique is much simpler than using standard save area logic. Similar to a PC instruction, a PR "undoes" a BAKR instruction by restoring status from the linkage stack.



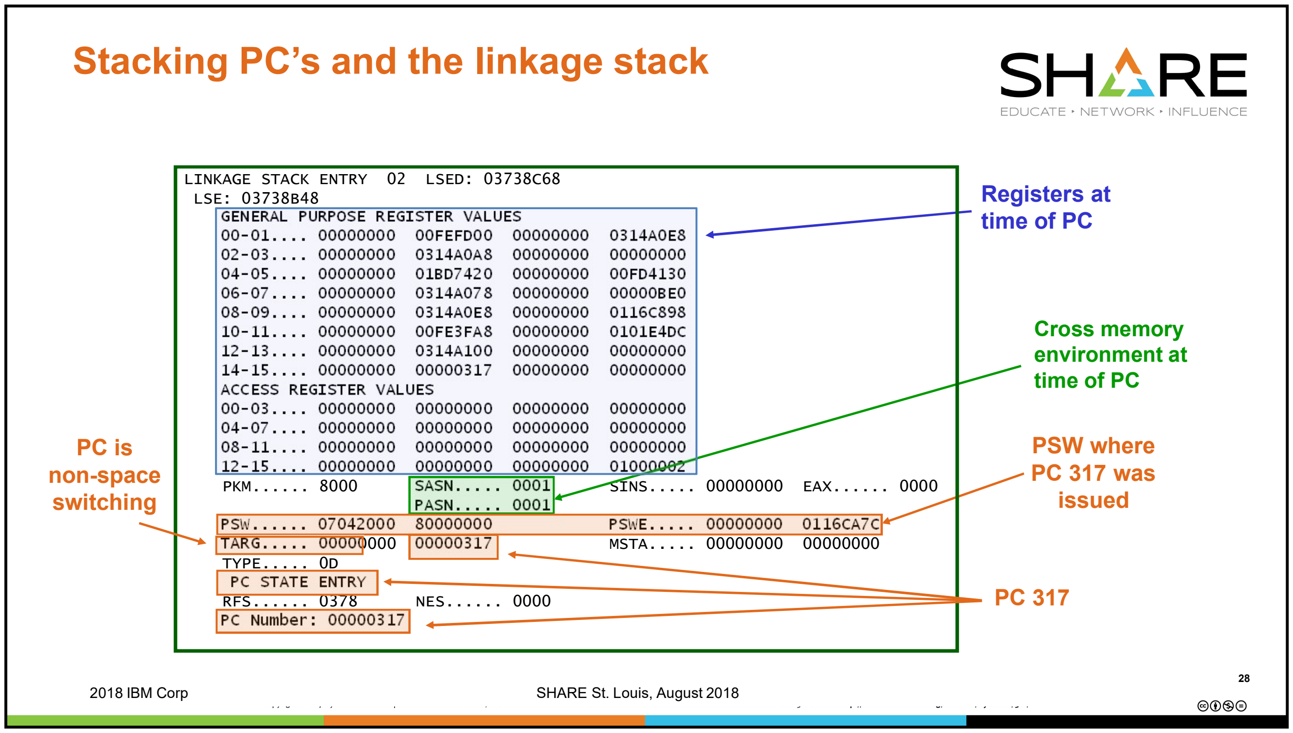
Reading a TCB's linkage stack

- Locating a TCB's linkage stack
  - Issue: IP SUMMARY FORMAT ASID(x'yy')
  - FIND 'TCB: 00xxxxxx' where xxxxxx is the address of your target TCB
  - FIND LINKAGE to get to the linkage stack data
- Linkage stack format
  - Stack could contain many entries or none
  - Entries are formatted oldest to newest
  - Only "in use" entries are formatted

2018 IBM Corp SHARE St. Louis, August 2018 27

There is nothing of interest in the linkage stack header and trailer entries. They are only mentioned here because they may be formatted out in SUMM FORMAT, albeit inconspicuously. The interesting linkage stack entries are the "state" entries which are where the save area information is stored.

Only "in use" linkage stack entries are formatted. Once an entry is popped off the linkage stack (i.e. once you see a PR for it, for example, in the system trace table), then you will no longer see that entry formatted in SUMM FORMAT. The data remains residually in the linkage stack entry until it is reinitialized as a result of a new stacking PC or BAKR.



Stacking PC's and the linkage stack

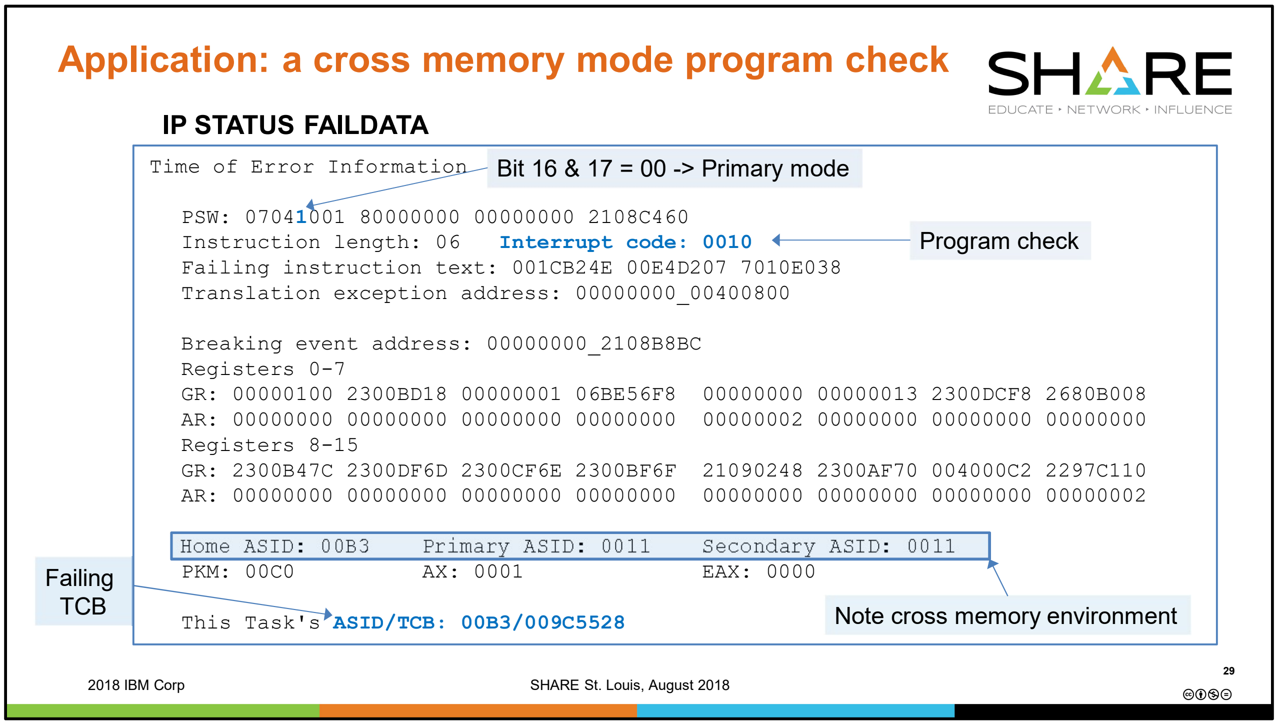
```
LINKAGE STACK ENTRY  02  LSED: 03738C68
 LSE: 03738B48
    GENERAL PURPOSE REGISTER VALUES
    00-01.... 00000000  00FEFD00  00000000  0314A0E8
    02-03.... 00000000  0314A0A8  00000000  00000000
    04-05.... 00000000  01BD7420  00000000  00FD4130
    06-07.... 00000000  0314A078  00000000  00000BE0
    08-09.... 00000000  0314A0E8  00000000  0116C898
    10-11.... 00000000  00FE3FA8  00000000  0101E4DC
    12-13.... 00000000  0314A100  00000000  00000000
    14-15.... 00000000  00000317  00000000  00000000
    ACCESS REGISTER VALUES
    00-03.... 00000000  00000000  00000000  00000000
    04-07.... 00000000  00000000  00000000  00000000
    08-11.... 00000000  00000000  00000000  00000000
    12-15.... 00000000  00000000  00000000  01000002
    PKM...... 8000      SASN..... 0001      SINS..... 00000000  EAX...... 0000
                        PASN..... 0001
    PSW...... 07042000  80000000            PSWE..... 00000000  0116CA7C
    TARG..... 00000000  00000317            MSTA..... 00000000  00000000
    TYPE..... 0D
     PC STATE ENTRY
    RFS...... 0378      NES...... 0000
    PC Number: 00000317
```

Registers at time of PC
Cross memory environment at time of PC
PSW where PC 317 was issued
PC is non-space switching
PC 317

This is a linkage stack entry created as a result of code in ASID X'1' issuing a PC 317.

Note that the Primary and Secondary Address Space Numbers (PASN and SASN) are from the time the PC was issued, not after it was executed. The first 2 bytes of the TARG indicate whether the PC is space-switching. If 0 (as in this example), it is non-space switching. If non-zero, then the PC routine will get control in the designated address space.

Note: the PASN field actually appears to the right of the EAX field in SUMMARY FORMAT. In the interest of readability, a field called PINS which was to the right of PASN has been omitted from this example.



Application: a cross memory mode program check

IP STATUS FAILDATA

```
Time of Error Information

  PSW: 07041001 80000000 00000000 2108C460
  Instruction length: 06   Interrupt code: 0010
  Failing instruction text: 001CB24E 00E4D207 7010E038
  Translation exception address: 00000000_00400800

  Breaking event address: 00000000_2108B8BC
  Registers 0-7
  GR: 00000100 2300BD18 00000001 06BE56F8  00000000 00000013 2300DCF8 2680B008
  AR: 00000000 00000000 00000000 00000000  00000002 00000000 00000000 00000000
  Registers 8-15
  GR: 2300B47C 2300DF6D 2300CF6E 2300BF6F  21090248 2300AF70 004000C2 2297C110
  AR: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000002

  Home ASID: 00B3     Primary ASID: 0011     Secondary ASID: 0011
  PKM: 00C0           AX: 0001               EAX: 0000

  This Task's ASID/TCB: 00B3/009C5528
```

Bit 16 & 17 = 00 -> Primary mode

Program check

Failing TCB

Note cross memory environment

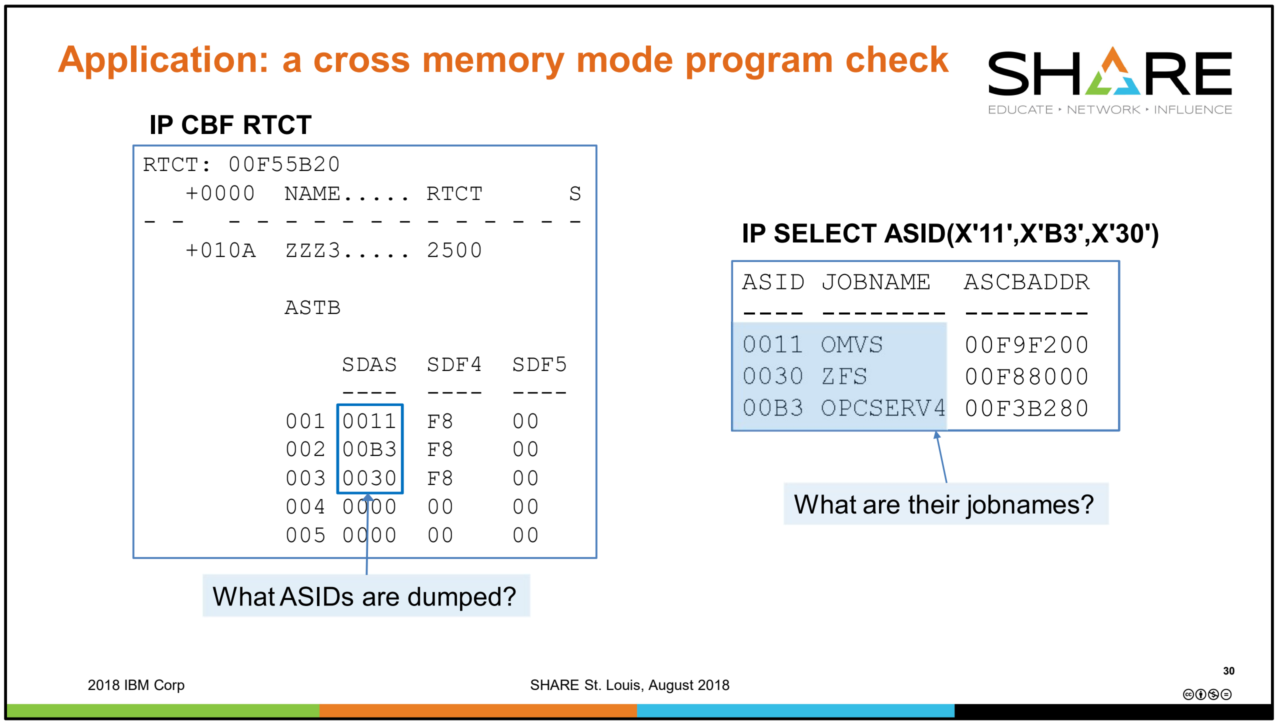
2018 IBM Corp   SHARE St. Louis, August 2018   29

This is Time of Error Information from a dump of a program check, an ABEND0C4 PIC10.

Note the PSW ASC bits. These are PSW bits 16 and 17, that is, the first 2 bits of the 5th nibble. This nibble contains a X’1’ = B’0001’. The first 2 bits are B’00’ which indicates Primary Mode. This means that both the instruction fetch and the data references are to the primary address space.

Underneath the registers at time of error we see the cross memory environment at time of error. The Home address space is ASID X’B3’. The primary and secondary ASIDs are both ASID X’11’. Remember that home is where the unit of work started off life, its original ASID, its job. Primary is the address space in which the unit of work is currently executing. The error occurred in primary mode, so it occurred when trying to access storage in the primary address space.

It is also handy to note the address of the failing TCB. The TCB lives in the home address space.



Application: a cross memory mode program check

SHARE EDUCATE • NETWORK • INFLUENCE

IP CBF RTCT

```
RTCT: 00F55B20
   +0000  NAME..... RTCT      S
- -   - - - - - - - - - - - - -
   +010A  ZZZ3..... 2500

          ASTB

              SDAS  SDF4  SDF5
              ----  ----  ----
          001 0011  F8    00
          002 00B3  F8    00
          003 0030  F8    00
          004 0000  00    00
          005 0000  00    00
```

What ASIDs are dumped?

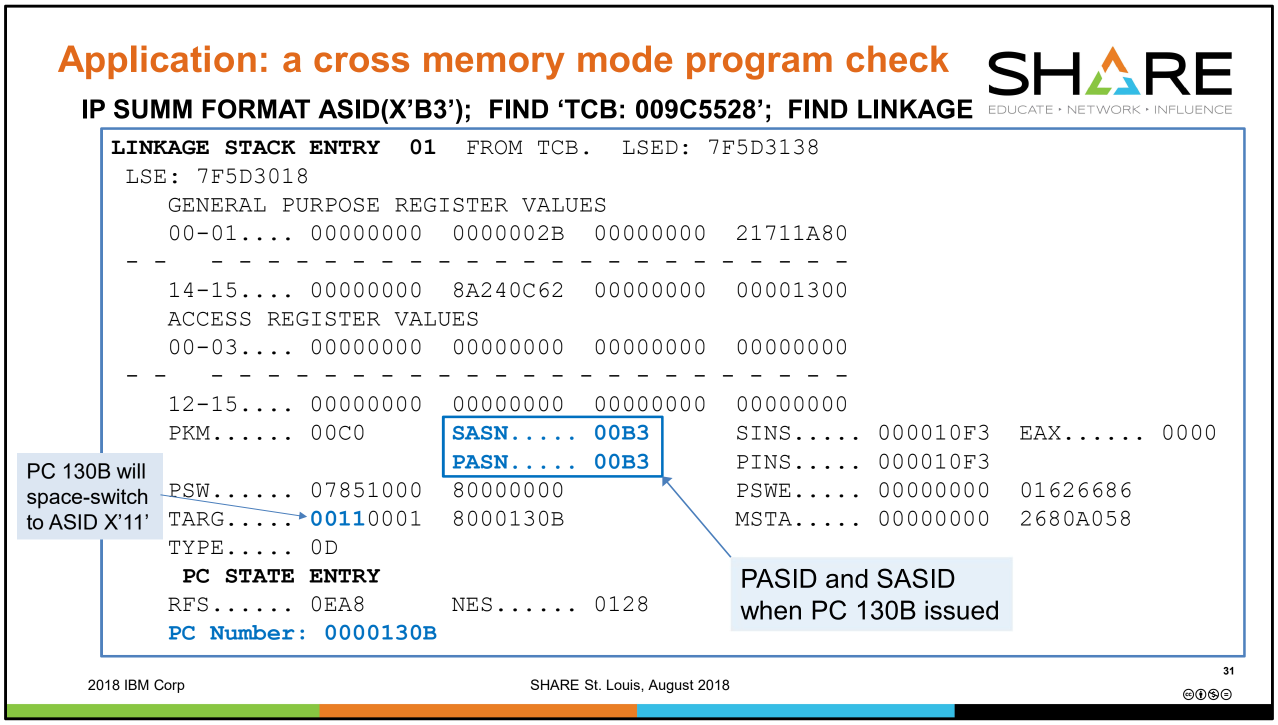
IP SELECT ASID(X'11',X'B3',X'30')

```
ASID JOBNAME   ASCBADDR
---- --------  --------
0011 OMVS      00F9F200
0030 ZFS       00F88000
00B3 OPCSERV4  00F3B280
```

What are their jobnames?

2018 IBM Corp SHARE St. Louis, August 2018 30

The ‘IP CBF RTCT’ command identifies the ASIDs included in an SVC dump. The ‘IP SELECT’ command allows us to associate a JOBNAME with each ASID. ‘IP SELECT ALL’ can be used to show all active ASIDs and their associated JOBNAMEs.



Application: a cross memory mode program check

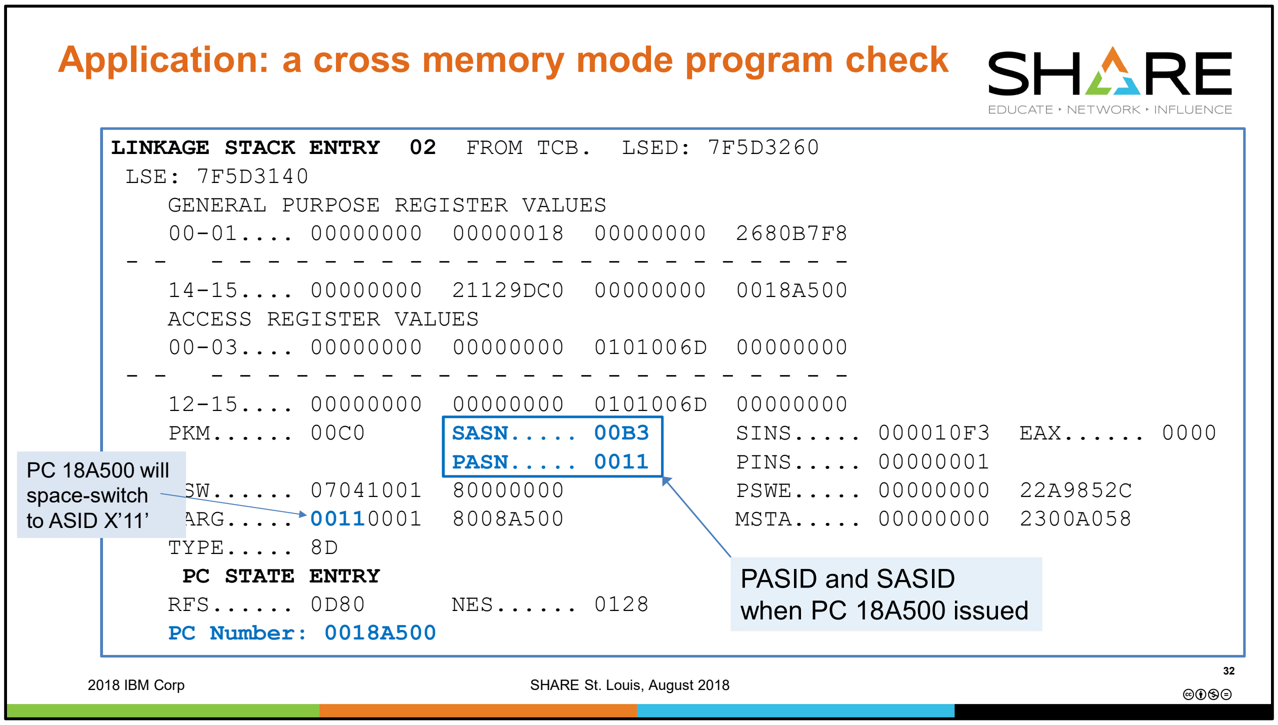
IP SUMM FORMAT ASID(X'B3'); FIND 'TCB: 009C5528'; FIND LINKAGE

```
LINKAGE STACK ENTRY  01  FROM TCB.  LSED: 7F5D3138
 LSE: 7F5D3018
    GENERAL PURPOSE REGISTER VALUES
    00-01.... 00000000  0000002B  00000000  21711A80
 - -   - - - - - - - - - - - - - - - - - - - - - - -
    14-15.... 00000000  8A240C62  00000000  00001300
    ACCESS REGISTER VALUES
    00-03.... 00000000  00000000  00000000  00000000
 - -   - - - - - - - - - - - - - - - - - - - - - - -
    12-15.... 00000000  00000000  00000000  00000000
    PKM...... 00C0      SASN..... 00B3      SINS..... 000010F3  EAX...... 0000
                        PASN..... 00B3      PINS..... 000010F3
    PSW...... 07851000  80000000            PSWE..... 00000000  01626686
    TARG..... 00110001  8000130B            MSTA..... 00000000  2680A058
    TYPE..... 0D
     PC STATE ENTRY
    RFS...... 0EA8      NES...... 0128
    PC Number: 0000130B
```

PC 130B will space-switch to ASID X'11'
PASID and SASID when PC 130B issued

The first linkage stack entry (LSE) under the error TCB shows a PC 130B being issued. At the time of the PC 130B being issued, HASID=PASID=SASID=X'B3'. As a result of the PC 130B being issued, a space switch occurs into ASID X'11'. Following the issuance of the PC, HASID=B3, PASID=11, and SASID=B3.

To determine the address of the PC routine which receives control in ASID 11 as a result of the PC 130B, consult the PC Information Table near the top of the SUMMARY FORMAT report.



Application: a cross memory mode program check

```
LINKAGE STACK ENTRY  02  FROM TCB.  LSED: 7F5D3260
 LSE: 7F5D3140
    GENERAL PURPOSE REGISTER VALUES
    00-01.... 00000000  00000018  00000000  2680B7F8
 - -   - - - - - - - - - - - - - - - - - - - - - - -
    14-15.... 00000000  21129DC0  00000000  0018A500
    ACCESS REGISTER VALUES
    00-03.... 00000000  00000000  0101006D  00000000
 - -   - - - - - - - - - - - - - - - - - - - - - - -
    12-15.... 00000000  00000000  0101006D  00000000
    PKM...... 00C0      SASN..... 00B3      SINS..... 000010F3  EAX...... 0000
                        PASN..... 0011      PINS..... 00000001
     SW...... 07041001  80000000            PSWE..... 00000000  22A9852C
     ARG..... 00110001  8008A500            MSTA..... 00000000  2300A058
    TYPE..... 8D
     PC STATE ENTRY
    RFS...... 0D80      NES...... 0128
    PC Number: 0018A500
```

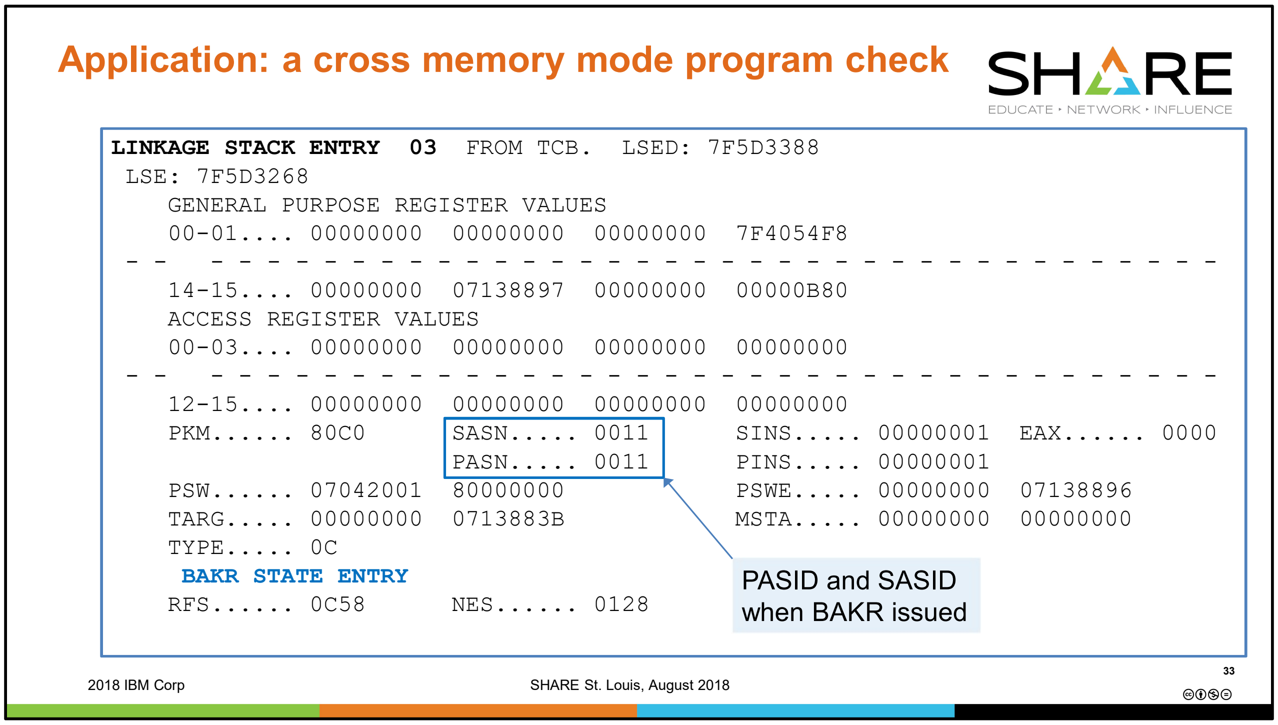
PC 18A500 will space-switch to ASID X'11'

PASID and SASID when PC 18A500 issued

The second linkage stack entry under the error TCB shows a PC 18A500 being issued. At the time of the PC 18A500 being issued, HASID=B3, PASID=11, and SASID=X'B3'. As a result of the PC 18A500 being issued, a space switch occurs into ASID X'11'. Since the unit of work was already running with PASID=11, this does not cause a change of the primary address space; however it does cause the SASID to be updated to become ASID 11.

Following the issuance of the PC 18A500, HASID=B3, PASID=11, and SASID=11.

In last line of this LSE, we see for the first time a "big" PC number, that is, one that is greater than X'0007FFFF'. This represents an architectural change introduced when the ASN-and-LX-Reuse facility was implemented. The X'00080000' bit in this word represents a flag which is part of the uncompressed PC number. You can find the uncompressed PC number in the IPCS SUMMARY FORMAT PC Information Table and in system trace table PC entries, in addition to on this "PC Number:" line (if present) of the LSE. However the second word of the LSE's TARG field will contain the compressed PC number. To derive the compressed PC number from the uncompressed PC number, turn off the flag bit, and shift all bits to the left of it right by 1 bit position. In the example above, 0018A500 becomes 0010A500 when the flag bit is turned off, then shifting X'001' right by 1 bit results in 0008A500. As another example, uncompressed PC number 00490113 becomes 00410113 when the flag bit is turned off, then shifting X'004' right by 1 bit results in 00210113.

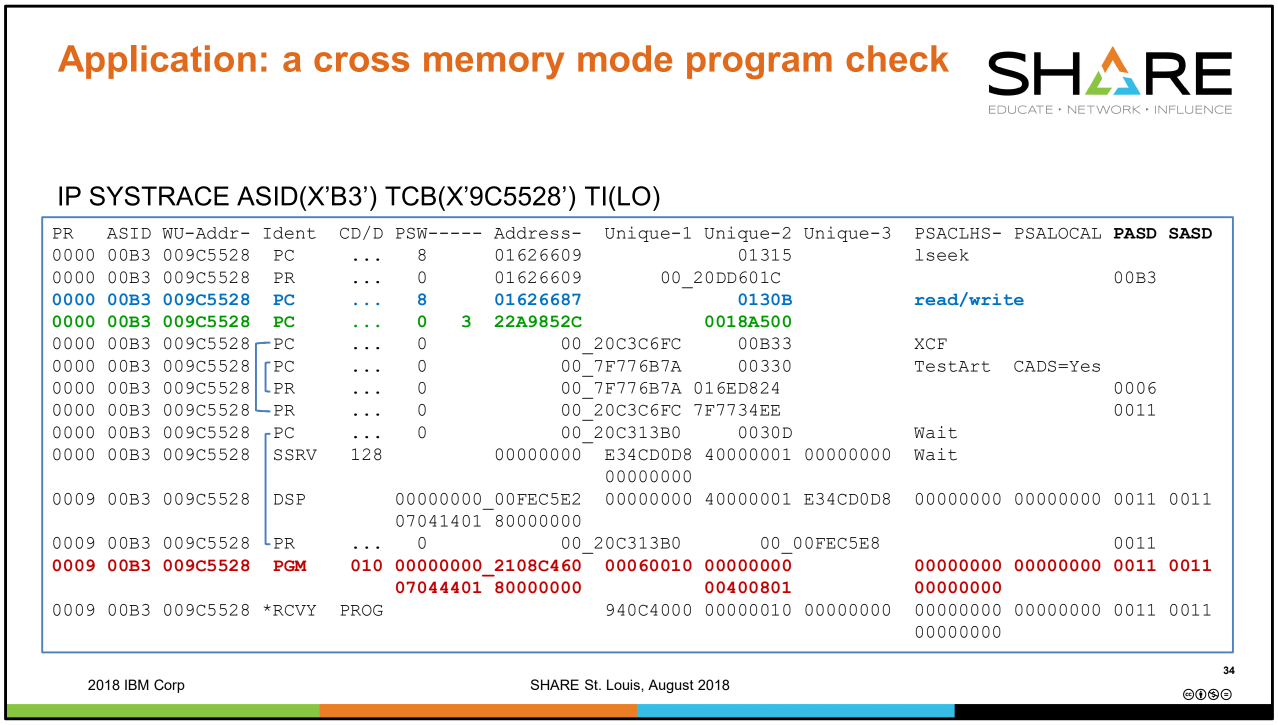


Application: a cross memory mode program check

```
LINKAGE STACK ENTRY  03  FROM TCB.  LSED: 7F5D3388
 LSE: 7F5D3268
    GENERAL PURPOSE REGISTER VALUES
    00-01.... 00000000  00000000  00000000  7F4054F8
- -   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    14-15.... 00000000  07138897  00000000  00000B80
    ACCESS REGISTER VALUES
    00-03.... 00000000  00000000  00000000  00000000
- -   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    12-15.... 00000000  00000000  00000000  00000000
    PKM...... 80C0      SASN..... 0011      SINS..... 00000001  EAX...... 0000
                        PASN..... 0011      PINS..... 00000001
    PSW...... 07042001  80000000            PSWE..... 00000000  07138896
    TARG..... 00000000  0713883B            MSTA..... 00000000  00000000
    TYPE..... 0C
     BAKR STATE ENTRY
    RFS...... 0C58      NES...... 0128
```

The third linkage stack entry under the error TCB was created as a result of a BAKR instruction. This instruction is often issued on entry to a routine for the express purpose of taking advantage of the linkage stack as a status save area. At the time of the BAKR being issued, HASID=B3, PASID=11, and SASID=11. A BAKR instruction does not affect Home, Primary, and Secondary.

The TARG field indicates the address where the BAKR was issued.



Application: a cross memory mode program check

IP SYSTRACE ASID(X’B3’) TCB(X’9C5528’) TI(LO)

```
PR   ASID WU-Addr- Ident  CD/D PSW----- Address-  Unique-1 Unique-2 Unique-3   PSACLHS- PSALOCAL PASD SASD
0000 00B3 009C5528  PC     ...  8       01626609            01315              lseek
0000 00B3 009C5528  PR     ...  0       01626609        00_20DD601C                              00B3
0000 00B3 009C5528  PC     ...  8       01626687            0130B              read/write
0000 00B3 009C5528  PC     ...  0   3   22A9852C          0018A500
0000 00B3 009C5528┌─PC     ...  0             00_20C3C6FC      00B33           XCF
0000 00B3 009C5528│┌PC     ...  0             00_7F776B7A      00330           TestArt  CADS=Yes
0000 00B3 009C5528│└PR     ...  0             00_7F776B7A 016ED824                                0006
0000 00B3 009C5528└─PR     ...  0             00_20C3C6FC 7F7734EE                                0011
0000 00B3 009C5528 ┌PC     ...  0             00_20C313B0      0030D           Wait
0000 00B3 009C5528 │SSRV   128              00000000  E34CD0D8 40000001 00000000   Wait
                   │                                  00000000
0009 00B3 009C5528 │DSP         00000000_00FEC5E2   00000000 40000001 E34CD0D8   00000000 00000000 0011 0011
                   │            07041401 80000000
0009 00B3 009C5528 └PR     ...  0             00_20C313B0        00_00FEC5E8                      0011
0009 00B3 009C5528  PGM    010 00000000_2108C460  00060010 00000000            00000000 00000000 0011 0011
                               07044401 80000000           00400801            00000000
0009 00B3 009C5528 *RCVY   PROG                   940C4000 00000010 00000000   00000000 00000000 0011 0011
                                                                               00000000
```

2018 IBM Corp SHARE St. Louis, August 2018 34

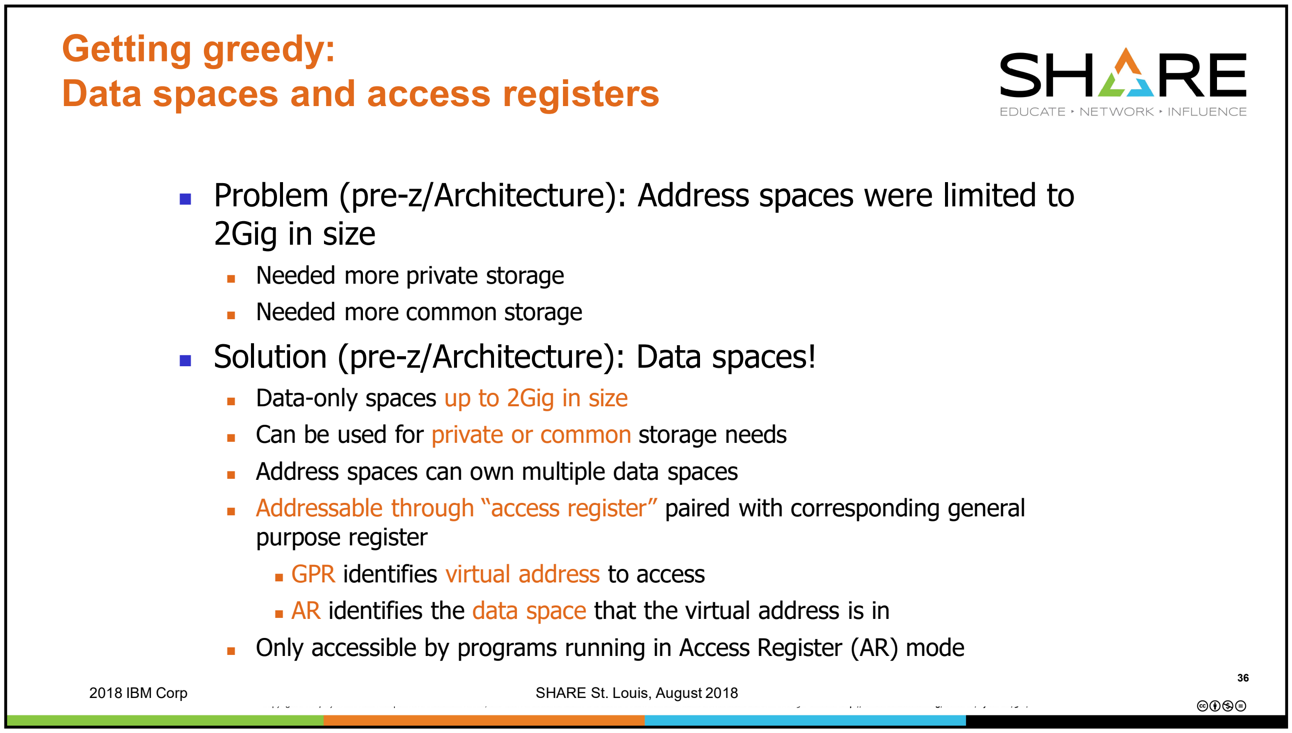
Note that we’ve formatted the system trace table using the ASID and the TCB filter so that we can focus just on activity under our error TCB. We can see the PC 130B and the PC 18A500, which we saw on our linkage stack, in our system trace table. They have no matching PR because processing as still going on within this PC routine environment at the time that the program check occurred.

A few PC’s have been issued after the PC 18A500, which was the last seen on the linkage stack. We don’t see these PC’s on the linkage stack because the PC routines have completed and issued the PR, which removes the entry from the linkage stack.

Note that some trace table entries trace the PASID and/or SASID at the time the event occurred.

Since linkage stack entries contain PSW and register information, they are very valuable for debugging!

EXTENDED ADDRESSABILITY



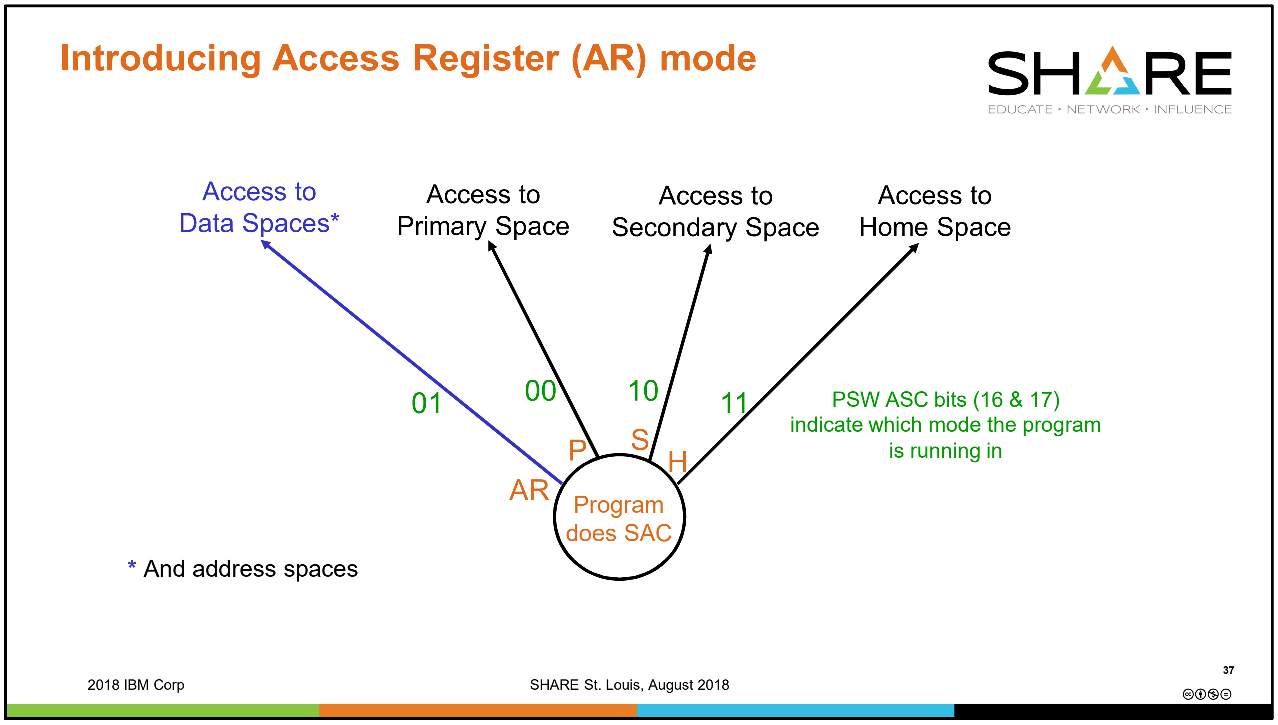
Getting greedy: Data spaces and access registers

- Problem (pre-z/Architecture): Address spaces were limited to 2Gig in size
  - Needed more private storage
  - Needed more common storage
- Solution (pre-z/Architecture): Data spaces!
  - Data-only spaces up to 2Gig in size
  - Can be used for private or common storage needs
  - Address spaces can own multiple data spaces
  - Addressable through "access register" paired with corresponding general purpose register
    - GPR identifies virtual address to access
    - AR identifies the data space that the virtual address is in
  - Only accessible by programs running in Access Register (AR) mode

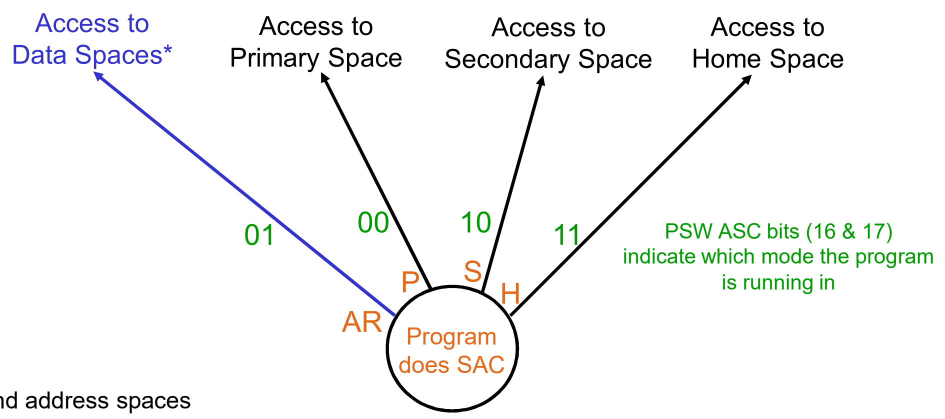
2018 IBM Corp SHARE St. Louis, August 2018 36

In the world prior to z/Architecture (pre-Year2000), address spaces were 2Gig in size. Jobs often found themselves needing more storage. The solution was to obtain data spaces to provide additional data repositories beyond the address space's private storage. Data spaces can also be used to hold common storage. Each data space can be up to 2Gig in size, and an address space can have many data spaces.

Access to a data space is accomplished via an access register. We will see that, when running with the PSW ASC bits indicating AR mode, an Access Register is used in conjunction with its corresponding General Purpose Register any time an instruction is executed that has that GPR as a base register. For example, a LOAD off of base register 6 would use AR6 as part of the translation. The GPR would indicate the virtual address to be accessed. The AR would indicate what space that virtual address was in.



Introducing Access Register (AR) mode



PSW ASC bits (16 & 17) indicate which mode the program is running in

* And address spaces

Dramatization via chat: Setting up a data space

yo zos rsm services, i was told to talk to u about getting some storage

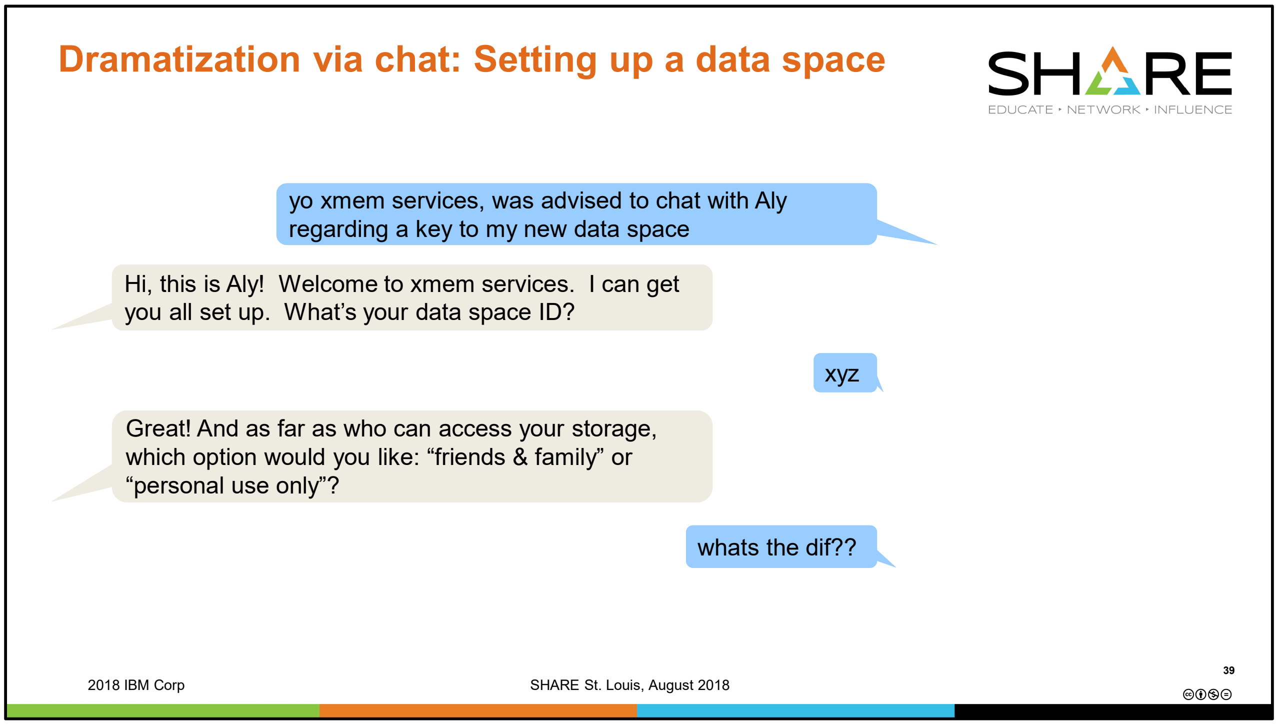
You've come to the right place, data spaces are us! Common or private? What size did you have in mind?

pvt, max i can get

Gotcha, 2gig it is, brb

Ok I have your data space ready. Its ID is "xyz". Now you just need to chat with Aly on the xmem service team. Give her this ID and she'll set you up with some buyer protection and give you the key to your new data space.

k ty!



Dramatization via chat: Setting up a data space

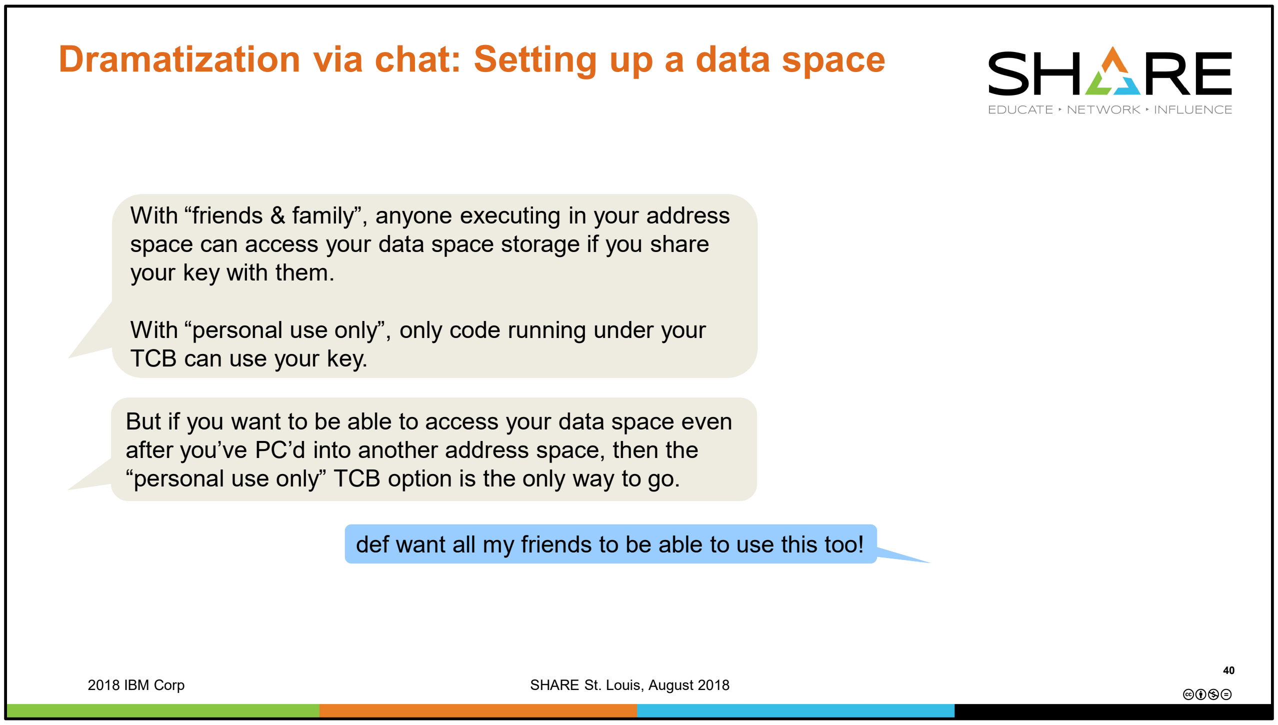
yo xmem services, was advised to chat with Aly regarding a key to my new data space

Hi, this is Aly! Welcome to xmem services. I can get you all set up. What's your data space ID?

xyz

Great! And as far as who can access your storage, which option would you like: "friends & family" or "personal use only"?

whats the dif??



Dramatization via chat: Setting up a data space

With “friends & family”, anyone executing in your address space can access your data space storage if you share your key with them.

With “personal use only”, only code running under your TCB can use your key.

But if you want to be able to access your data space even after you’ve PC’d into another address space, then the “personal use only” TCB option is the only way to go.

def want all my friends to be able to use this too!

Dramatization via chat: Setting up a data space

Ok, “friends & family” it is! One moment please while I get this set up for you.

Ok, I’ve set up your protection and activated your key. Your key code is 01xx00yy. To use it, just have your program place it in an access register, set up the corresponding general purpose register to point to the address you want to access, and use that register as the base register of an assembler instruction such as a LOAD, STORE, TM, MVC … any instruction that has a base reg!

Don’t forget you must be in AR mode for this to work!

awesome thx much!

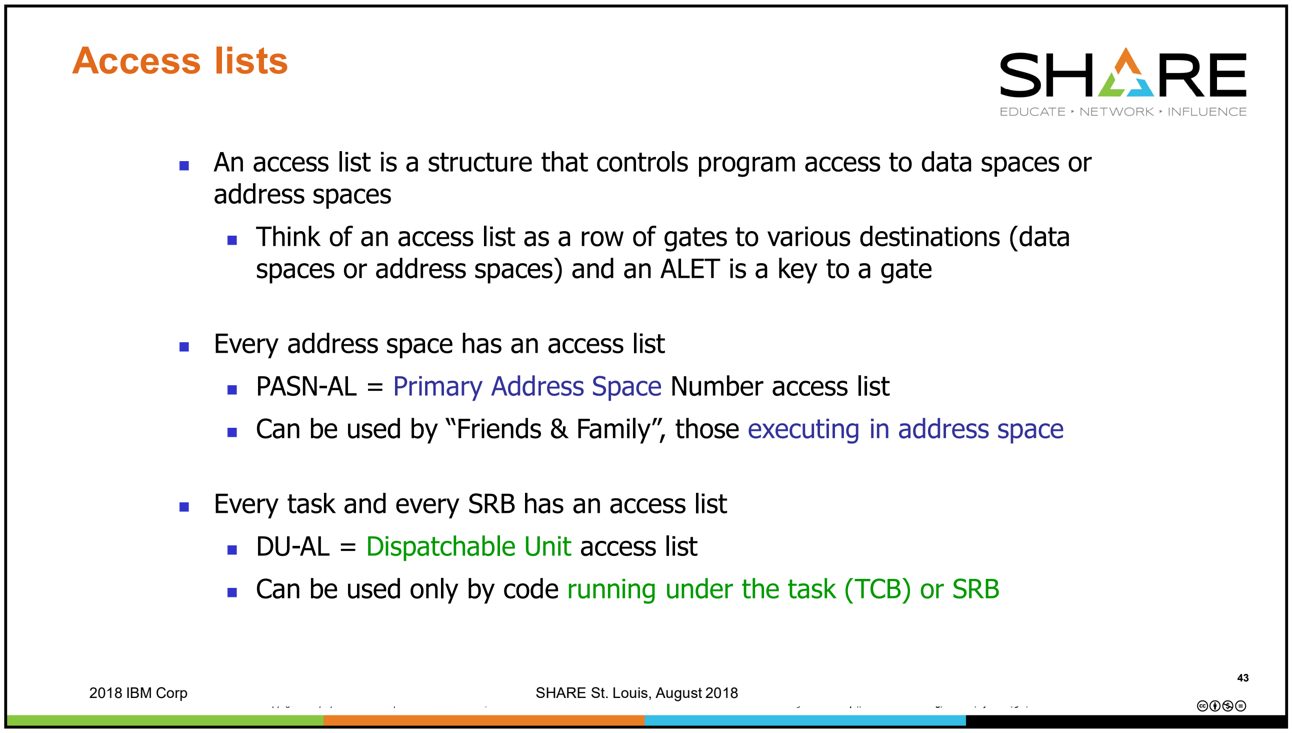
Dramatization via chat: Setting up a data space

SHARE
EDUCATE • NETWORK • INFLUENCE

You're very welcome. Is there anything else I can help you with today?

nope ty, u've been great, working with zos service is always a fantastic experience, u guys rock!

😇😉👍😀



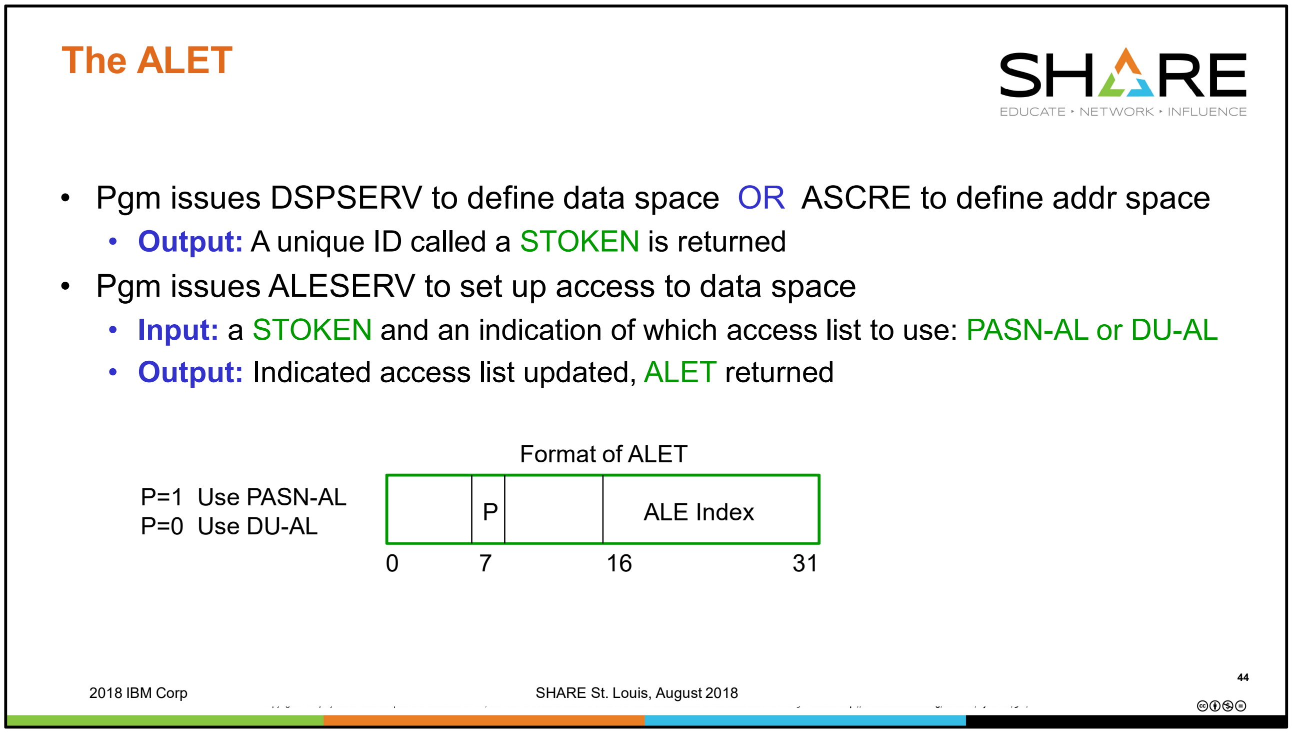
Access lists

- An access list is a structure that controls program access to data spaces or address spaces
  - Think of an access list as a row of gates to various destinations (data spaces or address spaces) and an ALET is a key to a gate
- Every address space has an access list
  - PASN-AL = Primary Address Space Number access list
  - Can be used by "Friends & Family", those executing in address space
- Every task and every SRB has an access list
  - DU-AL = Dispatchable Unit access list
  - Can be used only by code running under the task (TCB) or SRB

2018 IBM Corp SHARE St. Louis, August 2018 43

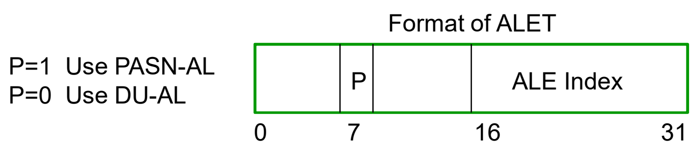
Every address space has an access list that can be 500+ entries in size, representing the theoretical ability to set up access to 500+ data spaces/address spaces for programs executing within a specific address space. The address space-related access list is called a PASN Access List.

Every TCB and SRB on the system also has an associated access list. Only programs running under this unit of work can use that access list.



The ALET

- Pgm issues DSPSERV to define data space OR ASCRE to define addr space
  - **Output:** A unique ID called a STOKEN is returned
- Pgm issues ALESERV to set up access to data space
  - **Input:** a STOKEN and an indication of which access list to use: PASN-AL or DU-AL
  - **Output:** Indicated access list updated, ALET returned



2018 IBM Corp SHARE St. Louis, August 2018 44

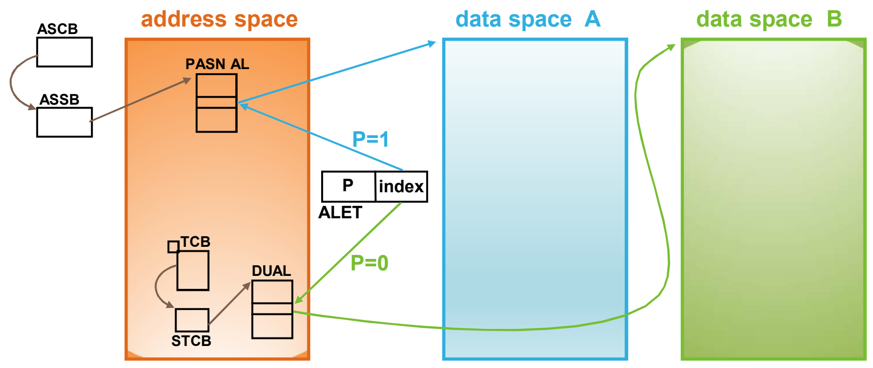
Bit 7 of an ALET is used to determine which access list to use – PASN-AL or DU-AL. The low half of the ALET is the index into the access list.

Examples:

ALET 01000014 – P bit on, use PASN-AL, index is x’0014’

ALET 0000002E – P bit off, use DU-AL, index is x’002E’

Translating the ALET

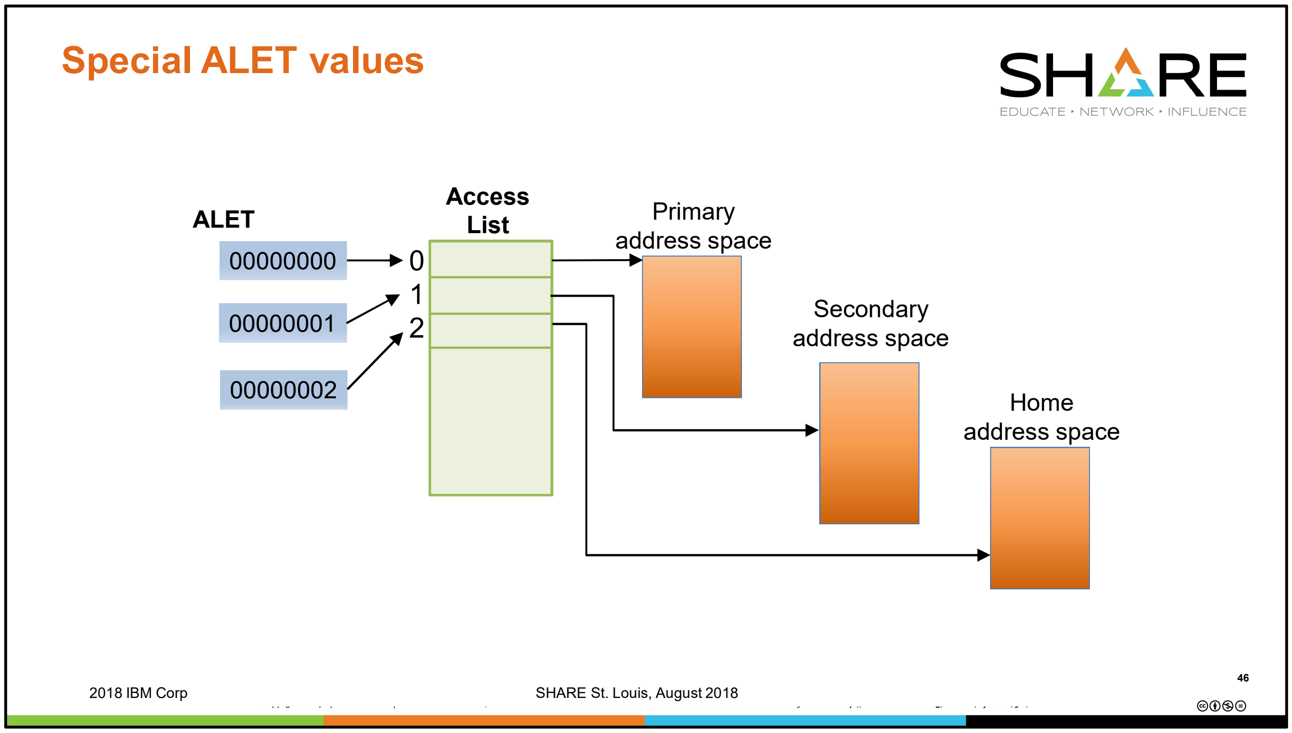


Bit 7 of an ALET, the P bit, determines which access list the ALET is associated with.

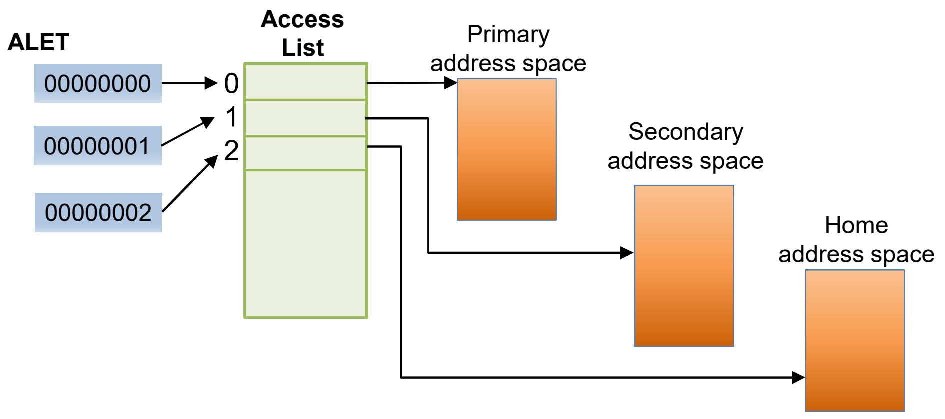
The low half-word of the ALET is the index into the designated access list.

The same ALET index can reach 2 entirely different dataspaces through the 2 different access lists, determined by bit 7 in the ALET.

A program running in AR mode can use an ALET to address a data space through either the PASN access list or the DUAL access list, depending on the content of the ALET.

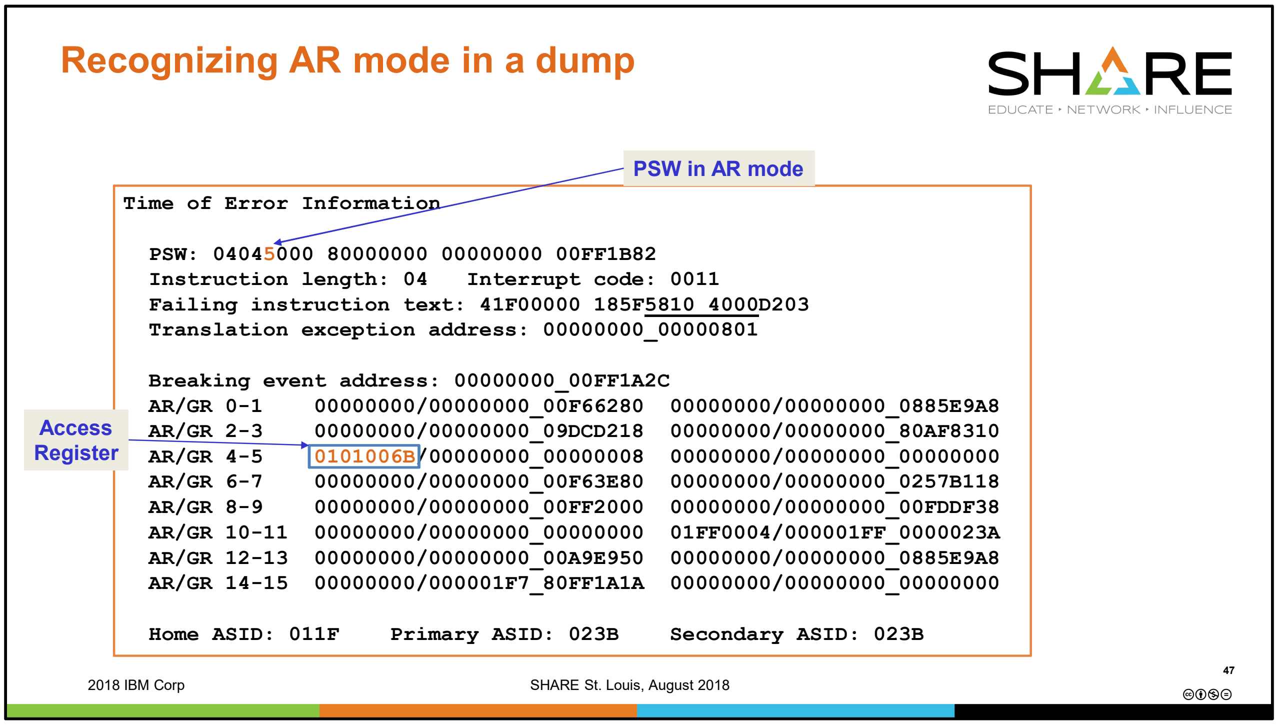


Special ALET values



When executing in AR mode (PSW ASC bits ’01’), a program can access PRIMARY, SECONDARY, and HOME by loading access registers (ARs) with 00000000, 00000001, or 00000002, respectively. All access lists are initialized so that the first 3 entries (index 0, 1 and 2) will reach these address spaces.

Consider a program running initially with empty (zero) access registers – a good programming practice. The program SACs to AR mode. All data references will occur in the current primary address space (where the program is executing) unless/until specific ARs are loaded with ALETs for addressability to other spaces.

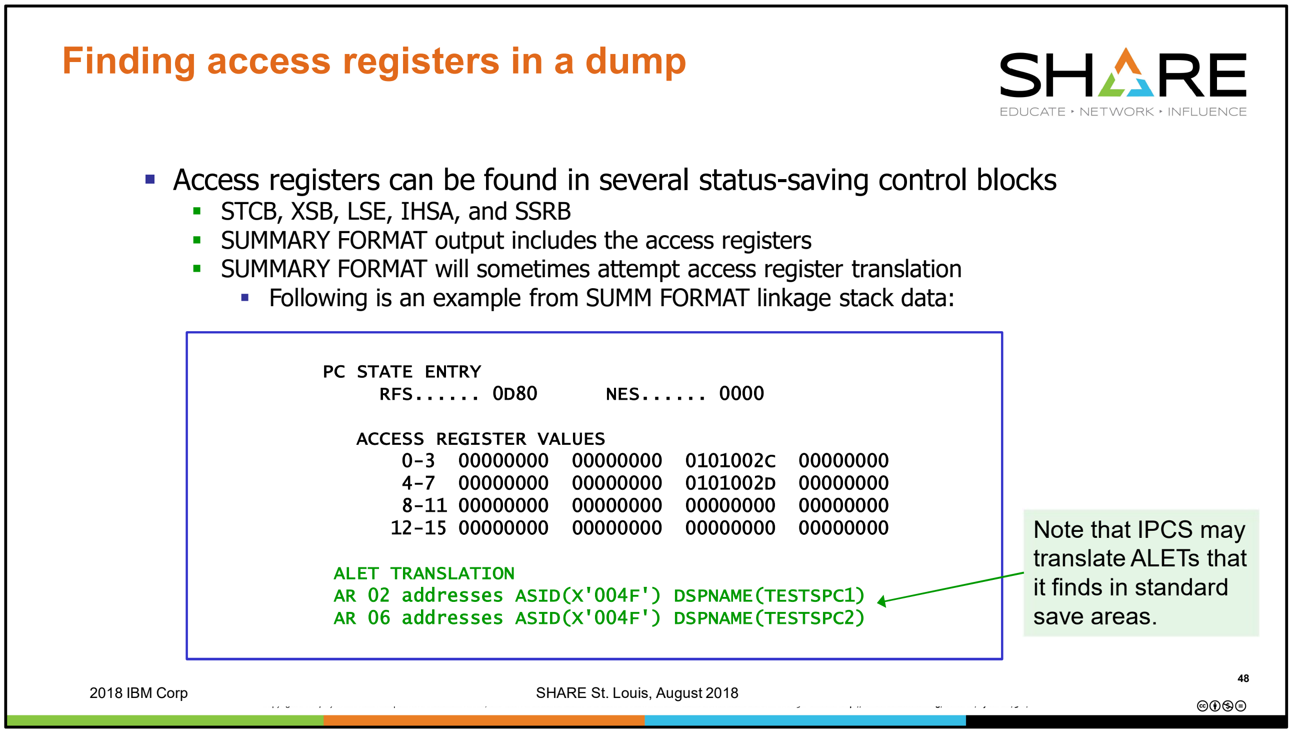


Recognizing AR mode in a dump

PSW in AR mode
Access Register

```
Time of Error Information

  PSW: 04045000 80000000 00000000 00FF1B82
  Instruction length: 04   Interrupt code: 0011
  Failing instruction text: 41F00000 185F5810 4000D203
  Translation exception address: 00000000_00000801

  Breaking event address: 00000000_00FF1A2C
  AR/GR 0-1    00000000/00000000_00F66280  00000000/00000000_0885E9A8
  AR/GR 2-3    00000000/00000000_09DCD218  00000000/00000000_80AF8310
  AR/GR 4-5    0101006B/00000000_00000008  00000000/00000000_00000000
  AR/GR 6-7    00000000/00000000_00F63E80  00000000/00000000_0257B118
  AR/GR 8-9    00000000/00000000_00FF2000  00000000/00000000_00FDDF38
  AR/GR 10-11  00000000/00000000_00000000  01FF0004/000001FF_0000023A
  AR/GR 12-13  00000000/00000000_00A9E950  00000000/00000000_0885E9A8
  AR/GR 14-15  00000000/000001F7_80FF1A1A  00000000/00000000_00000000

  Home ASID: 011F    Primary ASID: 023B    Secondary ASID: 023B
```

In this example of an ABEND0C4 PIC11 error, the PSW has bits 16-17 = 01 indicating the program is running in AR mode. This means that when a general purpose register indicating a virtual address to be accessed is used as a base register, the corresponding access register will be used to indicate the space (address space or data space) that the virtual address is in. In this case we see that the failing instruction used a base register of 4, so the ALET in AR4 was used as part of the translation. The ALET has the P bit (bit 7) on and so will translate against the PASN access list.



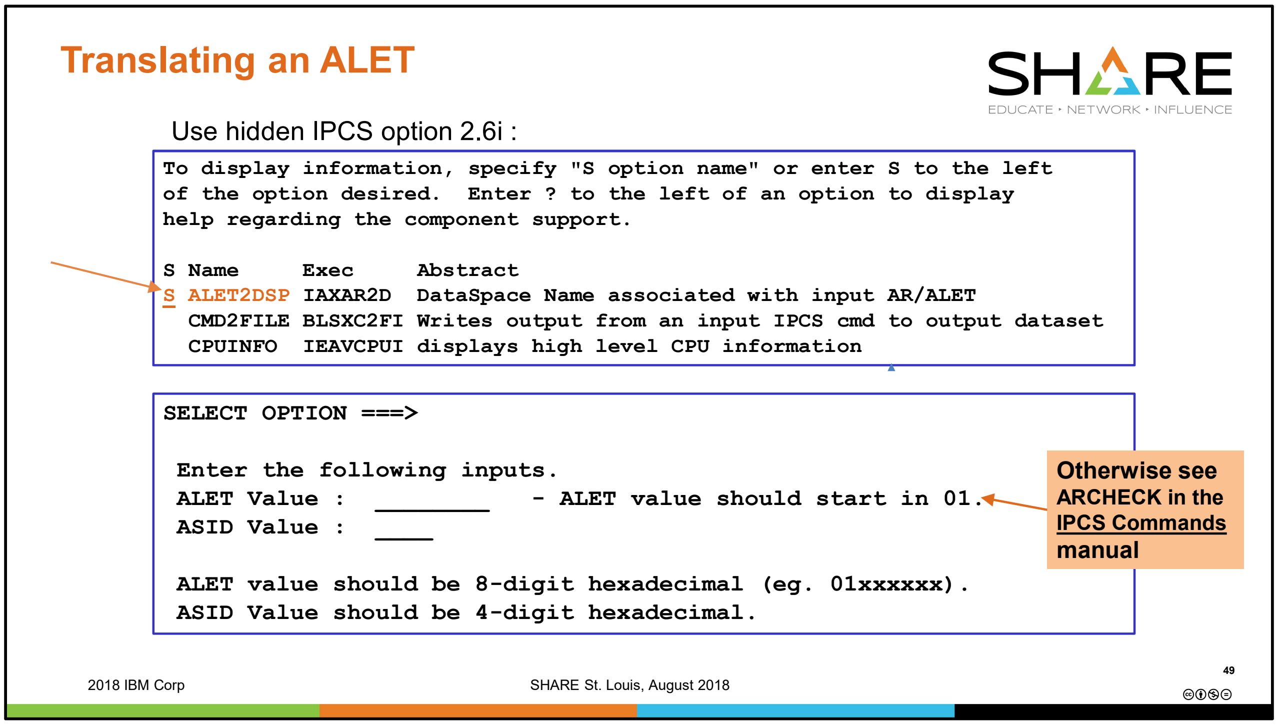
Finding access registers in a dump

- Access registers can be found in several status-saving control blocks
  - STCB, XSB, LSE, IHSA, and SSRB
  - SUMMARY FORMAT output includes the access registers
  - SUMMARY FORMAT will sometimes attempt access register translation
    - Following is an example from SUMM FORMAT linkage stack data:

```
            PC STATE ENTRY
                 RFS...... 0D80      NES...... 0000

               ACCESS REGISTER VALUES
                   0-3  00000000  00000000  0101002C  00000000
                   4-7  00000000  00000000  0101002D  00000000
                   8-11 00000000  00000000  00000000  00000000
                  12-15 00000000  00000000  00000000  00000000

             ALET TRANSLATION
             AR 02 addresses ASID(X'004F') DSPNAME(TESTSPC1)
             AR 06 addresses ASID(X'004F') DSPNAME(TESTSPC2)
```

Note that IPCS may translate ALETs that it finds in standard save areas.

IPCS will attempt to translate ALETs to a data space name in several IPCS reports such as SUMMARY FORMAT.



Translating an ALET

Use hidden IPCS option 2.6i :

```
To display information, specify "S option name" or enter S to the left
of the option desired.  Enter ? to the left of an option to display
help regarding the component support.

S Name     Exec     Abstract
S ALET2DSP IAXAR2D  DataSpace Name associated with input AR/ALET
  CMD2FILE BLSXC2FI Writes output from an input IPCS cmd to output dataset
  CPUINFO  IEAVCPUI displays high level CPU information
```

```
SELECT OPTION ===>

 Enter the following inputs.
 ALET Value :  _________    - ALET value should start in 01.
 ASID Value :  _____

 ALET value should be 8-digit hexadecimal (eg. 01xxxxxx).
 ASID Value should be 4-digit hexadecimal.
```

Otherwise see ARCHECK in the IPCS Commands manual

2018 IBM Corp SHARE St. Louis, August 2018 49

An undocumented (but safe!) REXX exec called ALET2DSP will help you easily translate an ALET for the PASN access list to its corresponding space. However this REXX exec does not handle ALETs for a DUAL access list. For DUAL ALETs, you will need to refer to the IP ARCHECK command. When passing a DUAL ALET to the ARCHECK command, you need to supply the ALET, the home ASID, and the address of the DUAL as pointed to by STCBALOV. Here is an example:

ARCHECK ALET(X'0001002B') ADDR(7F6C1600) ASID(X'3C') STR(ACCESSLIST)

ARCHECK can also be used to translate PASN AL ALETs. However in that case you will need to provide the address of the PASN AL as pointed to by ASSBPALV, along with the primary ASID.

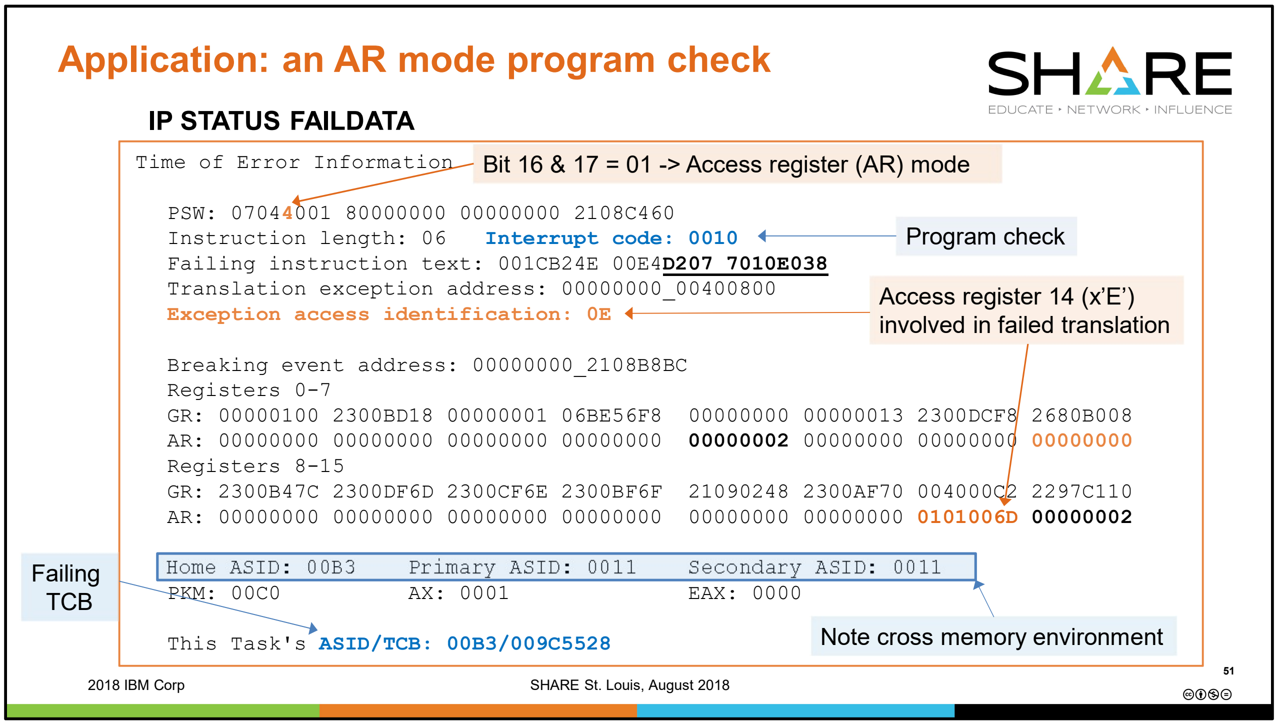
Displaying data space storage

- **To display data space storage....**
  - Issue: **IPCS LIST xxxxxxxx LEN(X'ww') ASID(X'yy') DSPNAME(zzzzzzzz)**
    - **xxxxxxxx** is the virtual storage address to be viewed
    - **ww** is the length of storage to be viewed
    - **zzzzzzzz** is the name of the data space
    - **yy** is the ASID of the address space with which the data space is associated

OR

  - Use IPCS option 1 Browse:

```
  PTR    Address              Address space
00001  0000B000.           ASID(X'001D') DSPNAME(MYSPACE)
```



Application: an AR mode program check

IP STATUS FAILDATA

```
Time of Error Information

  PSW: 07044001 80000000 00000000 2108C460
  Instruction length: 06   Interrupt code: 0010
  Failing instruction text: 001CB24E 00E4D207 7010E038
  Translation exception address: 00000000_00400800
  Exception access identification: 0E

  Breaking event address: 00000000_2108B8BC
  Registers 0-7
  GR: 00000100 2300BD18 00000001 06BE56F8  00000000 00000013 2300DCF8 2680B008
  AR: 00000000 00000000 00000000 00000000  00000002 00000000 00000000 00000000
  Registers 8-15
  GR: 2300B47C 2300DF6D 2300CF6E 2300BF6F  21090248 2300AF70 004000C2 2297C110
  AR: 00000000 00000000 00000000 00000000  00000000 00000000 0101006D 00000002

  Home ASID: 00B3    Primary ASID: 0011    Secondary ASID: 0011
  PKM: 00C0          AX: 0001              EAX: 0000

  This Task's ASID/TCB: 00B3/009C5528
```

Bit 16 & 17 = 01 -> Access register (AR) mode

Program check

Access register 14 (x'E') involved in failed translation

Failing TCB

Note cross memory environment

2018 IBM Corp

SHARE St. Louis, August 2018

51

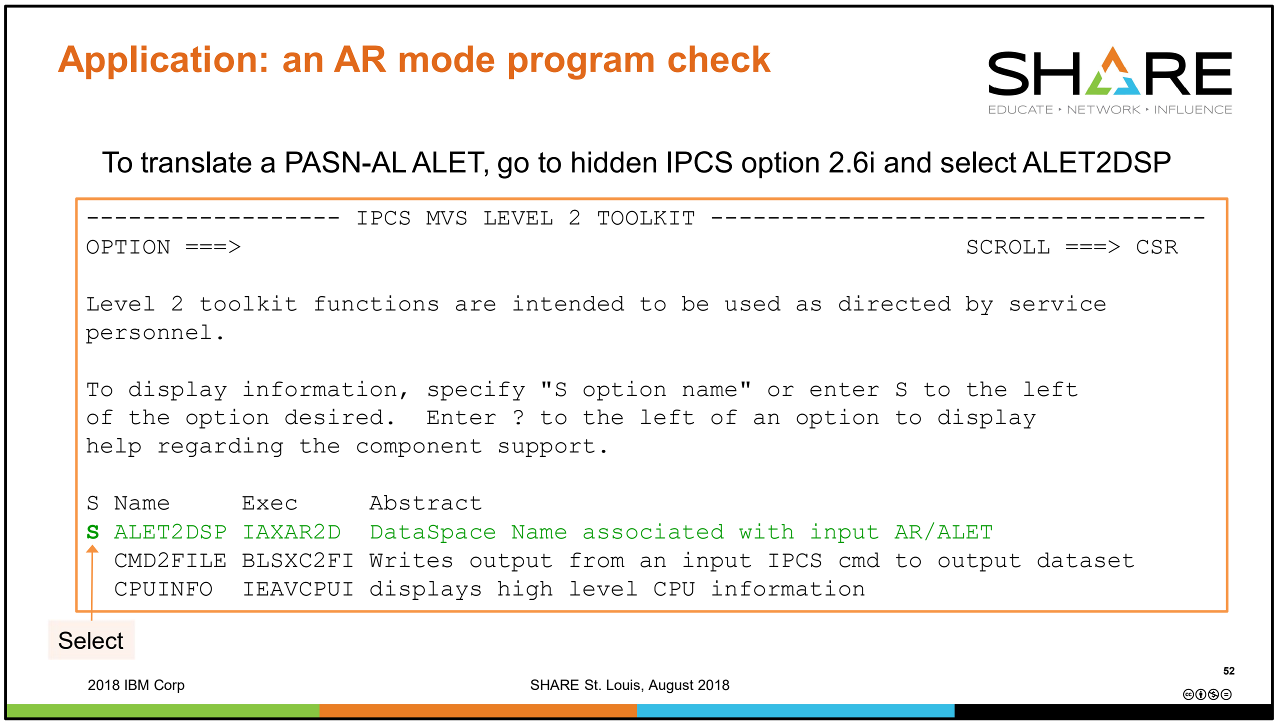
This is Time of Error Information from a dump of a program check, an ABEND0C4 PIC10.

Note the PSW ASC bits. These are PSW bits 16 and 17, that is, the first 2 bits of the 5th nibble. This nibble contains a X'4' = B'0100'. The first 2 bits are B'01' which indicates Access Register Mode. This means that the instruction fetch is from primary and the data references are determined by the ALET in the access register(s) corresponding to the instruction's base register(s). Access register 14 is identified as being involved in the translation failure.

Note that some access registers contain 00000002. An ALET of 00000002 indicates the home address space. If the access register corresponding to an instruction's base register contains an ALET of 00000002, then the storage access will be to the address indicated by the general purpose base register in the home address space.

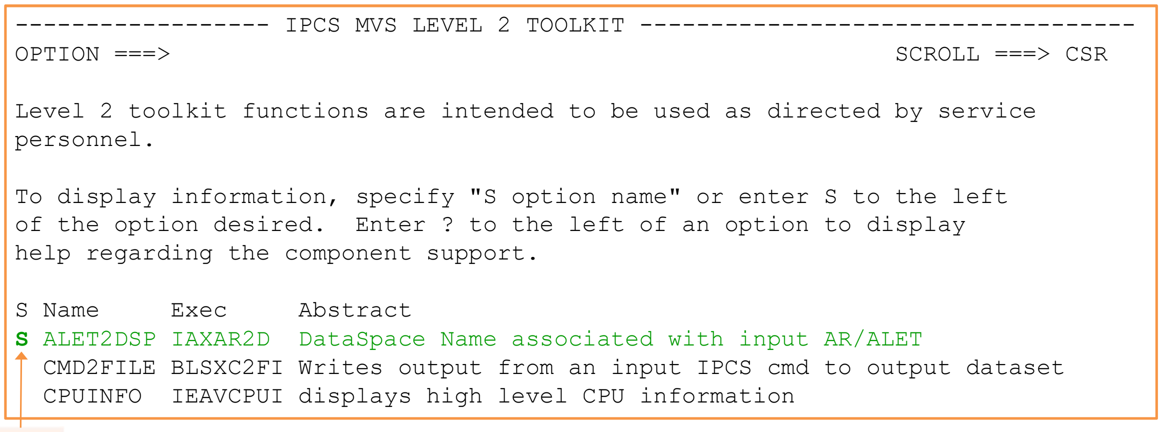
Many access registers contain 00000000, which indicates the primary address space.

Beneath the registers we see the cross memory environment identified. Home ASID=B3. Primary ASID=11. So use of an ALET of 00000002 accesses ASID B3. Use of an ALET of 00000000 accesses ASID 11. The failing instruction is attempting to move data from the space indicated by the ALET of 0101006D to the primary address space, ASID 11.



Application: an AR mode program check

To translate a PASN-AL ALET, go to hidden IPCS option 2.6i and select ALET2DSP



```
------------------ IPCS MVS LEVEL 2 TOOLKIT -----------------------------------
OPTION ===>                                                        SCROLL ===> CSR

Level 2 toolkit functions are intended to be used as directed by service
personnel.

To display information, specify "S option name" or enter S to the left
of the option desired.  Enter ? to the left of an option to display
help regarding the component support.

S Name     Exec     Abstract
S ALET2DSP IAXAR2D  DataSpace Name associated with input AR/ALET
  CMD2FILE BLSXC2FI Writes output from an input IPCS cmd to output dataset
  CPUINFO  IEAVCPUI displays high level CPU information
```

Select

2018 IBM Corp   SHARE St. Louis, August 2018   52

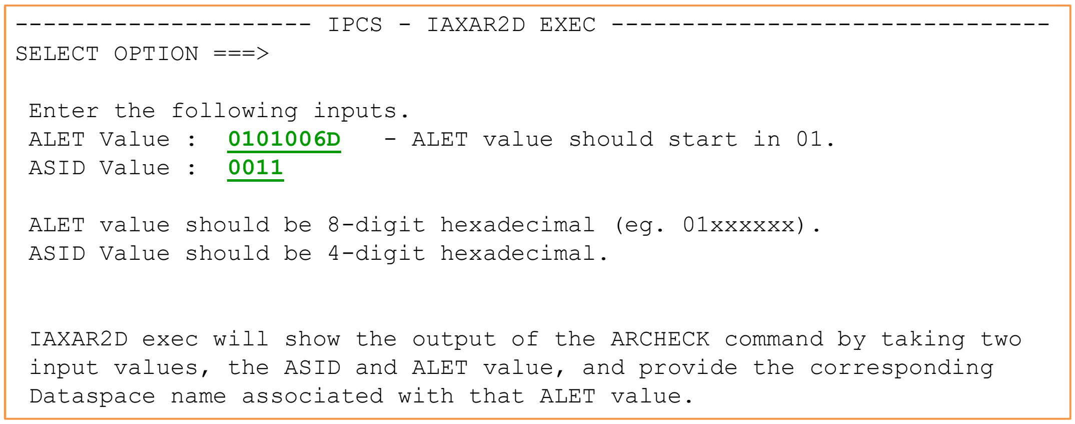
If the P-bit (bit 7 counting from 0) of the ALET is on, then the ALET is a primary ALET and maps to the PASN access list. In this case, exec ALET2DSP can be used to translate the ALET to a space. Choose IPCS option 2.6i and Select ALET2DSP. You will be prompted to enter the ALET and the primary ASID.

Note that IPCS option 2.6i contains many "as-is" rexx execs. These are "lightly" supported, meaning that if a defect is found in the exec, it will not be APAR'd, but it might be addressed in the next release. See August 2017 SHARE presentation "Mining z/OS Debugging Nuggets" for additional information about execs available under option 2.6i:

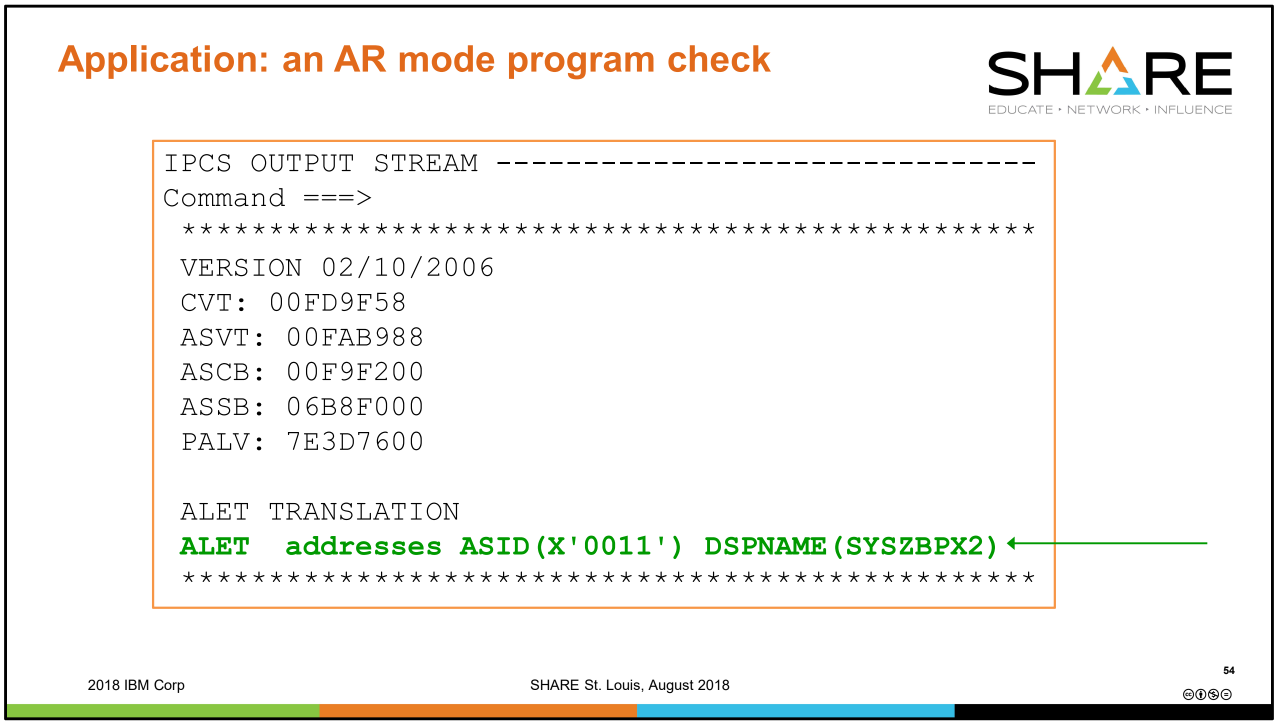
http://events.share.org/Summer2017/Public/SessionDetails.aspx?FromPage=Speakers.aspx&SessionID=3547&nav=true&Role=U%27

To translate a DU-AL ALET, use the ARCHECK command.

Application: an AR mode program check



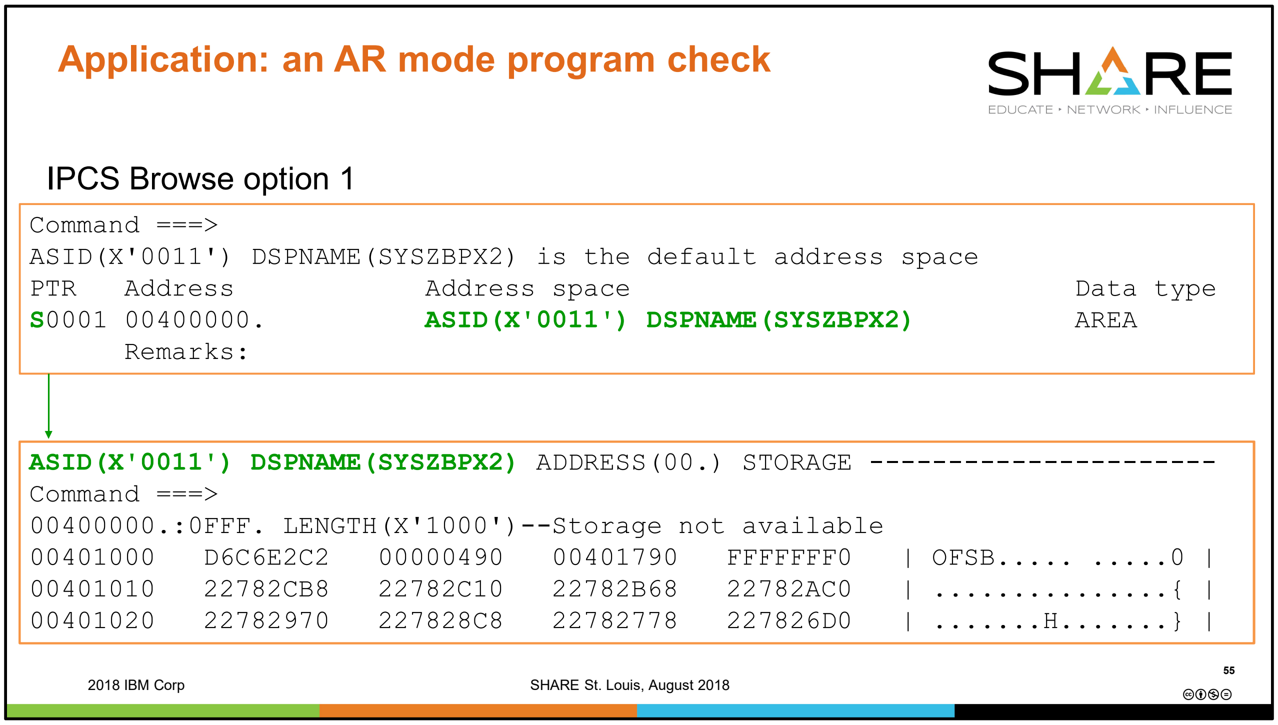
```
--------------------- IPCS - IAXAR2D EXEC -------------------------------
SELECT OPTION ===>

 Enter the following inputs.
 ALET Value :  0101006D    - ALET value should start in 01.
 ASID Value :  0011

 ALET value should be 8-digit hexadecimal (eg. 01xxxxxx).
 ASID Value should be 4-digit hexadecimal.


 IAXAR2D exec will show the output of the ARCHECK command by taking two
 input values, the ASID and ALET value, and provide the corresponding
 Dataspace name associated with that ALET value.
```



Application: an AR mode program check

```
IPCS OUTPUT STREAM -------------------------------
Command ===>
 ***********************************************
 VERSION 02/10/2006
 CVT: 00FD9F58
 ASVT: 00FAB988
 ASCB: 00F9F200
 ASSB: 06B8F000
 PALV: 7E3D7600

 ALET TRANSLATION
 ALET  addresses ASID(X'0011') DSPNAME(SYSZBPX2)
 ***********************************************
```

The output of the ALET2DSP exec shows that ALET 0101006D mapped against the PASN AL for ASID X’11’ translates to DSPNAME SYSZBPX2, owned by ASID X’11’.



Application: an AR mode program check

IPCS Browse option 1

```
Command ===>
ASID(X'0011') DSPNAME(SYSZBPX2) is the default address space
PTR   Address             Address space                        Data type
S0001 00400000.           ASID(X'0011') DSPNAME(SYSZBPX2)      AREA
      Remarks:
```

```
ASID(X'0011') DSPNAME(SYSZBPX2) ADDRESS(00.) STORAGE ----------------------
Command ===>
00400000.:0FFF. LENGTH(X'1000')--Storage not available
00401000   D6C6E2C2   00000490   00401790   FFFFFFF0   | OFSB..... .....0 |
00401010   22782CB8   22782C10   22782B68   22782AC0   | ...............{ |
00401020   22782970   227828C8   22782778   227826D0   | .......H.......} |
```

2018 IBM Corp SHARE St. Louis, August 2018 55

To browse storage in a data space, specify the data space name and the owning ASID. You can scroll through a data space and use indirection symbols to navigate a data space, just as you can with an address space.

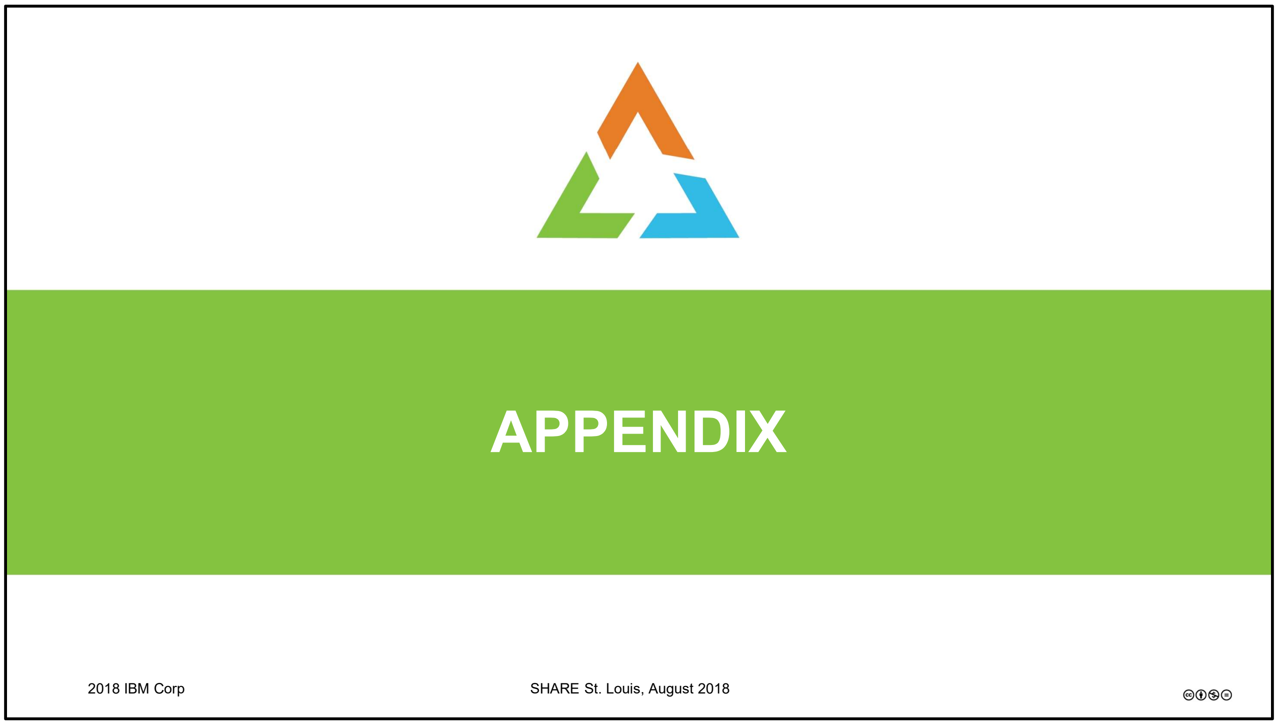
You can also LIST data space storage in a similar fashion, for example:

IP LIST 00400000 ASID(X’11’) DSPNAME(SYSZBPX2) LEN(X’10000’)

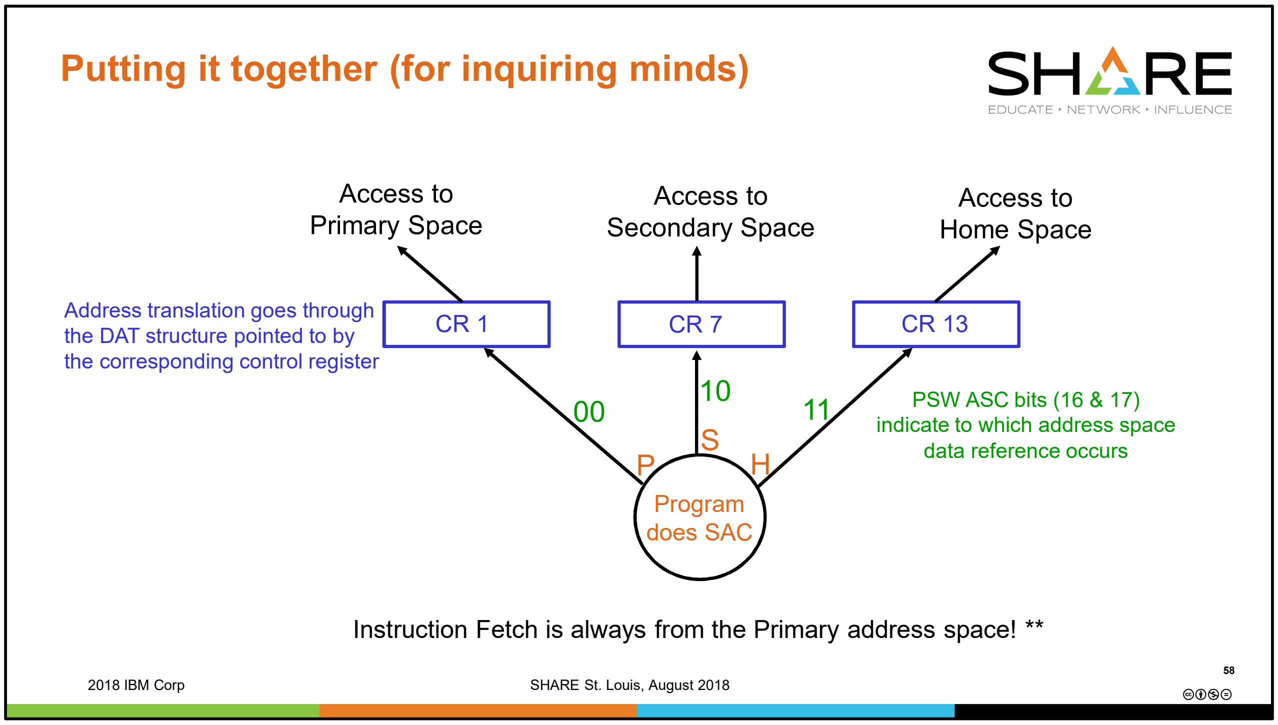
Application: an AR mode program check

```
IEA11054I Access Registers from STCB

  ACCESS REGISTER VALUES
      0-3   00000000  00000000  00000000  00000000
      4-7   00000002  00000000  00000000  00000000
      8-11  00000000  0101006D  00000000  00000000
     12-15  00000000  00000000  0101006D  00000002

ALET TRANSLATION
AR 04 addresses ASID(X'00B3')
AR 09 addresses ASID(X'0011') DSPNAME(SYSZBPX2)
AR 15 addresses ASID(X'00B3')
```

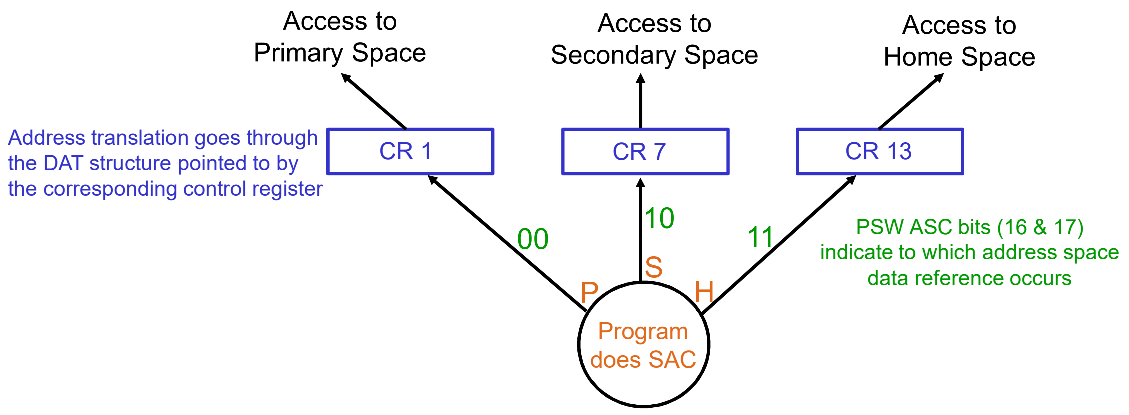
IP SUMM FORMAT report translates some ALETs



APPENDIX



Putting it together (for inquiring minds)



Access to Primary Space
Access to Secondary Space
Access to Home Space
Address translation goes through the DAT structure pointed to by the corresponding control register
CR 1
CR 7
CR 13
00
10
11
P
S
H
Program does SAC
PSW ASC bits (16 & 17) indicate to which address space data reference occurs

Instruction Fetch is always from the Primary address space! **

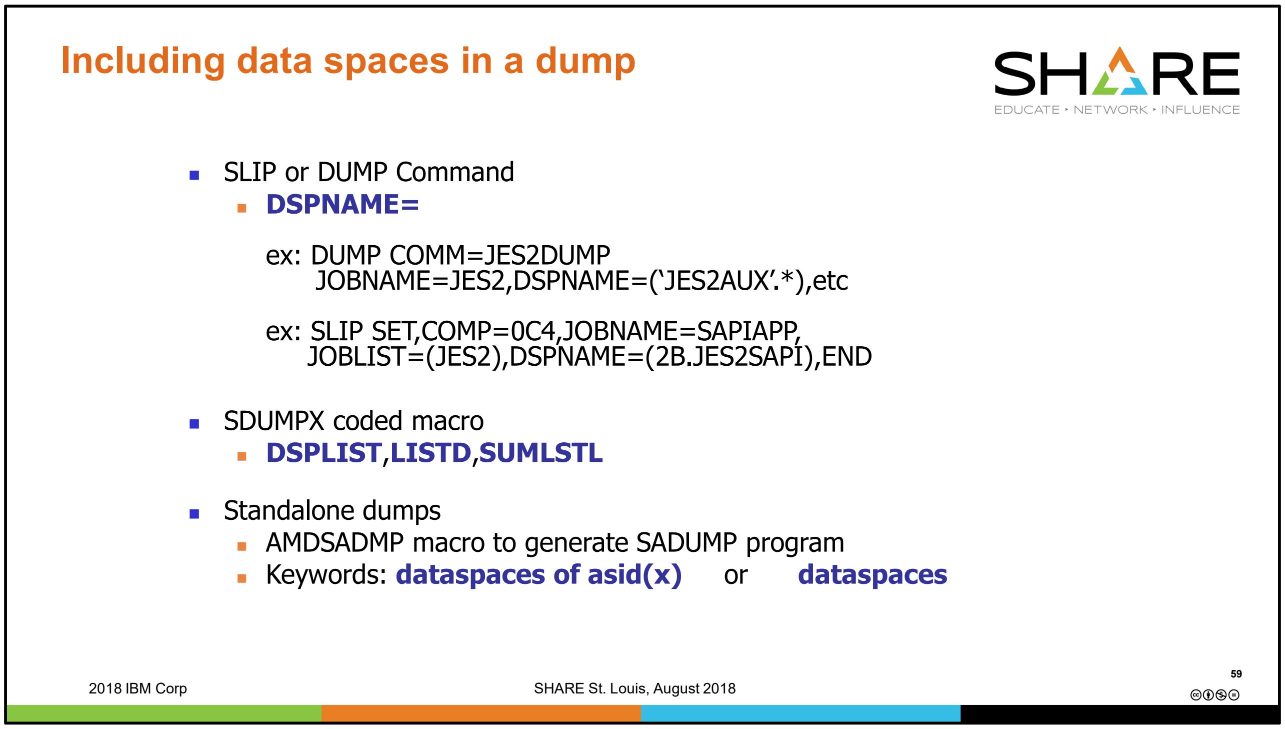
2018 IBM Corp

SHARE St. Louis, August 2018

58

For those who really like to know the in’s and out’s of what is going on behind the scenes, here is the bigger picture. The operating system has 3 control registers, CR1, CR7, and CR13, which are set up to point to the DAT structures (the dynamic address translation structure -- virtual-to-real address translation tables managed by RSM and read by hardware) for the primary, secondary, and home address spaces respectively. When an instruction which changes the primary or secondary address space is executed (for example, PC, PR, or SSAR), under the covers the control registers are updated accordingly. When an instruction which accesses storage, such as a LOAD or a STORE or a MVC is executed, the PSW ASC mode bit setting tells hardware which control register it should use for translation, i.e. which address space’s translation tables should be used for that instruction’s storage access.

To summarize, CR1 and CR7 and CR13 define WHAT are the primary, secondary, and home address spaces respectively, while the PSW ASC bit setting indicates WHICH of these 3 address spaces will be used for the executed instruction’s address translation.



Including data spaces in a dump

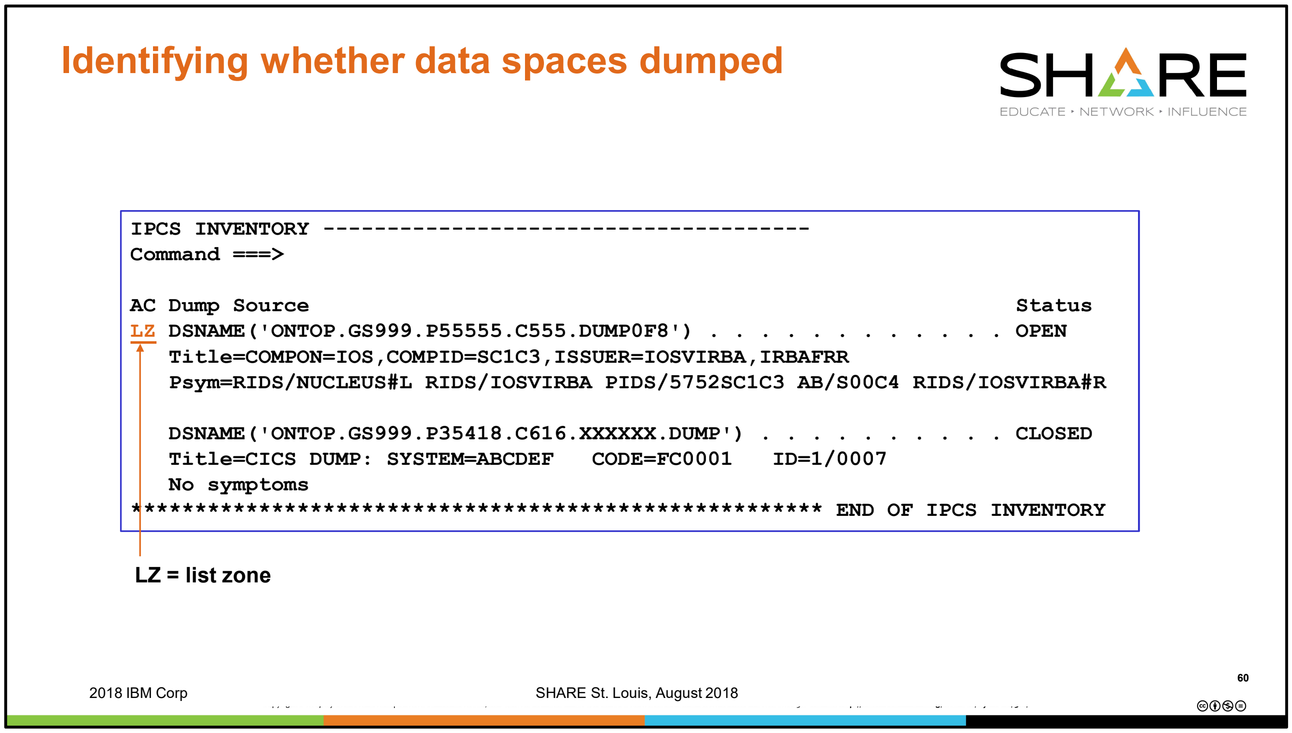
- SLIP or DUMP Command
  - **DSPNAME=**

    ex: DUMP COMM=JES2DUMP
    JOBNAME=JES2,DSPNAME=('JES2AUX'.*),etc

    ex: SLIP SET,COMP=0C4,JOBNAME=SAPIAPP,
    JOBLIST=(JES2),DSPNAME=(2B.JES2SAPI),END

- SDUMPX coded macro
  - **DSPLIST**,**LISTD**,**SUMLSTL**
- Standalone dumps
  - AMDSADMP macro to generate SADUMP program
  - Keywords: **dataspaces of asid(x)** or **dataspaces**

Various keywords designate data space names to include data space storage in dumps. Refer to System Commands, Authorized Assembler Services Reference, and MVS Diagnosis: Tools and Service Aids for more details.



Identifying whether data spaces dumped

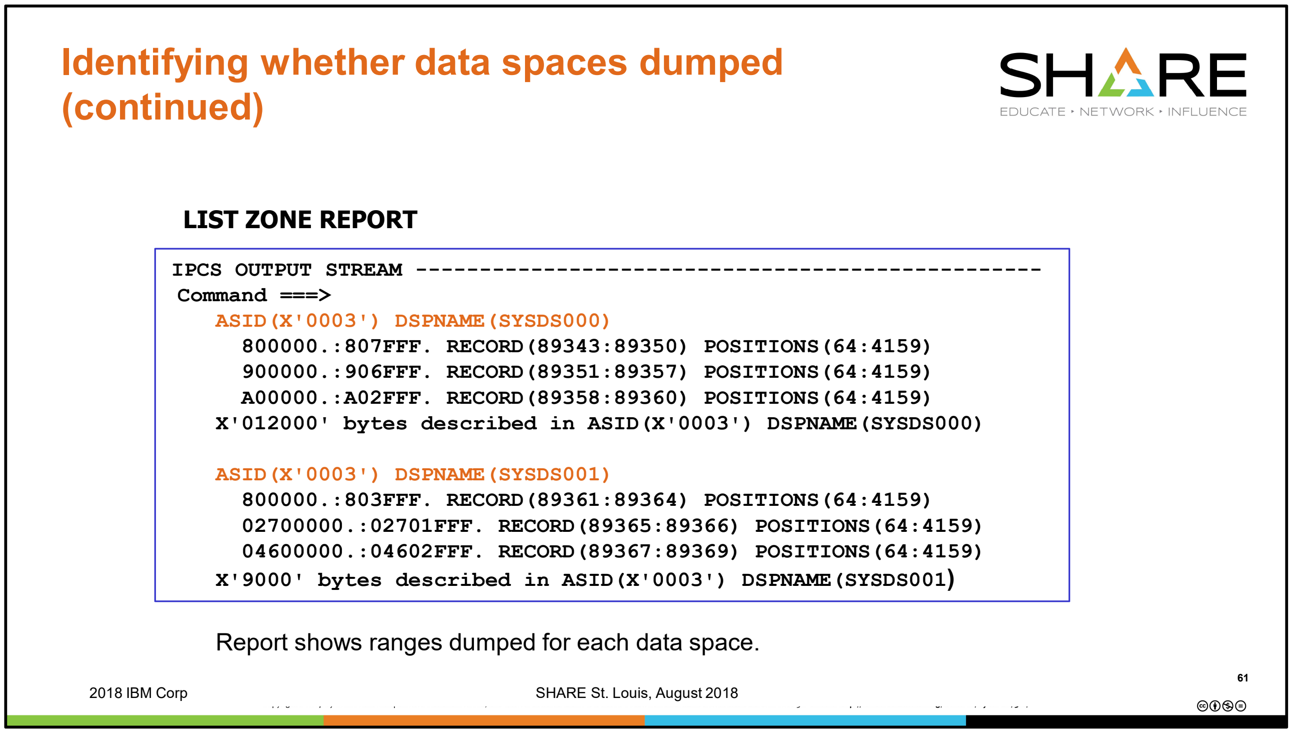
```
IPCS INVENTORY --------------------------------------
Command ===>

AC Dump Source                                                    Status
LZ DSNAME('ONTOP.GS999.P55555.C555.DUMP0F8') . . . . . . . . . . . . OPEN
   Title=COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVIRBA,IRBAFRR
   Psym=RIDS/NUCLEUS#L RIDS/IOSVIRBA PIDS/5752SC1C3 AB/S00C4 RIDS/IOSVIRBA#R

   DSNAME('ONTOP.GS999.P35418.C616.XXXXXX.DUMP') . . . . . . . . . . CLOSED
   Title=CICS DUMP: SYSTEM=ABCDEF   CODE=FC0001   ID=1/0007
   No symptoms
***************************************************** END OF IPCS INVENTORY
```

LZ = list zone

2018 IBM Corp SHARE St. Louis, August 2018 60

LZ (list zone) is a line command on the IPCS inventory panel that will display a comprehensive report of what the dump contains, including a dumped storage summary showing data spaces and storage ranges dumped.

LD (list dumped) is a subset of this report showing the dumped storage summary, including data spaces.



Identifying whether data spaces dumped (continued)

LIST ZONE REPORT

```
IPCS OUTPUT STREAM --------------------------------------------------
 Command ===>
   ASID(X'0003') DSPNAME(SYSDS000)
     800000.:807FFF. RECORD(89343:89350) POSITIONS(64:4159)
     900000.:906FFF. RECORD(89351:89357) POSITIONS(64:4159)
     A00000.:A02FFF. RECORD(89358:89360) POSITIONS(64:4159)
   X'012000' bytes described in ASID(X'0003') DSPNAME(SYSDS000)

   ASID(X'0003') DSPNAME(SYSDS001)
     800000.:803FFF. RECORD(89361:89364) POSITIONS(64:4159)
     02700000.:02701FFF. RECORD(89365:89366) POSITIONS(64:4159)
     04600000.:04602FFF. RECORD(89367:89369) POSITIONS(64:4159)
   X'9000' bytes described in ASID(X'0003') DSPNAME(SYSDS001)
```

Report shows ranges dumped for each data space.

2018 IBM Corp SHARE St. Louis, August 2018 61

Use ‘FIND DSPNAME’ and repeat FIND to step through all the data spaces dumped, or use ‘FIND JES2SAPI’ (or other dspname) to search for a specific data space in the body of the report.

Identifying data spaces owned by ASID

SHARE
EDUCATE • NETWORK • INFLUENCE

- **IPCS RSMDATA DSPACE ALL**   **displays all data spaces in the system**
- **IPCS RSMDATA DSPACE ASID(X'yy')**   **displays data spaces owned by ASID yy**

```
 JOBNAME   ASID DSP NAME OWNG TCB CUR B MAX B K T S R F TOT R
-------- ---- -------- -------- ----- ----- - - - - - -----
CONSOLE  000B IEAM02F2 007FD230 00100 00100 0 B A E N 00008
CONSOLE  000B IEEMCS01 007FD230 80000 80000 0 B S E Y 00024
```

See the RSMDATA chapter of
MVS Diagnosis: Reference
for an explanation of this report.

THANK YOU!!!