Assembler University 302

zNextGen User Experience: I'm Losing My Mind Trying to Figure Out Cross-Memory Routines!

SHARE in Anaheim
March 1, 2011
Session 8541

Author

This presentation was prepared by:

Kristine M. Harper
IMS R&D

NEON Enterprise Software
14100 Southwest Freeway
Suite 400
Sugar Land, TX 77478
Tel: 281-207-4978
E-mail: kristine.harper@neon.com

Agenda

- Terminology
- Why Cross-Memory?
- Cross Memory Environments
- Important Macros & Instructions
- Stacking PCs
- Address Space Swapping
- Overview of Locking
- Final Tips and Recommendations

Agenda

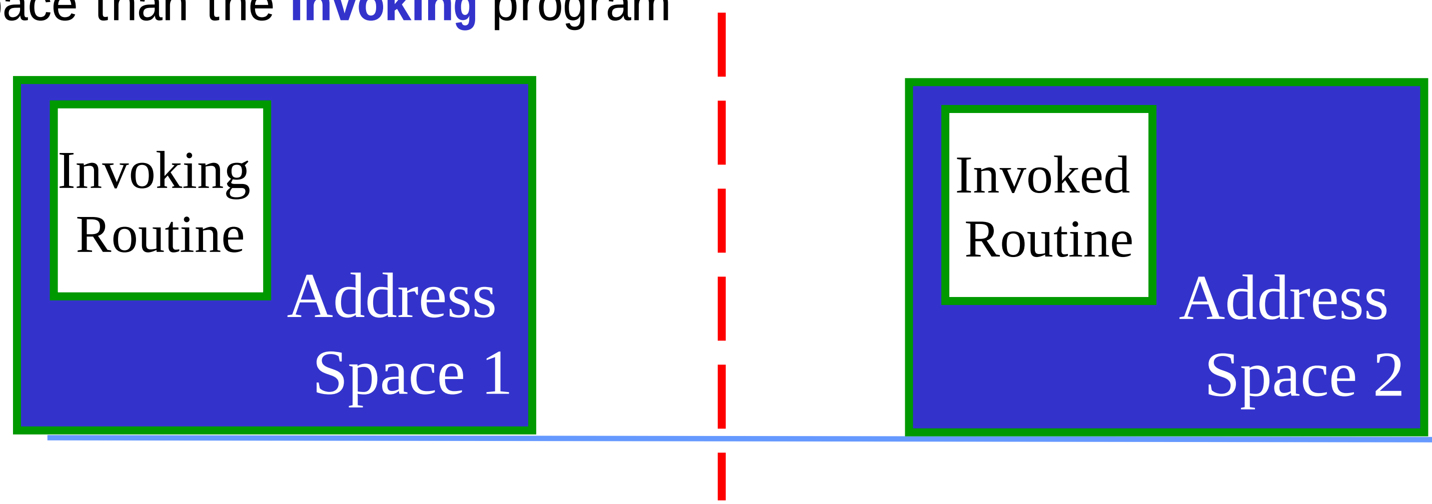
- Other important XMS topics we will not get to today: storage control, MVCP/MVCS, linkage conventions, system parameters, z/OS macros available/unavailable while in XM mode, recovery, data spaces, hiperspaces and many more XMS topics!
  - Maybe a two-part session in Orlando! ☺
- In this session, we'll be specifically covering **Synchronous XMS**
  - The other XMS method involves SCHEDULEing an SRB to run in another address space and that is called Asynchronous XMS

Terminology

Terminology

What is Cross Memory?

- Cross Memory Services, or XMS
  - Technique that MVS and z/OS applications use to invoke programs in other address spaces
  - Provides a synchronous method of communications between address spaces
  - In other words, the **invoked** routine resides in a different address space than the **invoking** program



Terminology

What is Cross Memory?

- From the MVS Programming Extended Addressability Guide:
  - Synchronous cross memory communication enables one program to provide services synchronously to other programs
  - Takes place between a user and a service provider when the user issues a PC (program call) instruction
- Cross Memory Environment occurs between a Primary Address Space and a Service Address Space
  - Requires authorization, linkage and entry tables

Terminology

What is Cross Memory?

- The caller runs under the *same* unit of work (UOW, TCB or SRB) in the same or a *different* address space
- For synchronous, at any given moment, the UOW is either running in one address space or another, but never in two different ones at the same time
- The UOW is just an operating system concept – the hardware is unaware of such a thing
- So this is why the set of services you can request in XM mode is limited and/or require special interfaces: because the UOW addressability may be in multiple address spaces!

Terminology

Home Address Space

- Address space to which a unit of work is associated with and whose address is pointed to by the PSA field PSAAOLD when the unit of work is executing*
- Address space in which the TCB or SRB are initially dispatched
- Remains the same during the life of work unit
- ASID of the Home Address Space = **HASID**
- Also known as **HASN** (Home Address Space Number)
- In Home Address Space mode, instructions and data are fetched from home

*Advanced Assembler Language and MVS Interfaces

SHARE
Technology · Connections · Results

Terminology

Primary Address Space

- Address space whose segment table (pointed to by CR1) is used to access instructions and data when the CPU is in primary mode
- Can be changed
- ASID of the Primary Address Space = **PASID**
- Also known as **PASN** (Primary Address Space Number)
- In Primary Address Space mode, instructions and data are fetched from primary

Secondary Address Space

- Address space whose segment table (pointed to by CR7) is used to access data when the CPU is in secondary mode
- Can be changed
- ASID of the Secondary Address Space = **SASID**
- Also known as **SASN** (Secondary Address Space Number)
- In Secondary Address Space mode, instructions are fetched from primary and data is fetched from secondary

SHARE Technology · Connections · Results

Terminology

ASC (Address Space Control) mode

- Determined by PSW bits 16-17 and tells the system where to find the referenced data (the data referenced by the address in the GPRs)
- When ASC mode is **secondary** (and hence Cross Memory Mode), the SAC is set to 256, and the data resides in the secondary address space

AR (Access Register) Mode

- Data referenced by a program resides in the address or data space pointed to by the ARs (instructions are fetched from primary)
- PSW bit 17 is 1

SHARE
Technology · Connections · Results

Terminology

A note about the PSW for Home, Primary and Secondary address spaces:

Mode	PSW Bits 16 and 17
Home Space	11
Primary Space	00
Secondary Space	10
AR Mode	01

Terminology

Space Switch Routine

- Code that is the target of the PC instruction that executes in another address space

Service Provider

- The program (ie, an address space) that provides services synchronously to other program (ie, other address spaces or users)

Terminology

PC Routine

- When the user program issues the PC instruction, the PC instruction transfers control to the PC routine
- PC routine is a service provider program that provides the requested service (or invokes other programs to provide the service), and then returns control to the user
- The PC routine executes under the same UOW as the user, which is where the synchronous part comes from
- PC Number
  - Identifies a specific PC routine
  - Created by the service provider and supplied to the user that issues the PC instruction
- PC routines can access data in the user's address space using ARs or by using the MVCP/MVCS instructions
  - For this presentation, we'll focus on using ARs

Terminology

A note about H/P/SASID and H/P/SASN

- The xASN is more of a hardware term (ie, Principles of Operation)
- The xASID is more of a software term
- But they are interchangeable and refer to the same thing. xASN will be used in this presentation

HASN = HASID

PASN = PASID

SASN = SASID

Terminology

When the Job Step is initially dispatched:

HASN = PASN = SASN

And so another way to think about XMS, Cross Memory mode exists when one or more of these conditions is true:

- Current PASN NE current HASN
- Current SASN NE current HASN
- ASC mode is secondary

Why Cross Memory?

Why Cross Memory?

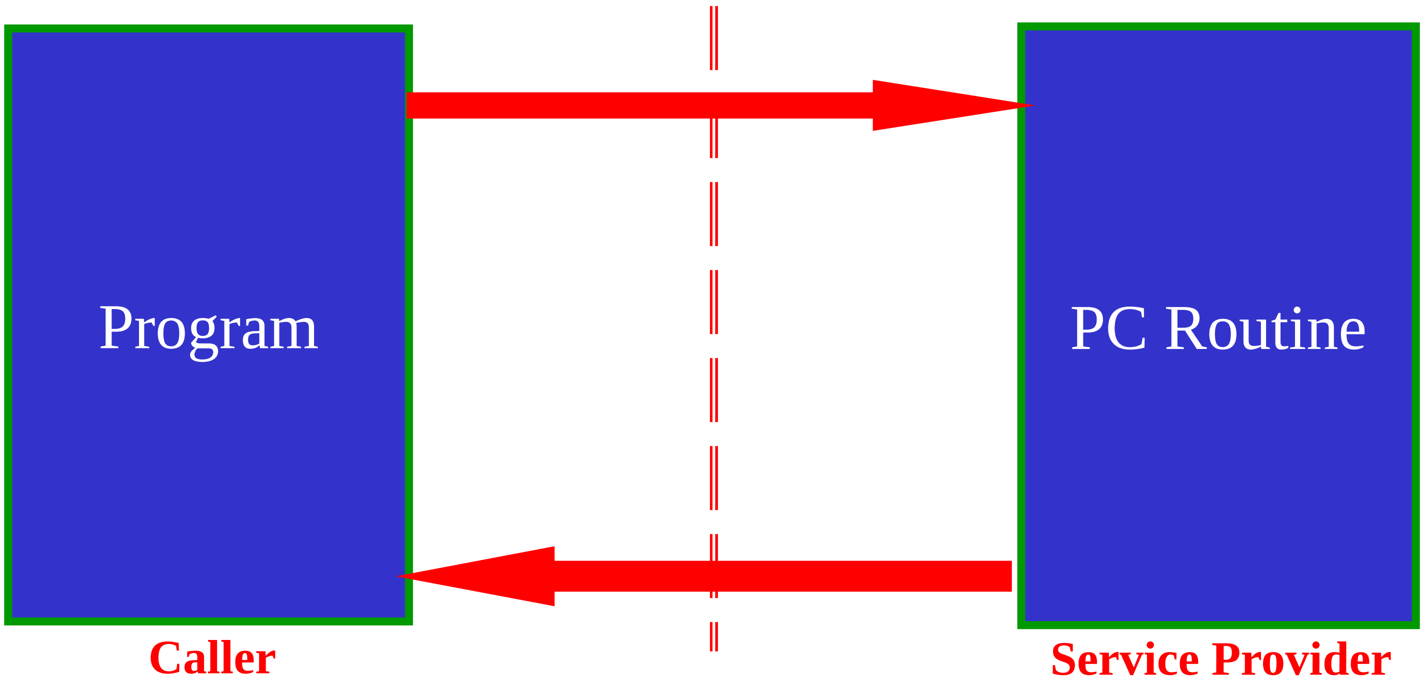
- Can provide virtual storage constraint relief
  - Data can be copied, moved or referenced by the server program directly from the client address space without having a buffer in common
- Can improve the integrity of the service and its data
  - Code is isolated from the calling program (service and data separate from the user)
- Most efficient way to transfer data between address spaces
- Provide authorized services to problem state programs
  - Unauthorized callers can have controlled access to authorized services
  - Would otherwise have to fully authorize the caller

Why Cross Memory?

- Common space can be preserved because large structures can be referenced in the Service Provider's address space
- Code path is much shorter than the SVC method for cross memory
- Compared to the limit of 256 SVC's, the number of possible PCs is very large

Cross Memory in Motion

Here is a basic example:



The program in the user's address space issues a PC instruction to pass control to the PC routine in the Service Provider's address space

After the PC routine executes, the stacking PC issues a PR instruction to return control back to the user's program

Cross Memory Environments

Cross Memory Environments

The Cross Memory Environment includes tables and linkages that connect the service provider's address space to:

- The user's address space
- The tables and linkages that provide the necessary authorization for the service provider

Multi-level authorization facility

Cross Memory Environments

Three main areas to consider for the XM environment:

- Cross Memory Authorization
  - Defines program and address space authorization
- Cross Memory Linkage
  - Defines data structures and tables
- Linkage Conventions
  - Defines programming conventions

Cross Memory Environments

Cross Memory Authorization

- Program authorization
  - The PKM (PSW key mask) is a 16-bit value (bits 0-15 of CR3) that is used to authorize problem state programs to use XMS
    - Represents PSW storage protections keys that are valid for programs to use
    - Used in an authority check (along with the AKM) to determine if a specific PC number is authorized
    - Can be changed by the PC and PT instructions
    - Supervisor state programs do not need PKM authorization

SHARE
Technology · Connections · Results

Cross Memory Environments

Cross Memory Authorization

- Address space authorization
  - System Authorization Table (**SAT**)
    - Entries define the PT and SSAR authority that another address space has, with respect to the address space that owns the SAT
    - Entries are indexed by **authorization indexes (AX)**
    - AX entry indicates if an address space is authorized to access other address spaces
    - If AX = 1, the entry has both PT and SSAR authority
    - If AX = 0, the entry has neither PT nor SSAR authority

SHARE
Technology · Connections · Results

Cross Memory Environments

Cross Memory Linkage

- Each address has a system linkage table and a linkage table associated with it
  - System linkage table: defines the XMS available to **all** address spaces
    - System LX's
  - Linkage table: defines the XMS available to a **specific** address space
    - Non-system LX's
- Linkage table entries are referenced by a linkage index (LX)
  - LX values are unique across the system and can be reserved thru the LXRES macro

Cross Memory Environments

Cross Memory Linkage

- Each LX points to an entry table
  - Each **entry table** describes one or more services (i.e., programs call by the PC instruction) offered by the service provider's address space
  - And each program description is reference by an **entry index (EX)**
  - The EX is used to locate an entry (**ETE**) in the entry table
  - **The program described by the ETE is the one that will receive control as a result of the PC instruction**

Cross Memory Environments

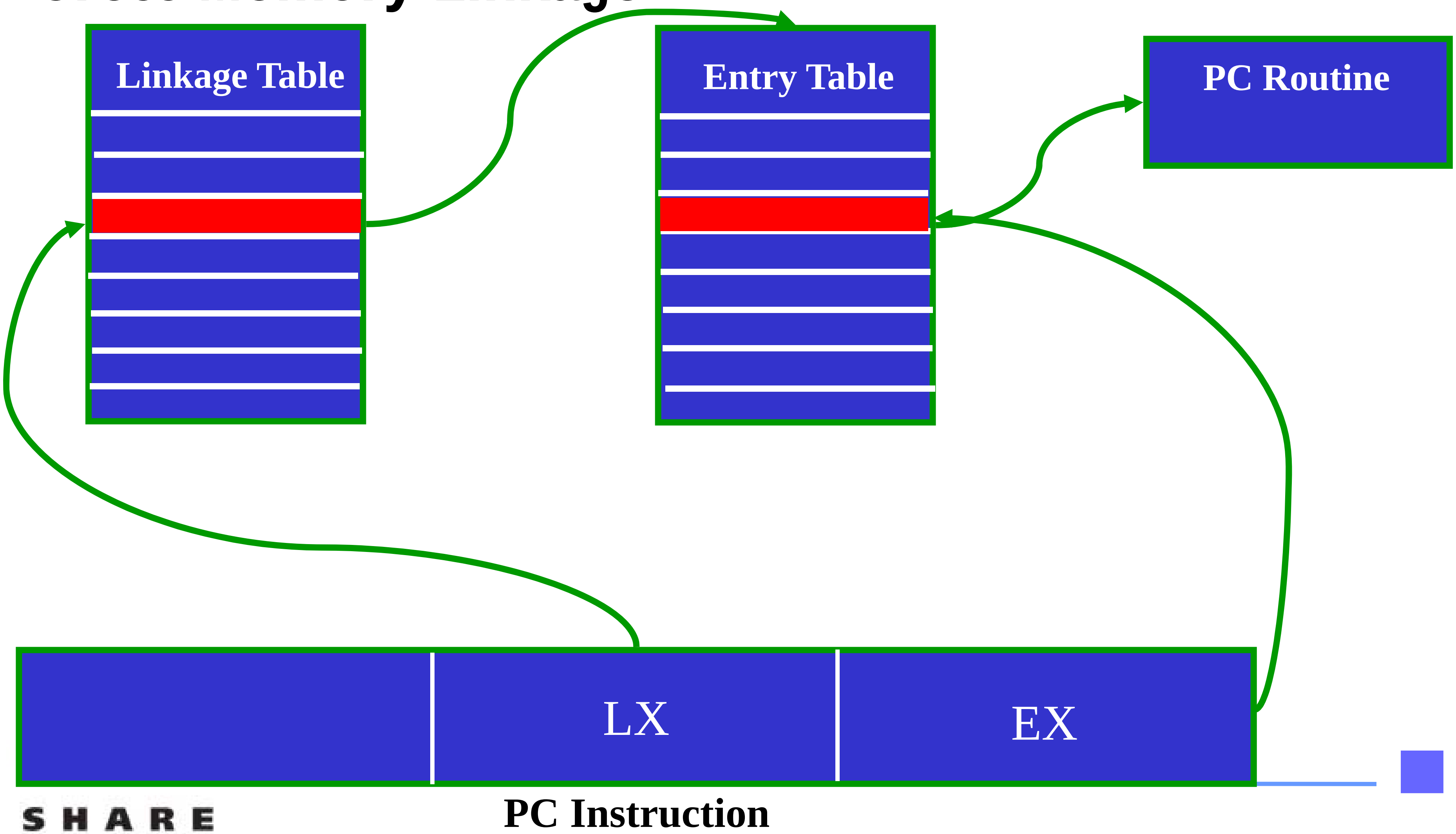
Cross Memory Linkage

- PC instruction format: PC $D_2(B_2)$
- The PC number comes from bits 12-31 of the address of the specified operand:

- Bits 12-23 specify the LX value
- Bits 24-31 specify the EX value

Cross Memory in Motion

Cross Memory Linkage



Important Macros and Instructions

Important Macros and Instructions

Macros used for XMS

- ATSET: Sets the authority of the PT and SSAR instructions in the HASN's authority table entry
- AXEXT: Sets the AX value for an address space
- AXFRE: Frees up an AX value for reuse
- AXRES: Reserves an AX in the authorization table
- AXSET: Sets the AX value for the home address space
- ETCON: Connects an entry table to a linkage table at the specified LX (linkage index) in the home address space

Important Macros and Instructions

Macros used for XMS

- ETCRE: Creates a PC entry table from PC routine definitions
- ETDEF: Defines a program in the entry table – the PC routine definitions that are used by ETCRE as input
- ETDES: Destroys an entry table
- ETDIS: Disconnects an entry table from a linkage table
- LXFRE: Frees up an LX value for reuse
- LXRES: Reserves an LX in the linkage table

SHARE
Technology · Connections · Results

Important Macros and Instructions

Example*, using AXSET to obtain PT and SSAR authority to all address spaces:

```
Set my PT/SSAR authority index

AXSET AX==Y(1)
```

*Note that all the examples included here are in reference to space switching PC routines

Important Macros and Instructions

Example, using LXRES to reserve an LX:

```
------------------------------------------------------------
Allocate a System Linkage Index
------------------------------------------------------------
LXRES LXLIST=KMH123_LXWA,        Obtain an LX
      SYSTEM=YES,                make it a System LX
      MF=(E,KMH123_LXRESWA)

IF (LTR,R15,R15,NZ)
  $ERR LXRES_FAILED
  J    KMH123_9000
ENDIF ,
```

Important Macros and Instructions

Where KMH123_LXWA and KMH123_LXRESWA:

```
**          LXRES Work Area
*
            DS     0D
KMH123_LXRESWA LXRES LXLIST=0,
               SYSTEM=YES,
               MF=L
KMH123_LXRESLN EQU *-KMH123_LXRESWA
```

```
***         SS PC Routine Area
***
            DS     0D
KMH123_LXWA    DS     0F                  1/3 LXLIST Area
KMH123_LXCNT   DS     F                   2/3 Number of LX's requested
KMH123_LXVAL   DS     F                   3/3 LX Number
            DS     0D
KMH123_ETWA    DS     0F                  1/3 ETCRE Work area
KMH123_ETCNT   DS     F                   2/3 Number of ET's created
KMH123_ETTOK   DS     F                   3/3 ET Token from ETCRE
```

Important Macros and Instructions

Example, using LXFRE to release an LX:

```
IF     (ICM,R0,15,CSVTAUTH,NZ)     Do we have a PC Number?
  MVC    AUTHDL_LXCNT,=F'1'        Set the number of LX's acquire
  ST     R0,AUTHDL_LXVAL           Save the PC Number
  LXFRE LXLIST=AUTHDL_LXWA,        Since this is a non-system LX,+
       FORCE=YES,                  the only address space that   +
       MF=(E,AUTHDL_LXFREWA)       should be connected is us.
```

Important Macros and Instructions

Where AUTHDL_LXWA and AUTHDL_LXFREWA:

```
*-------------------------------------
*        LXFRE WORK AREA
*-------------------------------------
         DS    0D
AUTHDL_LXFREWA LXFRE LXLIST=0,
               FORCE=YES,
               MF=L
AUTHDL_LXFRELN EQU *-AUTHDL_LXFREWA
```

```
AUTHDL_LXWA    DS 0F        1/3 LXLIST Area
AUTHDL_LXCNT   DS F         2/3 Number of LX's requested
AUTHDL_LXVAL   DS F         3/3 LX Number
```

SHARE
Technology · Connections · Results

Important Macros and Instructions

Example, using ETDEF to build an Entry Table Entry:

```
L      R5,CSVTXMCMD      Get PC Routine Address
L      R6,CSVTXMARR      Get ARR Address

LA     R8,ETDEF_AREA
LA     R7,ETRTN0-ETDEFWA(,R8)

ETDEF  TYPE=SET,         Set my value
       ETEADR=(R7),
       ROUTINE=(R5),
       PARM1=(CSVTREG),
       ARR=(R6),
       SSWITCH=YES,
       PC=STACKING,
       ASCMODE=PRIMARY,
       RAMODE=31,
       STATE=SUPERVISOR,
       EK=0,
       PKM=REPLACE,
       EKM=(0:15),
       AKM=(0:15)
```

This PC routine is defined as a space switch routine

Important Macros and Instructions

Where ETDEF_AREA:

```
ETDEF_AREA        DS      CL(ETDEFLN)         ETDEF work area
```

- ETDEFLN is the total size of all the entry table entries
- ETEADR points to R7 in the example, and R7 points to ETRTN0, which is the routine that will be invoked

Important Macros and Instructions

Example, using ETCRE to create the Entry Table:

```
ETCRE ENTRIES=ETDEF_AREA          Create my Entry Table
IF    (LTR,R15,R15,NZ)            Any errors?
  J     BLDETE_EXIT                 Yes, branch
ENDIF ,

ST    R0,ETTOK                    Save the Token value
MVC   ETCNT,=F'1'                 Set number of Entry tables
```

Important Macros and Instructions

Where ETDEF_CNT:

```
                DS      0D
ETWA            DS      0F        1/3 ETCRE Work area
ETCNT           DS      F         2/3 Number of ET's created
ETTOK           DS      F         3/3 ET Token from ETCRE
```

SHARE
Technology · Connections · Results

Important Macros and Instructions

Example, using ETCON to connect the Entry Table to the LX:

```
---------------------------------------------------
 Connect the Entry Table to the Linkage Index
---------------------------------------------------
 ETCON TKLIST=ETWA,
       LXLIST=LXWA,
       MF=(E,ETCONWA)
```

SHARE
Technology · Connections · Results

Important Macros and Instructions

Where ETCONWA:

```
              DS        0D
ETCONWA       ETCON     TKLIST=0,
                        LXLIST=0,
                        MF=L
ETCONLN       EQU       *-ETCONWA
```

Important Macros and Instructions

Instructions used for XMS

- PC: Program Call
  - Causes a program (the PC routine) in another address space to receive control
- SSAR: Set Secondary ASN
  - Used to set an address space to the Secondary Address Space
- EPAR: Extract Primary ASN
  - Places the PASID into a GPR
- ESAR: Extract Secondary ASN
  - Places the SASID into a GPR

Important Macros and Instructions

Instructions used for XMS

- MVCK: Move with Key
  - Moves data between storage areas that have different storage protection keys
- MVCP: Move to Primary*
  - Moves data from the SASN to the PASN
- MVCS: Move to Secondary*
  - Moves data from the PASN to the SASN
- IAC: Insert Address Space Control
  - Indicates current ASC mode
- SAC/SACF: Set Address Space Control/FAST
  - Sets bits 16-17 for the ASC mode

*Alternative way for PC routines to access from other address spaces or data spaces vs using AR's

Important Macros and Instructions

Example, using IAC and SACF to set the ASC mode to primary:

```
LA      R2,0                              I like it clean
IAC     R2                                Current ASC mode
ST      R2,WA#IAC                         Save it
SACF    SACP                              Into primary mode
```

Then you can later go back to the original mode:

```
L       R2,WA#IAC                         Original ASC mode
SACF    0(R2)                             Back to original mode
```

```
WA#IAC                  DS      F                   ASC mode
```

SHARE
Technology · Connections · Results

Stacking PCs

Stacking PCs

PC Linkages

- Types of PC Linkage used to invoke PC Routines
  - Stacking PC
    - The user's environment is saved by the system on the linkage stack
    - When the PC routine is done, it issues the PR instruction to restore the user's environment and control is returned to the user
    - **Stacking PC Linkage** is highly recommended and is what we'll focus on for this presentation

Stacking PCs

Basic and Stacking PC Shared Functionality

- The PKM (PSW key mask) authority of the PC routine can be increased
- The PC routine can receive control in problem or supervisor state
- The PC routine can be a space switch routine or a non-space switch routine

Stacking PC Only Functionality

- The PKM authority of the PC routine can be decreased
- The PSW key of the PC routine can be set from data in the entry table
- The PC routine can receive control in AR mode
- Linkage stack is automatically used to save and restore user's environment

Address Space Swapping

Address Space Swapping

What is swapping?

- Used by **SRM** (System Resource Manager) to control which address spaces should have access to system resources
- Swapping helps to balance the use of resources
- Can help with performance and throughput
- There are several kinds of domain-related swaps and system-related swaps

Address Space Swapping

Stacking PCs ←→ Space Switch ←→ Swapping

- If a stacking PC routine causes a space switch (when the primary address space changes), it must be running in a non-swappable address space
- To make an address space non-swappable (before you create the space switching PC), you need to use the SYSEVENT macro
- Then after you are done with your space switching PC, you can make it swappable again using the SYSEVENT macro

SHARE
Technology · Connections · Results

Address Space Swapping

SYSEVENT Macro

- SYSEVENT DONTSWAP
  - This will make your address space non-swappable
  - Notifies the SRM that the address space can't be swapped out
- SYSEVENT OKSWAP
  - This will make your address space swappable again
  - Notifies the SRM that the address space is eligible for swapping

Address Space Swapping

SYSEVENT Macro Examples:

```
******************************************************************
*          Set Swappable                                         *
******************************************************************
DRP7100  DS     0H
         ST     R14,W#DRP7100               Save return address
         $NSXKMH BEFORE,SYSEVENT
         SYSEVENT OKSWAP,ENTRY=BRANCH       Make swappable
         $NSXKMH AFTER
         IF     (LTR,R1,R15,NZ)             If error save RC
           $NSXKMER X'71',ACTION=SETRC      Error OKSWAP failded
           BRAS  R14,DRP9000                Issue error message
         ELSE   ,                           SYSEVENT successful
           NI    W#FLAG,255-W#FLAGNONSWP    Indicate swappable
         ENDIF  ,
         L      R14,W#DRP7100               Restore return address
         BR     R14                         Return
```

When you specify ENTRY=BRANCH,
R13 must contain the address of a 72-byte save area

Address Space Swapping

SYSEVENT Macro Examples:

```
*****************************************************************
*         Set Non Swappable                                     *
*****************************************************************
DRP7000  DS     0H
         ST     R14,W#DRP7000                  Save return address
         $NSXKMH BEFORE,SYSEVENT
         SYSEVENT DONTSWAP,ENTRY=BRANCH         Make non swappable
         $NSXKMH AFTER
         IF     (LTR,R1,R15,NZ)                If error save RC
           $NSXKMER X'70',ACTION=SETRC         Error DONTSWAP failed
           BRAS  R14,DRP9000                   Issue error message
         ELSE   ,                              SYSEVENT successful
           OI    W#FLAG,W#FLAGNONSWP           Indicate non swappable
         ENDIF  ,
         L      R14,W#DRP7000                  Restore return address
         BR     R14                            Return
```

Overview of Locking

Overview of Locking

Cross Memory Local (CML) Lock

- The local lock of an address space other than the home address space
- Allows XMS to serializes functions and storage allocation
- One CML lock per address space
- This is a suspend lock, versus a spin lock

CML Address Space

- The address space, other than the home address space, whose local lock is held as a CML lock
- After the CML lock is obtained, the CML address space doesn't have to remain the primary or secondary address space

Overview of Locking

- The CML lock is obtained by using the SETLOCK macro
- The issuing program must be in supervisor state and PSW key 0
- Owning the CML lock allows for address space level synchronization
- Owning the CML lock creates an active link between the CML address space and the address space that owns the lock (usually the home address space) – and so neither address space can swapped out

SHARE
Technology · Connections · Results

Overview of Locking

SETLOCK OBTAIN Example:

```
-----------------------------------------------------------------
Obtain the Cross-Memory Local Lock
-----------------------------------------------------------------
$KHASCB ,                         Get Primary ASCB address
LR      R6,R1                     Save ASCB address
LR      R11,R1                    Save ASCB address

SETLOCK OBTAIN,TYPE=CML,MODE=UNCOND,REGS=USE,ASCB=(11)
```

Overview of Locking

SETLOCK RELEASE Example:

```
-------------------------------------------------------------
Release the Cross-Memory Local Lock
-------------------------------------------------------------
LR      R11,R6                          Set ASCB address
SETLOCK RELEASE,TYPE=CML,REGS=USE,ASCB=(11)
```

Overview of Locking

SETLOCK vs. SYSEVENT

- SETLOCK obtains a lock for a very short period of time (microseconds) and is used for serialization (not for making an address space non-swappable)
- SYSEVENT can make an address space non-swappable for a long period of time (hours, days, weeks)

Final Tips and Recommendations

Final Tips and Recommendations

- XMS is **NOT** an easy topic to learn
- There are many different ways to establish and work with XMS
  - You should focus on the way that works best for your project
- Start with a sandbox-type of program to get the basics down
- If at first you don't succeed, try, try, try and try again!

Final Tips and Recommendations

- Remember that resource management is different with XMS
  - What happens if you invoke a program in another address and the program terminates?
- You must consider where your resources came from – your own resources vs. those that came from cross memory users
- The PC routine execution time is tied to the home address space, but that may not be where the routine actually executes

SHARE
Technology · Connections · Results

Final Tips and Recommendations

- Programs running in XM mode don't have access to MVS macros unless the documentation explicitly mentions it
- Programs running in XM mode cannot issue any SVCs except ABEND (i.e., if a macro is dependent on an SVC, you won't be able to use it in XM mode)

Resources

I am thankful to many people who helped me learn XMS! I'm still learning, but without their help, this presentation would have been impossible.

- Tom Harper
- Dave Kreiss
- Tony Lubrano
- Michael Stack

SHARE
Technology · Connections · Results

Resources

Other resources:

- Advanced Assembler Language and MVS Interfaces (yes, I am blessed to own a copy!)
- MVS Programming Extended Addressability Guide
  - http://publib.boulder.ibm.com/infocenter/zos/v1r11/index.jsp?topic=/com.ibm.zos.r11.ieaa500/cmc.htm
- Lend Me Your EAR: The ART of MVS/ESA Programming, SHARE session by Joel Sarch
  - http://www.cbttape.org/ftp/infolib/SHARE72-O324-O325-O326.pdf

The Next Generation of Mainframe Professionals.