Lend Me Your EAR: The ART of MVS/ESA™ Programming

Joel M. Sarch

amdahl Corporation

Installation Code AMD

1 March 1989

SHARE 72

Sessions O324, O325, O326

MVS SCP Project

Lend Me Your EAR: The ART of MVS/ESA™ Programming

All rights reserved

1

Disclaimer

The ideas, concepts, and information contained within this presentation are offered, in the tradition of SHARE, Inc., for the enlightenment and edification of the audience, and as a source of ideas and inspiration for further investigation and development.

Neither the speaker nor Amdahl Corporation is responsible for any errors or misrepresentations made herein.

Every attempt has been made to verify the accuracy of the information contained in this presentation. However, it is the responsibility of anyone using this information to verify its accuracy and currentness for him/herself.

2

Trade Marks

The following are trademarks of the International Business Machine Corporation:

- ESA/370™
- MVS/ESA™
- MVS/XA™
- MVS/SP™
- Hiperspace™
- IBM™
- International Business Machines Corporation™

UNIX™ is a trademark of American Telephone and Telegraph (AT&T)

3

Prologue

Let's begin by examining:

- Dual Address Space Facility
- Advanced Address Space Facility
- Cross-Memory Services Facility
- Multiple Address Space Facility
- Outline of the Presentation

4

Dual Address Space Facility (DAS)

- Set of 370-XA instructions
- *Direct* communication between address spaces
  - — Direct access to data in two address spaces simultaneously
  - — Data movement directly from one address space to another
  - — Direct execution of code resident in the private area of another A/S
- Switch to supervisor state or system PSW key within same address space without incurring an interrupt
- Authorization or debarment of access to another address space

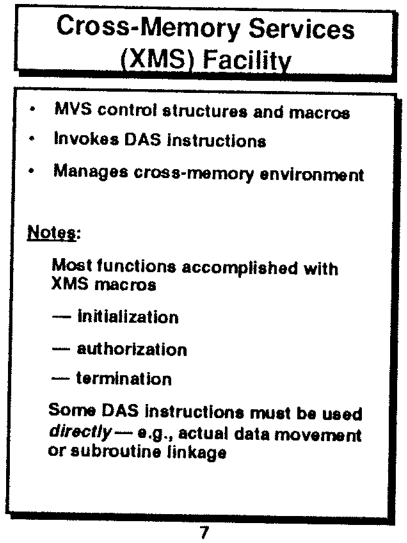
5

Advanced Address Space Facility (AASF)

- Extends DAS
- Direct communication with up to 16 distinct address spaces simultaneously
- Data-only address spaces
- Enhanced mechanism for executing external code:
  - — stack for saving status, registers
  - — tighter security
- Generally enhanced authorization mechanism with more options
- A few new instructions

6

Lend Me Your EAR: The ART of MVS/ESA™ Programming



Cross-Memory Services (XMS) Facility

- MVS control structures and macros
- Invokes DAS instructions
- Manages cross-memory environment

<u>Notes</u>:

Most functions accomplished with XMS macros

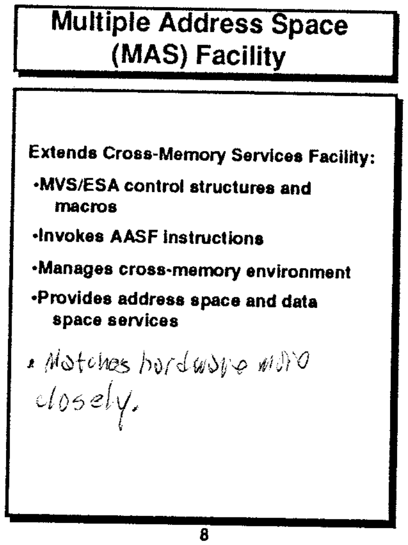
— initialization

— authorization

— termination

Some DAS instructions must be used *directly* — e.g., actual data movement or subroutine linkage

7



Multiple Address Space (MAS) Facility

Extends Cross-Memory Services Facility:

•MVS/ESA control structures and macros

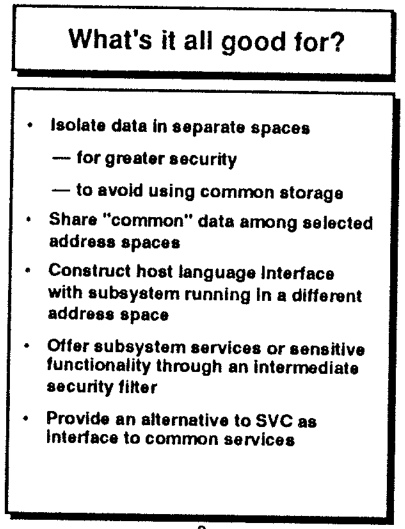
•Invokes AASF instructions

•Manages cross-memory environment

•Provides address space and data space services

• Matches hardware more closely.

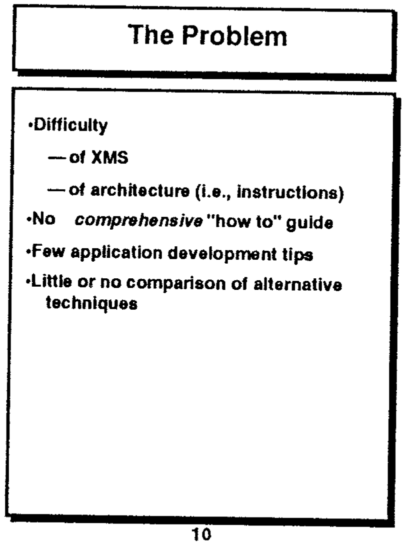
8



What's it all good for?

- Isolate data in separate spaces
  - — for greater security
  - — to avoid using common storage
- Share "common" data among selected address spaces
- Construct host language interface with subsystem running in a different address space
- Offer subsystem services or sensitive functionality through an intermediate security filter
- Provide an alternative to SVC as interface to common services

9



The Problem

•Difficulty

—of XMS

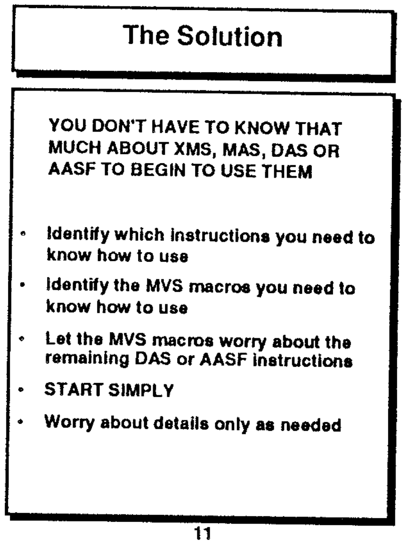
—of architecture (i.e., instructions)

•No *comprehensive* "how to" guide

•Few application development tips

•Little or no comparison of alternative techniques

10

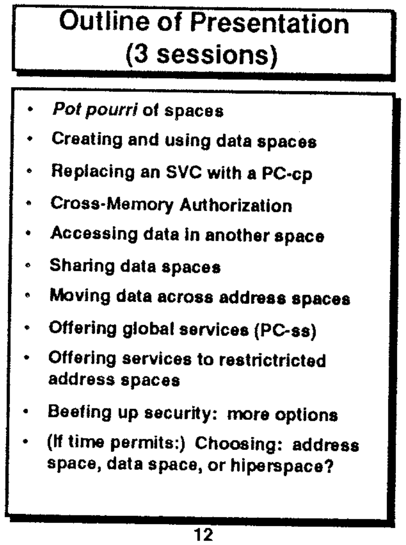


The Solution

YOU DON'T HAVE TO KNOW THAT MUCH ABOUT XMS, MAS, DAS OR AASF TO BEGIN TO USE THEM

- Identify which instructions you need to know how to use
- Identify the MVS macros you need to know how to use
- Let the MVS macros worry about the remaining DAS or AASF instructions
- START SIMPLY
- Worry about details only as needed

11



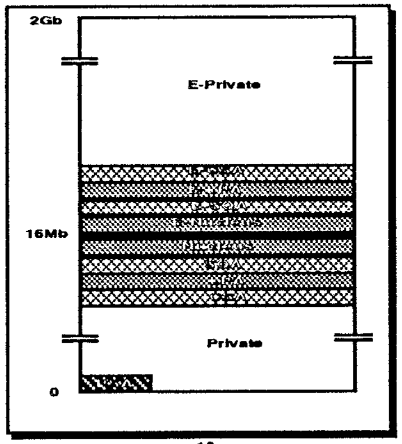
Outline of Presentation (3 sessions)

- *Pot pourri* of spaces
- Creating and using data spaces
- Replacing an SVC with a PC-cp
- Cross-Memory Authorization
- Accessing data in another space
- Sharing data spaces
- Moving data across address spaces
- Offering global services (PC-ss)
- Offering services to restrictricted address spaces
- Beefing up security: more options
- (If time permits:) Choosing: address space, data space, or hiperspace?

12

Lend Me Your EAR: The ART of MVS/ESA™ Programming

Address Space



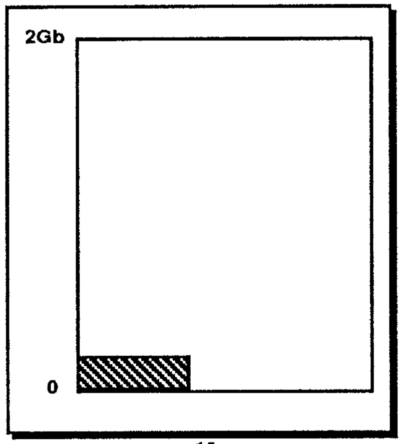
13

Address Space (continued)

- Traditional MVS address space
- Instructions + data
- Fragmented by common area on either side of 16-megabyte line
- Can address up to 16 address spaces simultaneously
- Directly addressable
- Byte addressable
- Backed by real storage

14

Data Space



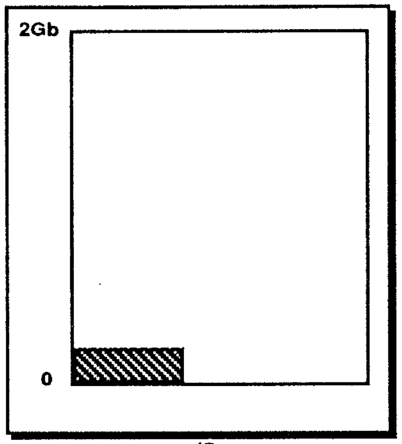
15

Data Space (continued)

- Non-traditional object
  - — No ASCB, ASVT
  - — Managed slightly differently
- Contains only data, no executable instructions
- Either 2 gigabytes or 2Gb-4K usable length
- No common or system areas, hence, no fragmentation
- Can address up to 14 data spaces simultaneously
- Directly addressable
- Byte addressable
- Backed by real storage

16

Hiperspace



17

Hiperspace (continued)

- Non-traditional object
  - — No ASCB, ASVT
  - — Managed differently
- Contains only data, no executable instructions
- Either 2 gigabytes or 2Gb-4K usable length
- No common or system areas, hence, no fragmentation
- Can address any number of spaces simultaneously
- Not directly addressable — must be moved to address space
- Block addressable
- Backed by paging storage, only
  - — expanded storage
  - — auxiliary storage

18

Lend Me Your EAR: The ART of MVS/ESA™ Programming

Data Space Services

DSPSERV CREATE —
creates a data space

DSPSERV DELETE —
destroys a data space

DSPSERV RELEASE —
releases space in a data space for reuse

DSPSERV DEFINE —
defines/undefines data space area for I/O

Notes: Can create data space from DREF (disabled reference) storage.

Cannot use access methods (including VSAM) for data space I/O. May use DIV, however.

19

Creating a Data Space

label	DSPSERV	CREATE,STOKEN=*address* ,BLOCKS=*size-operand* ,CALLERKEY ,FPROT=NO ,ORIGIN=*origin-address* ,MF=S

NOTES:

size-operand may be:

(maximum,initial)

(addr-of-max, addr-of-min)

0 (use installation default)

origin-address specifies a fullword into which the lowest usable location in the data space is placed (0 or 4K)

20

Private Space Facility

- Disables Fetch-Protection-Override
- Disables Low-Address Protection
- Prevents use of common TLB entries not from a different space

* If PSF not active, cannnot use 0-4K of a data space (because of Low-Address Protection)

DSPSERV CREATE returns an ORIGIN depending on whether the hardware can support the use of addresses 0-4K (i.e., whether PSF is active)

21

*Private Space Facility

STOKENs

- 8 bytes (64 bits)
- Not reused within IPL
- Every space has an STOKEN
  - — address space
  - — data space
  - — hiperspace
- MVS/ESA macros tend to use STOKENs, rather than ASIDs, to identify address spaces
- DSPSERV CREATE returns an STOKEN for the data space

22

Access Registers

- 16 access registers associated with 16 general registers used as *base* registers in *operand* accesses, only

GR0	⟷	PRIMARY
GR1	⟷	AR1
GR2	⟷	AR2
⋮	⋮	⋮
GR15	⟷	AR15

Note: The contents of AR0 are not examined

23

Address Space Varieties

Three address space varieties:

- Primary
  - — Same address space from which instructions normally are fetched
  - — Associated with Primary Address Space Number (PASN, a.k.a. PASID))
- Secondary
  - — Also addressable by DAS
  - — Associated with Secondary Address Space Number (SASN, a.k.a. SASID)
- Home
  - — Dispatched address space (ASCB is in PSAAOLD)
  - — May execute instructions from A/S

24

Lend Me Your EAR: The ART of MVS/ESA™ Programming

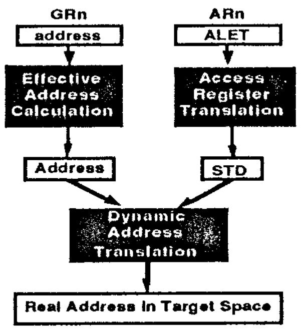
Address Space Control Modes

- Primary space mode
  - — Instructions fetched from primary
  - — Data fetched from primary
- Secondary space mode
  - — Instructions fetched from primary
  - — Data fetched from secondary
- Home space mode
  - — Instructions fetched from home
  - — Data fetched from home
- AR mode
  - — Instructions fetched from primary
  - — Data fetched from space determined by access register translation (ART)

25

Access Register Translation (ART)

For 1 ≤ n ≤ 15:

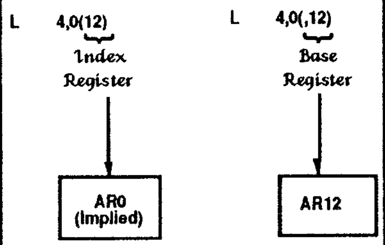


GRn
address
Effective Address Calculation
ARn
ALET
Access Register Translation
Address
STD
Dynamic Address Translation
Real Address In Target Space

26

Base Register vs. Index Register

*



27

Access Lists

- To access a space through an AR, need an access list entry token (ALET)
- An ALET represents an entry in an access list (ALE)
- Every program has two access lists it can use:
  - — Primary address space access list (PASN-AL)
  - — Dispatchable unit access list (DU-AL)

28

Access Lists: PASN-AL vs DU-AL

PASN-AL:

- Available to any program running in this primary address space
- If you switch to another address space (via Program Call — not yet discussed), you change PASN-AL
- Should contain A/S-related entries

DU-AL:

- Available to this dispatchable unit (TCB or SRB)
- If you switch to another address space (via Program Call), the DU-AL goes with you

old PASID/SASID

29

From STOKEN to ALET

Use the ALESERV~~E~~ service to:

- Put an Access List Entry (ALE) into
  - — PASN-AL *or*
  - — DU-AL
- Return an Access List Entry Token (ALET)

Use the Load Access Register Multiple (LAM) instruction to load the ALET into an access register

Load ORIGIN into corresponding general register

30

Lend Me Your EAR: The ART of MVS/ESA™ Programming

ALESERV

- Builds Access List Entry (ALE)
- Generates Access List Entry Token (ALET)

label	ALESERV	ADD,STOKEN=*address* ,ALET=*address* ,AL={WORKUNIT / PASN}

Notes:

- STOKEN is required input
- ALET is returned as output
- AL=WORKUNIT adds ALE to the DU-AL

  AL=PASN adds ALE to the PASN-AL

31

Set Address Space Control (SAC)

Changes PSW bits 16-17:

Operand		Mode
* Decimal		
~~X~~'00'	00	Primary
~~X~~'256'	01	Secondary
~~X~~'512'	10	AR
~~X~~'768'	11	Home

32

Special ALETs

X'00000000'	Primary (hardware)
X'00000001'	Secondary (hardware)
X'00000002'	Home (Software and LAE instruction)

33

Useful AR Instructions

- Copy Access (CPYA)
  - — copies one AR to another
- Extract Access (EAR)
  - — copies an AR to a GR
- Set Access (SAR)
  - — copies a GR to an AR
- Load Access Multiple (LAM)
  - — loads a range of ARs from storage
- Store Access Multiple (STAM)
  - — stores a range of ARs
- Load Address Extended
  - — loads a GR with an address
  - — loads corresponding AR with ALET or zero, depending on ASC mode and base register

34

Sharing Data Spaces

- Passing ALETs
  - — Passed like parameters
  - — Must make sense to receiver. For example:
    - ° Can't pass ALET for ALE on DU-AL to another TCB
    - ° Can't pass ALET for ALE on PASN-AL to program in another address space
    - ° Can't pass ALET for private ALE across a PC interface that changes the EAX

35

Sharing Data Spaces (continued)

SCOPE=SINGLE on DSPSERV CREATE prevents ALET for a data space from being passed beyond the creating A/S

- On PASN-AL
  - — restricted to programs running in owner's A/S
- On DU-AL
  - — restricted to TCBs and SRBs with owner's home address space
- SCOPE=SINGLE is the default

SCOPE=ALL on DSPSERV CREATE:

- Available to programs in any A/S
- Owning A/S must be non-swappable while sharing the data space

36

Lend Me Your EAR: The ART of MVS/ESA™ Programming

Replacing SVC with PC-cp

Suppose an installation-installed SVC routine:

- exists solely to issue privileged instructions or change key
- executes entirely within one address space
- doesn't need system locks
- doesn't need to receive control disabled for interruptions

Why not convert this SVC into a PC-cp?

37

Replacing SVC with PC-cp (continued)

Drawbacks of SVCs:

- Only 256 SVCs
- IBM owns all SVCs below 200
- ESRs are below 200
- Vendor products frequently require SVCs
- Installations usually have local SVCs
- SVCs receive control supervisor state key 0 — no flexibility

38

Why convert SVC to PC-cp

- Avoid an SVC interruption
- Bypass SVC table processing, lock management
- Bypass SVC exit processing
- Restrict access to callers running in certain protection keys
- Restrict access to callers who know the right PC number
- Quickest way into supervisor state or system key
- Control when the routine is available:
  - — PC is dynamic
  - — Routine can be replaced without re-IPL
  - — Maximum duration: one IPL
  - — Security hooks

39

Creating a PC-cp Routine

- Generate a PC-number
  - — Linkage index
  - — Entry-table entry
- Communicate routine characteristics to the operating system:
  - — Routine name or entry point
  - — Authorization key mask
  - — Execution key mask
  - — State (problem or supervisor)

° Communicate PC-number to potential callers

° Design and write the routine

° Code PC instructions in calling routine

40

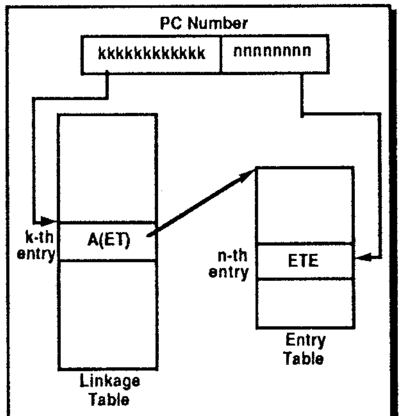
Generating PC-number

- PC number is 20 bits long — bits 12-31 of PC instruction effective address
- High-order 12 bits (12-23) are the *linkage index* (LX)
  - —Used to locate the *Entry Table*
- Low-order 8 bits (24-31) are the *entry index* (EX)
  - —Used to locate *Entry-Table Entry* (ETE) within the Entry Table

000000000000	LLLLLLLLLLLL	EEEEEEEE
0	12	24 31

41

PC Routine Lookup



42

Lend Me Your EAR: The ART of MVS/ESA™ Programming

Reserving a Linkage Index

To get a linkage index assigned, use the LXRES macro

```
          LA      0,LXLIST
name      LXRES   LXLIST=(0),SYSTEM=YES
            :
LXLIST    DC      F'1'              Number of LXs
          DS      F                 First (only) LX
```

- For global resource, request system LX
- System linkage index ETs automatically connected to *all* address spaces

43

Entry Table Definition

To define an Entry Table (ET), use the ETDEF macro

TYPE=INITIAL	builds a header
TYPE=ENTRY	describes a PC routine
TYPE=FINAL	signals end of Entry Table

44

ETDEF TYPE=ENTRY

{ PROGRAM= / ROUTINE= }	{ program name / routine address }
STATE=	PROBLEM or SUPERVISOR
SSWITCH=	YES (PC-ss) NO (PC-cp)
ASCMODE=	PRIMARY or AR
EAX=	Extended Authorization Index

(continued)

45

ETDEF TYPE=ENTRY (continued)

EK=	PSW key upon entry
PKM=	OR or REPLACE
AKM=	list of acceptable caller keys
EKM=	list of execution key
RAMODE=	31 *or* 24
SASN=	OLD *or* NEW

46

Entry Table Creation

To create an Entry Table, use the ETCRE macro

```
          LA      2,ENTYTABL
name      ETCRE   ENTRIES=(2)
            :       :
TKLIST    DS      0F                Token List
TKCNT     DC      F'1'              Token Count
TKVAL     DS      F                 Token Value
            :       :
ETBL      DS      0D                Entry Table
```

47

Must be in supervisor state and in Key0-

Connecting the Entry Table

To associate the *Entry Table* with the *Linkage Index*, use the ETCON macro

```
          LA      0,TKLIST          Token List
          LA      2,LXLIST
name      ETCON   LXLIST=(2),TKLIST=(0)
            :       :
TKLIST    DS      0F                Token List
TKCNT     DC      F'1'              Token Count
TKVAL     DS      F                 Token Value
LXLIST    DC      F'1'              Number of LXs
          DS      F                 First (only) LX
```

TkLIST → ?

48

Lend Me Your EAR: The ART of MVS/ESA™ Programming

Locating the PC Routine

- Build a PC number table
  - — maps service number to PC number
  - — macro addressing service number can retrieve assicuated PC number
- Package all Cross-Memory parameters together in CSA (e.g., an XMEM vector table
- Use some commonly-addressable anchor — for example:
  - — The Subsystem Interface
  - — CVTUSER
  - — A data set

This is frequently one of the most difficult issues in designing Cross-Memory Services application

49

Building a PC Number

The LXRES macro returns the LX value in the form:

00 0L LL 00

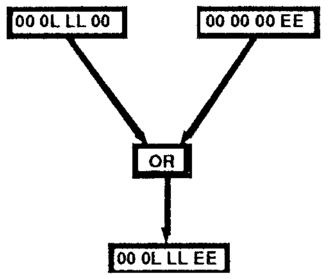
The EX corresponds to the entry number in the ET. (Numbering begins with 0).

The EX occupies only the low-order byte:

00 00 00 EE

50

Building a PC Number (continued)



REMINDER: EX=8 corresponds to the *ninth* entry in the table

51

PC-cp Routine

- Should be re-enterable
- No base register — must issue BALR or BASR instruction
- Receive control in ETE-specified key
- May change to any key in (extended) PSW Key Mask
- Receive control in
  - — supervisor state
  - — problem state
- State, Key and key masks specified in ETDEF macro

52

PC-cp Routine (continued)

- Loaded under job step task of A/S that created ET
- PC-cp routine may request system services, since it is not in Cross-memory mode (More on this later!)
- Registers 0, 1, and 15 used to pass parameters
- Exit using PR
- Must not use Checkpoint/Restart
- If using system services, supply register save area in GR13

53

Stacking PC

Two types of PC instructions:

- Stacking
- Non-stacking — old form

We shall only discuss the stacking PC

In stacking PC, the following are saved automatically:

- General purpose registers (GRs)
- Access Registers (ARs)
- PSW Key Mask (PKM)
- Extended Authorization Index (EAX)
- PASN and SASN
- PC number

54

Program Return (PR)

- Restores:
  - GRs 2-14
  - ARs 2-14
- Returns control to next sequential instruction after the PC

55

Cross-Memory Authorization

- Authorized Program Facility
- Authority Table Authorization
- PSW Key Mask Authorization
- Private Space Authorization

57

Supervisor State Key 0-7

The following macros require caller in either supervisor state or key 0-7, enabled and unlocked:

- AXRES
- AXSET
- ATSET
- LXRES
- ETCRE
- ETCON

APF-authorized program can use MODESET to enter supervisor state or system key

59

Useful Stack Instructions

- Extract Stacked Registers (EREG)
  - Returns range of GRs
  - Returns corresponding ARs
- Extract Stacked State (ESTA)
  - Returns non-register stack info
  - Returns 8-byte modifiable area
- Modify Stacked State (MSTA)
  - Stores 8-bytes into modifiable area
- Branch and Stack (BAKR)
  - Creates a branching stack entry
  - Saves GRs, ARs, PSW, *et al.*
  - Unstacked by PR

56

Authorized Program Facility (APF)

A program is APF Authorized if

- running in supervisor state *or*
- running with system key (0-7) *or*
- the following hold simultaneously:
  - linked into authorized library
  - linkedited with PARM='AC=1' or SETCODE AC(1)

APF-authorized user is somewhat like a UNIX superuser

An APF-authorized program can get into supervisor state or change key by issuing the MODESET macro

58

Authority Table Authorization

AXRES	Reserve an Authorization Index for use by the A/S
AXSET	Set the address space's active AX to a specific reserved AX
ATSET	Set the address space's Authority Table (AT)

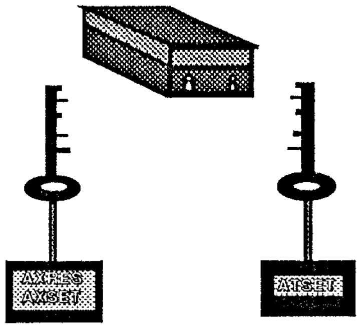
Notes:

- Every address space has an Authority Table
- *n*th governs access by address spaces having active AX=n

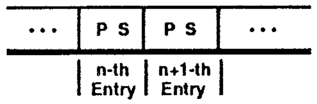
60

Lend Me Your EAR: The ART of MVS/ESA™ Programming

Authority Table Authorization (continued)



Authority Table Entry



"P" = Authorization bit for PT instruction

"S" = Authorization bit for SSAR instruction

Wresting AT Authorization

1. Invoke AXRES to reserve an AX.
2. Invoke AXSET to set address space to the reserved AX.
3. Schedule SRB to run in other address space.
4. SRB invokes ATSET to set the AT entry in the other address space.
5. SRB posts originating routine.

Note: Must be APF-Authorized

Global AXs

AX=~~0~~1 All cross-memory authorization *permitted*

AX=~~1~~0 All cross-memory authorization *denied*

Note: Can issue AXSET AX=1 or 0; do not have to reserve AXs 0 and 1.

Must be APF-Authorized to issue AXSET

PSW Key Mask Authorization

Simian scheme:

Authorization key mask

Program status word key mask

Execution key mask

PSW Key Mask

String of 16 bits, numbered 0-15. Every work unit (TCB or SRB) has a PKM.

If the n-th bit is 1, then:

- Storage may be set to key *n* by the SPKA instruction
- PC instruction may call a routine available to callers with key *n* authorization
- Certain instructions may access storage having protect key *n*

  (e.g., MVCP, MVCS)

PKM for key 0, 2, 8, and 15:

1010 0000 1000 0001

AKM and EKM

Authorization Key Mask:

- String of 16 bits, numbered 0-15
- Every PC routine has an AKM
- n-th bit = 1 means callers with key *n* PKM-authorization may PC to this routine

Execution Key Mask:

- String of 16 bits, numbered 0-15
- Every PC routine has an EKM
- EKM used to form called routine's PKM:
  - — If PKM=OR on ETDEF, then old PKM and EKM are OR'ed to form new PKM
  - — If PKM=REPLACE on ETDEF, then EKM becomes the new PKM

67

QUIZ

Suppose:

•Problem task executing in key 8
•PKM authorizes keys 4, 5, and 8

•Supervisor-state PC routine defined
•AKM specifies keys 0 and 4
•EKM specifies keys 0-3

•Task invokes the PC routine

Questions:

1. Will the PC routine gain control?
2. If so, with what PKM will the PC routine execute
   a) when PKM=OR for the PC routine?
   b) when PKM=REPLACE for the PC routine?
3. In which state will the PC routine execute? Why?

68

Answers

Task PKM is	0000 1100 1000 0000
PC AKM is	1000 1000 0000 0000
ANDing:	0000 1000 0000 0000

Since the AND is non-zero, caller is authorized

Task PKM is	0000 1100 1000 0000
PC EKM is	1111 0000 0000 0000
ORing:	1111 1100 1000 0000

With PKM=OR, the new PKM will authorize 0-5 and 8, supervisor state

With PKM=REPLACE, the new PKM will authorize 0-3, supervisor state

69

Cross-Memory Data Movement — Pre-ESA

- Move to Primary (MVCP) *or* Move to Secondary (MVCS)
  - — 256 bytes at a time
  - — check authority each time
- Move Long (MVCL) to CSA

  PC-ss to other A/S

  Move Long from CSA
  - — two moves
  - — required allocating CSA buffer
  - — required code in both spaces
- Need authorization to other space

70

ESA Cross-Memory Data Movement

- MVC or MVCL directly
  - — source operand in one A/S
  - — target operand in other A/S
- Need to have one of following:
  - — ALET of other A/S
  - — Authorization to other space as secondary address space
- With the ALET, require authorization to other space if it is *private*

71

Cross-Memory Access With ALET

For a public space:

- Load the base register with address in target A/S
- Load the ALET into corresponding AR
- SAC to AR-mode
- Process as with data spaces

Private spaces will be discussed later

72

Cross-Memory Access Without ALET

If AT-authorized to target A/S:

- Must know target address space ASID
- SSAR to target address space
- Load base register with address in target A/S
- Load corresponding AR with ALET X'00000001'
- SAC to AR-mode
- Process as with data spaces

Note: Authority Table authorization to target space is required for SSAR

Wresting the ALET

1. Schedule SRB to other address space and WAIT.
2. SRB routine to obtains STOKEN of other (home) address space using ALESERV EXTRACTH .
3. SRB posts originating routine.
4. Add the ALE to either the DU-AL or the PASN-AL using ALESERV ADD. Must specify CHKEAX=NO
5. Load the base address and ALET into a GR/AR pair.

Creating an Address Space

```
name   ASCRE   ASNAME='space-name'
               ,STPARM=  parm-addr
               ,INIT=  routine name or
                       routine address
               ,ODA=  output data
                      address
               ,AXLIST= AX-list-addr
               ,TKLIST= token-list addr
               ,LXLIST= LX-list addr
               ,ASPARM= parm-string
                       addr (accessed
                       by ASEXT)
```

These are only some of the operands

Creating an Address Space (continued)

AXLIST=
- Used to pass AXs (32 maximum)
- Each AX in list will become AX operand of ATSET from new A/S (PT=YES,SSAR=YES)
- Enables creating A/S to access new A/S

TKLIST=
- Used to pass ET tokens (32 max.)
- Will become TKLIST operand of ETCON from new A/S

LXLIST=
- Used to pass LXs (32 maximum)
- Will become LXLIST operand of ETCON from new A/S

TKLIST and LXLIST enable new A/S to access creating A/S

Creating an Address Space (continued)

<u>Output Data Address (ODA= operand)</u>:

STOKEN
A(ASCB)
EAERIMWT ECB
EAEASWT ECB
Reserved

Creating an Address Space (continued)

- ASCRE returns STOKEN
- Issue ALESERV ADD to add an ALE and get an ALET
  - — Must be EAX-authorized — *or*
  - — Can specify CHKEAX=NO if supervisor state or key 0-7
- Can be addressed like data space
- Initialization routine used to:
  - — Load routines
  - — LXRES, ETDEF, ETCRE, ETCON
- A/S can be addressed through PC-ss

Lend Me Your EAR: The ART of MVS/ESA™ Programming

Creating an Address Space (continued)

Initialization Routine:

- Specified in INIT= operand of ASCRE
- Can be used to
  - — load service routines
  - — build entry tables
- Can communicate with creating A/S using two ECBs
  - — EAERIMWT
  - — EAEASWT
- Return codes:

  0 — Continue A/S Initialization

  4 — Terminate

79

Creating an Address Space (continued)

Creator	Initialization Routine
Invoke ETCRE	
Wait on EAERIMWT	
	Load PC Routines
	Build ETs
	Post EAERIMWT with Go/NoGo status
Optionally pass more parameters	Wait on EAEASWT
Post EAEASWT	
	Optionally continue processing
	Return to system

80

Extended Authorization Index (EAX)

Extended Authorization Index is a value in a control register

- Set by PC

  EAX=*eax-value* specified in ETDEF when creating PC routines
- Previous value restored by PR
- Must be EAX-authorized to A/S to invoke ALESERV ADD to:
  - — create ALE for A/S
  - — return an ALET

81

EAX Authorization

A program is EAX-authorized to a space if <u>any</u> of the following obtains:

- The space has public access
  - —the ALE is marked public-access
  - —all data spaces are public-access
- The EAX is the same as the ALEAX
- The space's AT grants authorization
  - —the EAX indexes into the AT
  - —the S-bit in the ATE is 1
  - —similar to AX

82

Instituting Private Access

To make one's own space private-access:

- Issue AXRES (optional)
- Locate ALE
  - — ART described in Principles of Opaeration Manual
  - — Use ALET and pointers to DU-AL or PASN-AL to locate the ALE
- Set private-access bit in the ALE
- Set the ALEAX to desired value

  Other A/S with AX equal to this ALEAX may access this A/S
- Use ATSET to set S-bit in desired ATEs

  Any other A/S's with AX matching ATE with S-bit set may access this A/S

83

Cross-Memory Mode

- Home, primary, and secondary spaces are not all the same space

  <u>or</u>
- Executing in secondary-space mode

84

Lend Me Your EAR: The ART of MVS/ESA™ Programming

Swappability

The following are non-swappable:

- Home space
- Space marked non-swappable in the Program Properties Table (PPT)
- Spaces made non-swappable by SYSEVENT TRANSWAP or SYSEVENT DONTSWAP
- Spaces whose local lock is held (locally or as CML)

The following must be non--swappable:

- Target of PC, PT, PR
- Source and Target of MVCP, MVCS

85

AR-Mode Restrictions

- Must issue SYSSTATE ASCENV=AR for proper AR-mode expansions
- Must issue SYSSTATE ASCENV=P for proper primary-mode expansions
- Must use corresponding X-macro:
- * Use [STORAGE] macro to obtain storage in A/S

86

Cross-Memory Mode Restrictions

- May not issue SVC (except ABEND)
- May not use system services unless documented as specifically available in Cross-Memory Mode
- Only one job step may create ETs with space-switching ETEs
- Subsequent job steps cannot use AXRES, LXRES, or ETCRE
- PC routines may not use Checkpoint/Restart

87

Similarities Between PC-cp and PC-ss Routines

- No base register coverage — must issue BALR or BASR on entry
- Receive control in key designated in ETE
- Supervisor/Problem state as designated in ETE
- Usually re-enterable
- Exit via Program Return (PR)

88

Differences Between PC-cp and PC-ss Routines

- PC-ss routines execute in other address spaces
- PC-ss routines may not use some system services
- Space switching complicates error recovery
- Space switching PC and PR may generate space-switching event interruptions
- More likely to reside in private area

89

Review: Service to All Address Spaces

- Issue LXRES with SYSTEM=YES
- Invoke ETDEF and ETCRE to construct Entry Table
  - — ET entries define PC routines
  - — Current primary or space-switching
- ETCON to own address space connects the ET to every space in the system

90

Lend Me Your EAR: The ART of MVS/ESA™ Programming

Address-Space Specific Service Authorization

- Issue LXRES with SYSTEM=NO
- Invoke ETDEF and ETCRE to construct Entry Table
  - — ET entries define PC routines
  - — Current primary or space-switching
- Schedule SRB to specific other A/S
  - – SRB routine issues ETCON to connect the ET to the home (i.e., other) address space

91

Providing Services Across Two Address Spaces

Service Provider Address Space	Application Address Space
Reserve an AX (AXRES)	
Set the AX for the A/S to the reserved value (AXSET)	
Define and create an Entry Table (ETDEF, ETCRE)	
Schedule SRB to run in application A/S	
	Set AT entry (ATSET) corresponding to specified AX
Wait on ECB	
	Reserve an LX (LXRES)
	Connect specified ET to reserved LX (ETCON)
	XMPOST the ECB
Resume Processing	

92

Hiperspaces

Hiperspaces are:

- 4K-2G long
- block-manipulable
- *not directly addressable* — must be moved to main storage
- backed by expanded storage and/or auxiliary storage
- data-only spaces
  - — no system areas
  - — cannot execute instructions

93

Hiperspaces (continued)

Two types of Hiperspace:

- Scroll (a.k.a. *standard*)
  - — backed by expanded and auxiliary storage
  - — authorized and non-authorized
  - — no loss of data if ES fills up
  - — large permanent objects
- Cache (a.k.a. *expanded storage only*)
  - — Backed only by ES
  - — authorized callers, only
  - — data discarded as ES fills up
  - — temporary objects

94

Hiperspace Services

- DSPSERV CREATE
- DSPSERV DELETE
- DSPSERV RELEASE
- DSPSERV DEFINE

- HSPSERV — to move blocks between hiperspace and main storage

95

Using Hiperspaces

1. Create using DSPSERV, providing
   - name
   - size in 4K blocks
   - area for space token (STOKEN)
   - area for ORIGIN (0 or 4K)
   - hiperspace type
2. Use STORAGE service to allocate a main storage window of desired size
3. Use HSPSERV to move data between hiperspace and main storage window, providing
   - STOKEN
   - address range
     - — source origin
     - — target origin
     - — number of blocks

96

Lend Me Your EAR: The ART of MVS/ESA™ Programming

Hiperspace Applications

<u>Scroll type</u>
- Similar to data space applications
- Isolate data
  - — only *window* is addressable
- Suitable for large amounts of data to avoid using real storage, paging storage
- Avoid using AASF or access register mode

<u>Cache-type</u>
- Expanded Storage Only
- Used to create software analogue of a cache
- Pages discarded — LRU algorithm

97

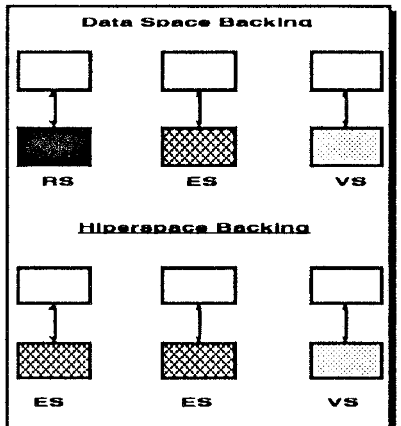
Address Space *vs.* Data Space

<u>Address space:</u>
- •Avoids ART
- •Can intermix instructions and data
- • Can require EAX authority for access
- •Creation requires supervisor state or key 0-7
- •Adding an ALE requires EAX-authority

<u>Data space:</u>
- •Provides VSCR
- •Avoids space fragmentation
- •Provides data isolation
- •Finer granularity of data sharing
- •Can create and access without being authorized

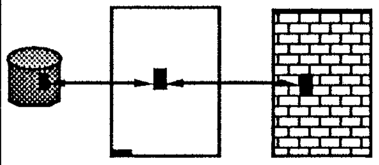
98

Data Space vs Hiperspace - 1



99

Data Space vs Hiperspace - 2



- Data space and hiperspace pages both move through main storage
- Scroll-type hiperspace can limit number of main storage pages
- Data space can avoid using expanded storage when main storage is available

100

Summary: What We Covered

- Address spaces, data spaces and hiperspaces
  - — Creating
  - — Accessing
  - — Sharing
- Replacing an SVC with a PC-cp
- Cross-Memory Authorization
- Moving data across spaces
- Offering services
  - — global
  - — restricted to specific address spaces
- Choosing: address space, data space, or hiperspace?

101

Summary: What We Omitted

- Control Registers
- ALDs and ALEs
- PCAUTH
- HASID, PASID, SASID, and ASID Rain
- PASTEs and DUCTs
- AFT, AST, and other such tables
- EPAR, ESAR, LASP, IAC, and other such instructions (Look them up in the Principles of Operation manual)
- Cleaning up — AXFRE, LXFRE, ETDIS, ETDES
- Debugging (Don't make any misteaks!)
- Recovery (Don't ABEND or crash)
- How to handshake between address spaces (application design)

102

Bibliography

SA22-7200	ESA/370 Principles of Operations
GA28-1854	MVS/ESA SPL: Application Development Guide — Extended Addressability
GA28-1852	MVS/ESA SPL: Application Development Guide
GA28-1857	MVS/ESA SPL: Application Development Macro Reference
GC28-1843	MVS/ESA Callable Services for High Level Languages