z/OS Introduction and Workshop

Operating System Overview

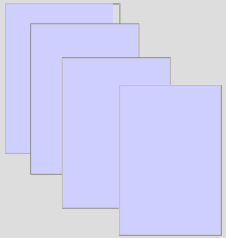
Unit objectives

After completing this unit, you should be able to:

- Describe an address space
- Describe virtual storage
- Describe paging
- List 3 types of address spaces
- List 3 types of memory storage
- Describe system integrity using key-controlled protection

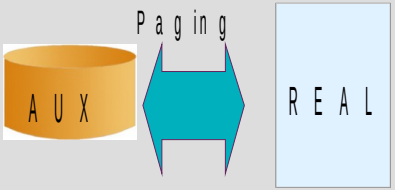
Hardware resources managed by z/OS

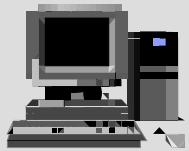
z/OS Environment



Address spaces

Physical storage





Operator communication

Reliability, availability, and serviceability

Data integrity

System Tasks – No need to know it all

How the operating system works is not required
to develop, maintain and port business applications

Pages, Frames and Slots

PSA
CVT
ASTE
ASVT
ASCB
CSA
Master Scheduler
PCAUTH
RASP

DAT
RSM
ASM
VSM
Virtual Storage
WLM
SVC
Protect Keys

What is z/OS?

System z 'flagship' operating system

64-bit operating system

Ideally suited for processing large workloads for many concurrent users

Designed for:

1) Serving 1000s of users concurrently
2) I/O intensive computing
3) Processing very large workloads
4) Running mission critical applications securely

Operating System

Comprised of modules, system programs (macros), system components

Information about the system, resources, and tasks are in *control blocks*

Management of physical storage:

1) Real storage
2) Auxiliary storage
3) Virtual storage

System Tasks are known as Address Spaces

z/OS and its related subsystems require address spaces of their own to provide a functioning operating system.

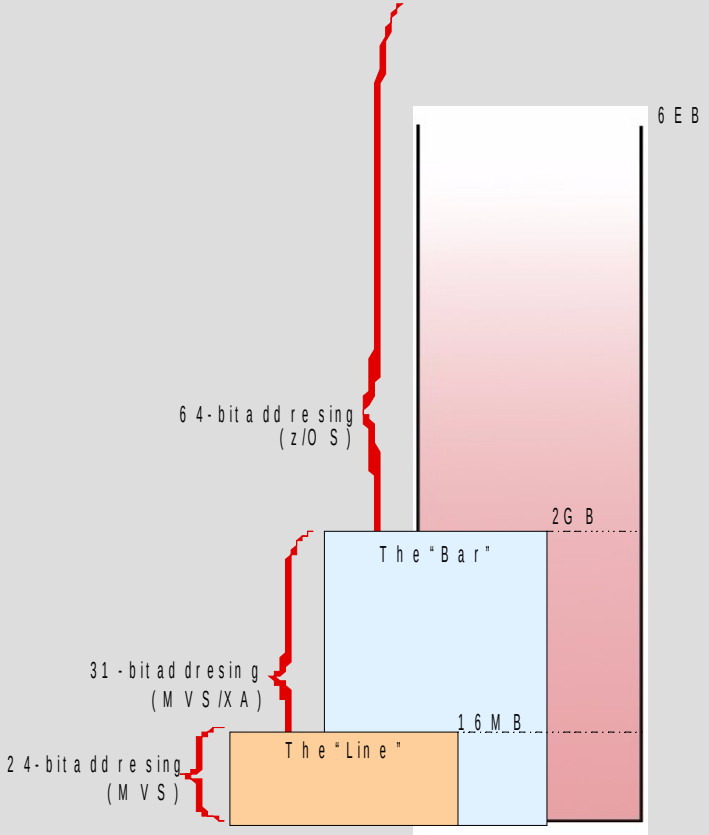
System address spaces are started after initialization of the master scheduler. These address spaces perform functions for all the other types of address spaces that start in z/OS.

Middleware address spaces exist for major system functions and middleware such as Db2, CICS, and IMS

TSO/E address spaces are created for every user who logs on to z/OS

Address spaces are created for every batch job that runs on z/OS.

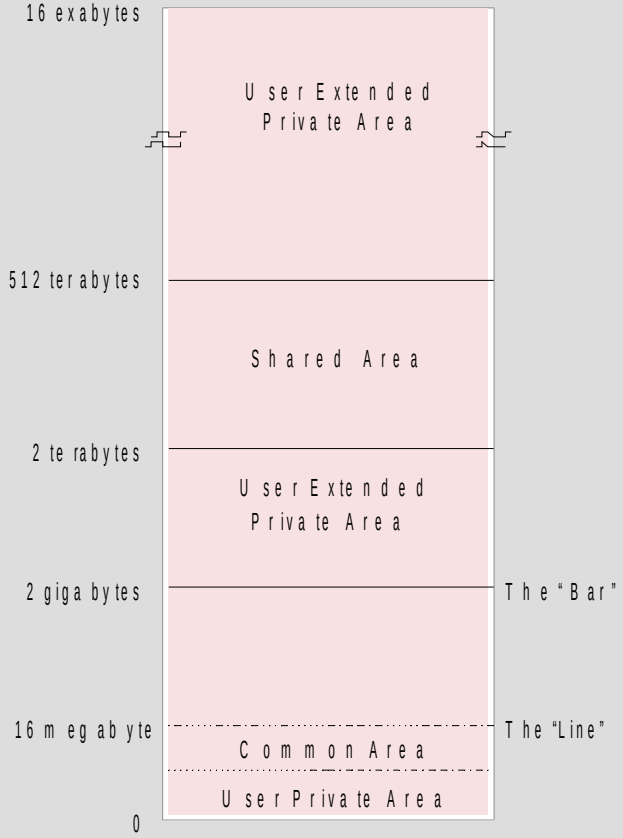
The address space concept



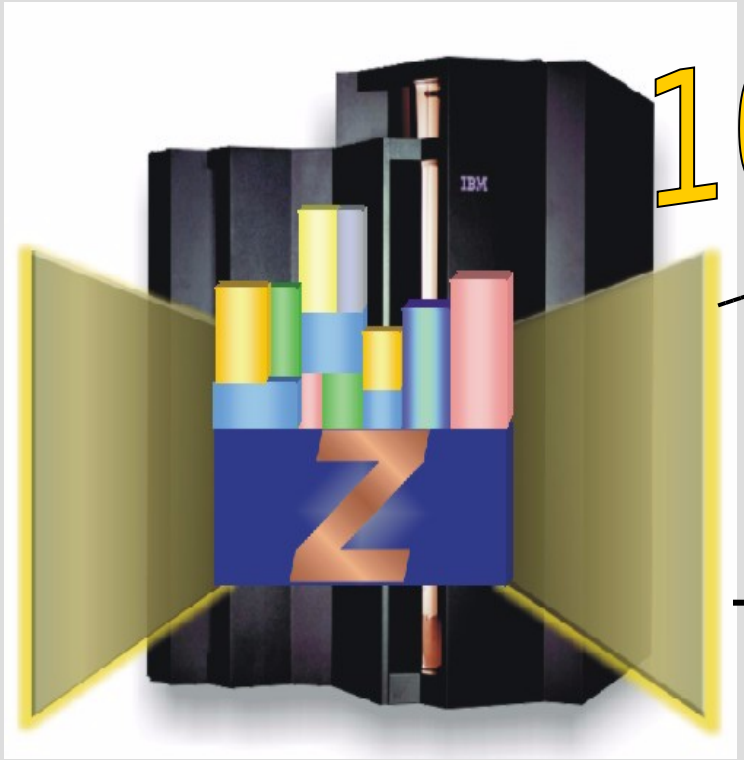
"Bar" *(31 bit) max address*
2 GB of address locations

"Line" *(24 bit) max address*
16 MB of address locations

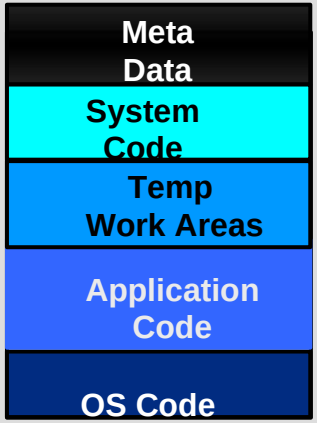
64-bit address space map



User Runtime Container: Address Space



100s to 1000s



Work Load Management (WLM)

With workload management, you define performance goals and assign a business importance to each goal.

Goals:

- **Response-Time**
- **Execution Velocity**
- **Discretionary**

Importance level (1-5)

Goal is 1 or below (meeting goals)

All is well

Goal is above 1 (failing to meet goals)

Revise performance goals or increase capacity

Middleware for z/OS are a collection of address spaces

- Middleware is typically something between the operating system and an end user or end-user applications.

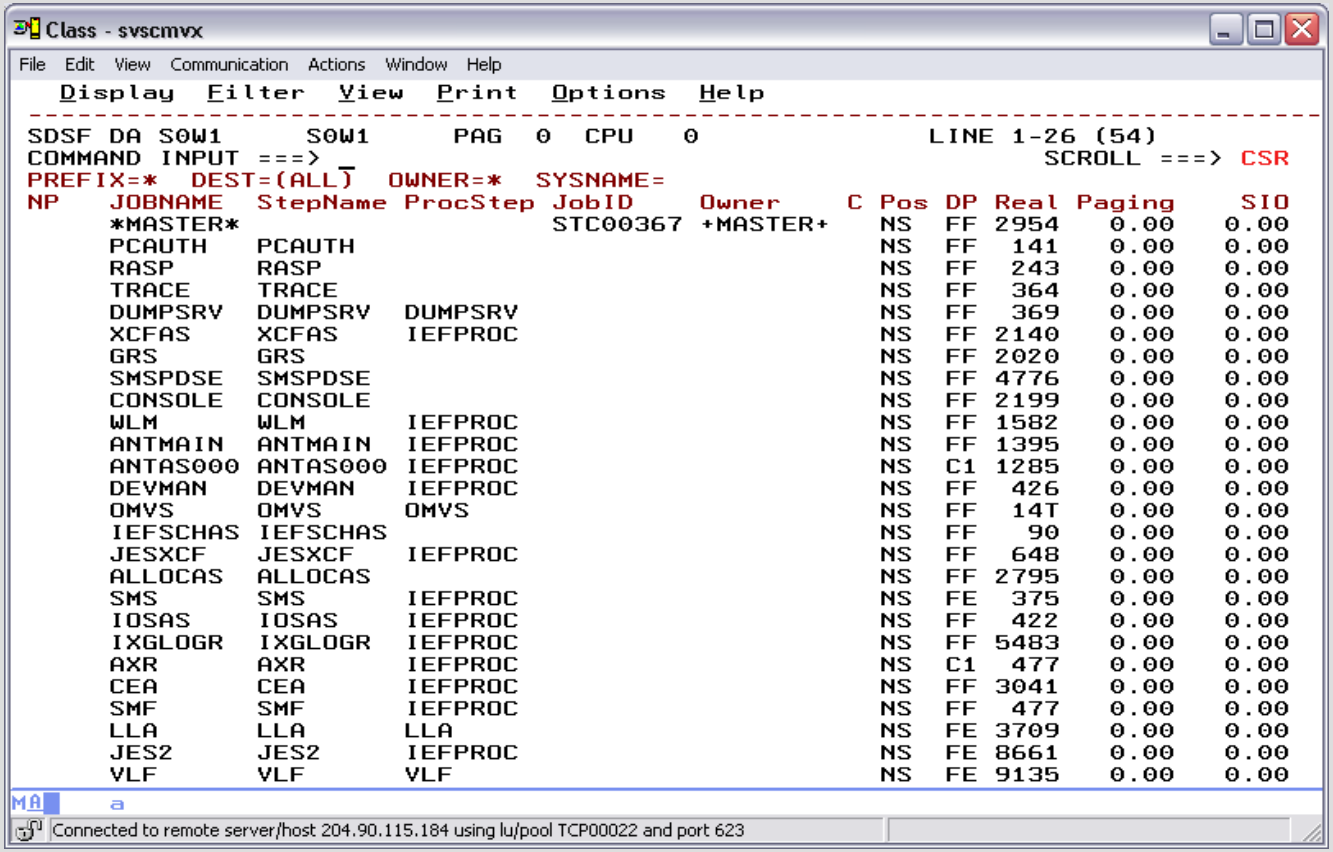
- Middleware supplies major functions not provided by the operating system.

- Typical z/OS middleware includes:
  - Database systems
  - Web servers
  - Message queuing and routing functions
  - Transaction managers
  - Java virtual machines
  - XML processing functions

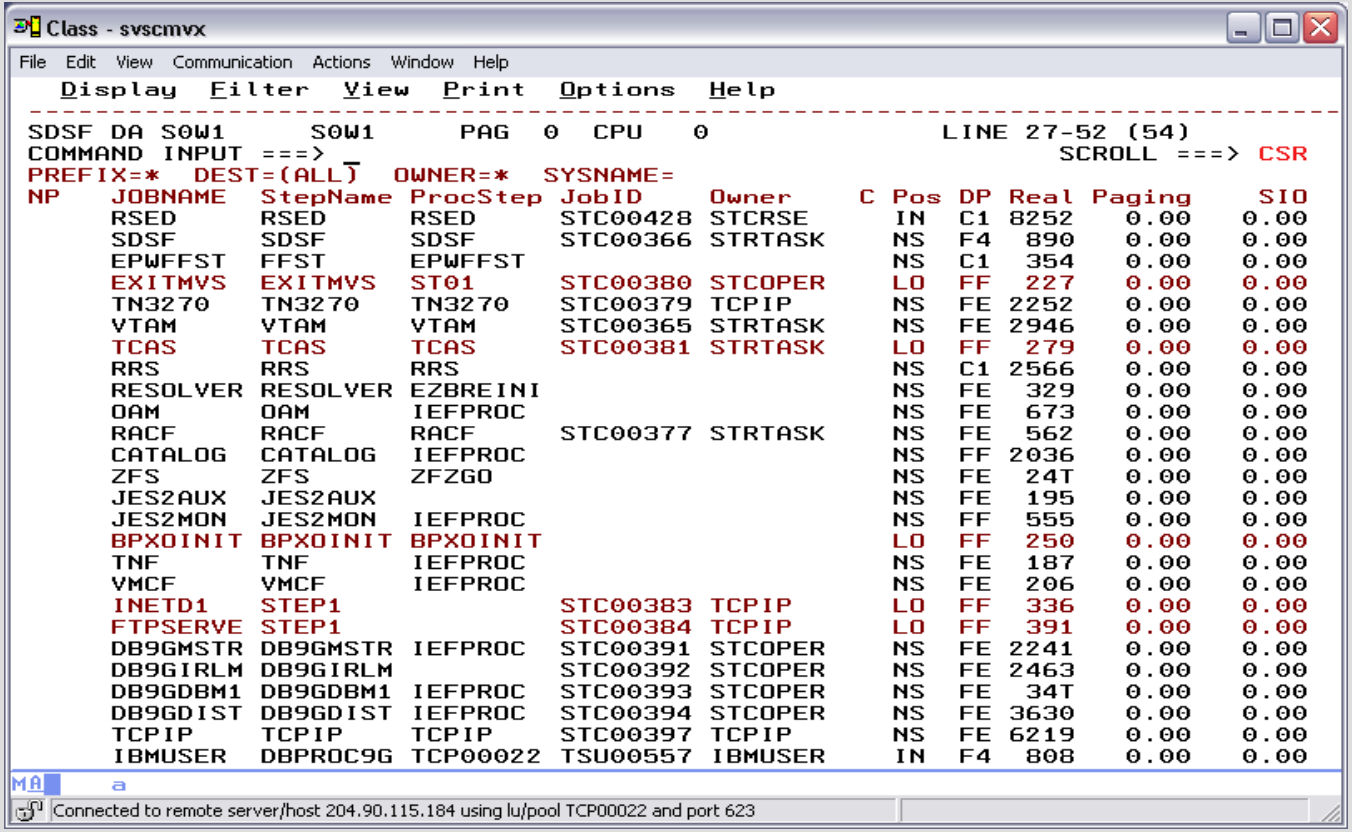
Defining characteristics of z/OS

- Uses address spaces to ensure isolation of private areas
- Ensures *data integrity*, regardless of how large the user population might be.
- Can process a large number of concurrent batch jobs, with automatic workload balancing
- Allows security to be incorporated into applications, resources, and user profiles.
- Allows multiple communications subsystems at the same time
- Provides extensive recovery, making unplanned system restarts very rare.
- Can manage mixed workloads
- Can manage large I/O configurations of 1000s of disk drives, automated tape libraries, large printers, networks of terminals, etc.
- Can be controlled from one or more operator terminals, or from application programming interfaces (APIs) that allow automation of routine operator functions.

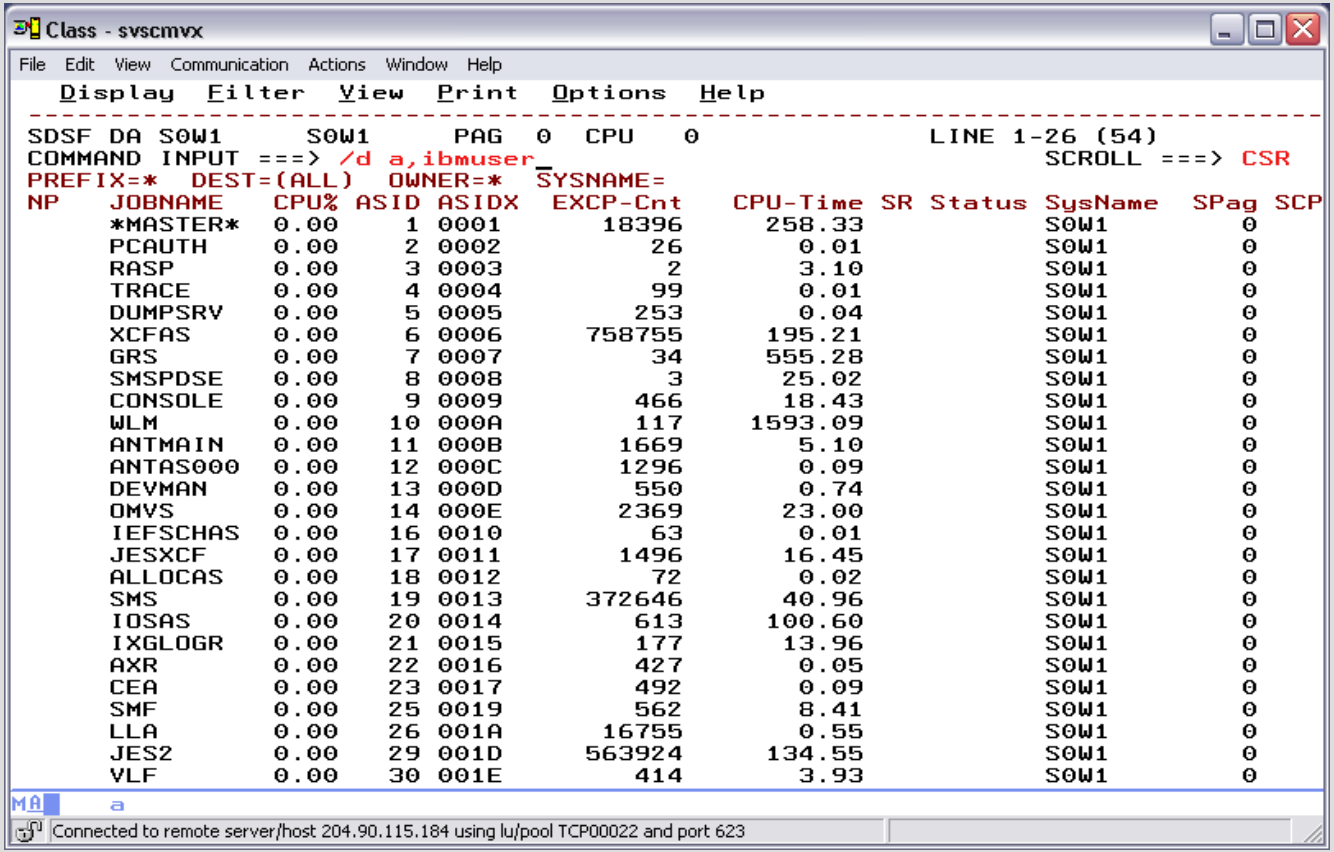
Address Spaces – SDSF display active



Address Spaces – SDSF display active …forward (F8)

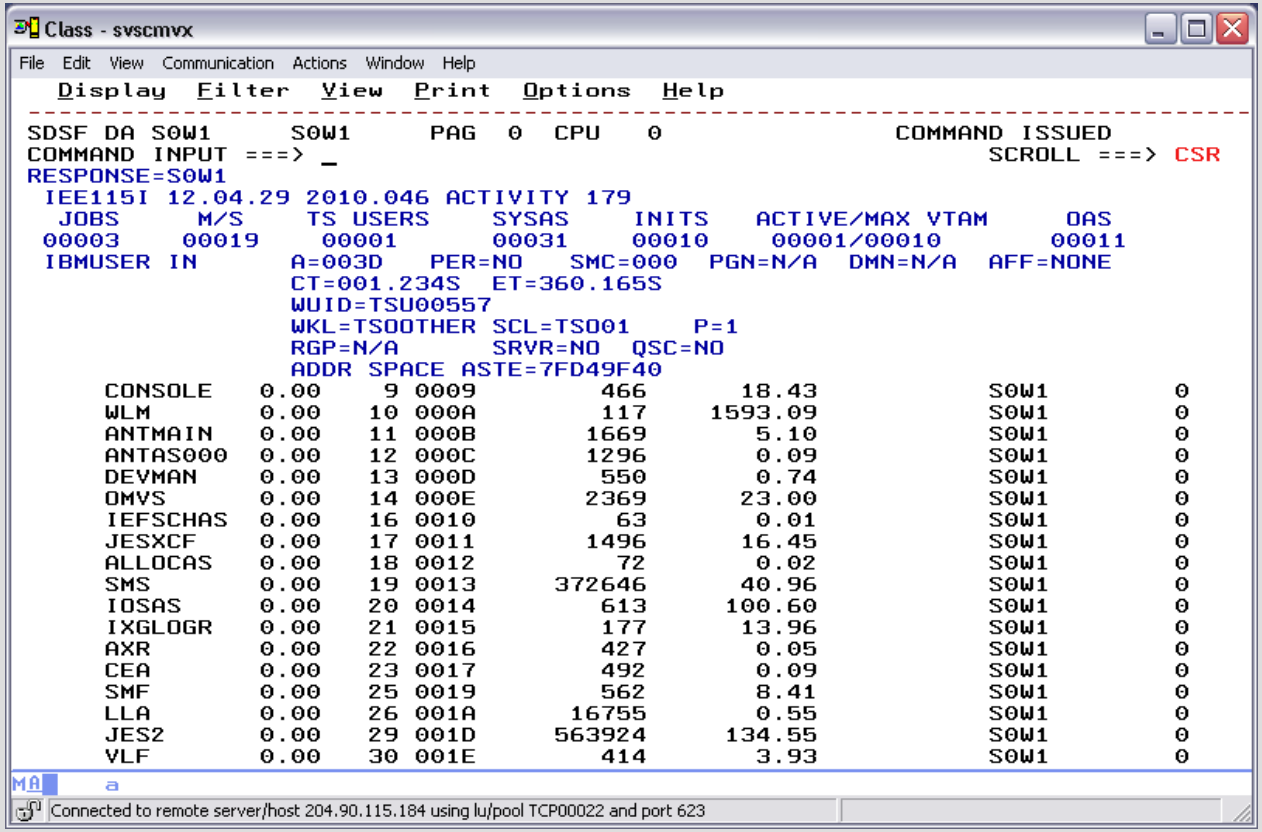


Address Spaces – SDSF display active …shift right (F11)



Address Spaces – MVS command

..display IBMUSER address space output



Address Space – Let's Look Up Message ID IEE115I

IBM Look@ Web Site

Provides status information about job or task
Provides status information about system

What is “IN” immediately following job or task name?
What is “A=” ?
What is “WUID=” ?
What is “CT=” and “ET=”?
What is “WKL=”, “SCL=” and “P=”?

Address Space – A technical definition

An address space is a consecutive sequence of integer numbers (virtual addresses), together with the specific transformation parameters which allow each number to be associated with a byte location in storage. The sequence starts at zero and proceeds left to right.

When a virtual address is used by a CPU to access main storage, it is first converted, by means of dynamic address translation (DAT), to a real address

What is in an address space?

z/OS provides each user with a unique address space and maintains the distinction between the programs and data belonging to each address space.

While an address space includes system code and data as well as user code and data, it maps all of the available addresses. Thus, not all of the mapped addresses are available for user code and data.

System Tasks - Virtual storage concepts

Virtual storage is an “illusion” created through z/OS management of real storage and auxiliary storage through tables.

The running portions of a program are kept in real storage; the rest is kept in auxiliary storage

A contiguous range of addressable virtual storage available to a user or program or the operating system is an *address space*

Each user or separately running program is represented by an address space (each user gets a limited amount of private storage)

System Tasks - How virtual storage works

Virtual storage is divided into 4-kilobyte ***pages***

Transfer of pages between auxiliary storage and real storage is called ***paging***

When a requested address is not in real storage, an interruption is signaled and the system brings the required page into real storage

z/OS uses tables to keep track of ***pages***

Dynamic address translation (DAT)

Frames, pages, slots are all repositories for executable code and data.

System Tasks – Pages, Frames and Slots

The pieces of a program executing in virtual storage must be moved between real and auxiliary storage:

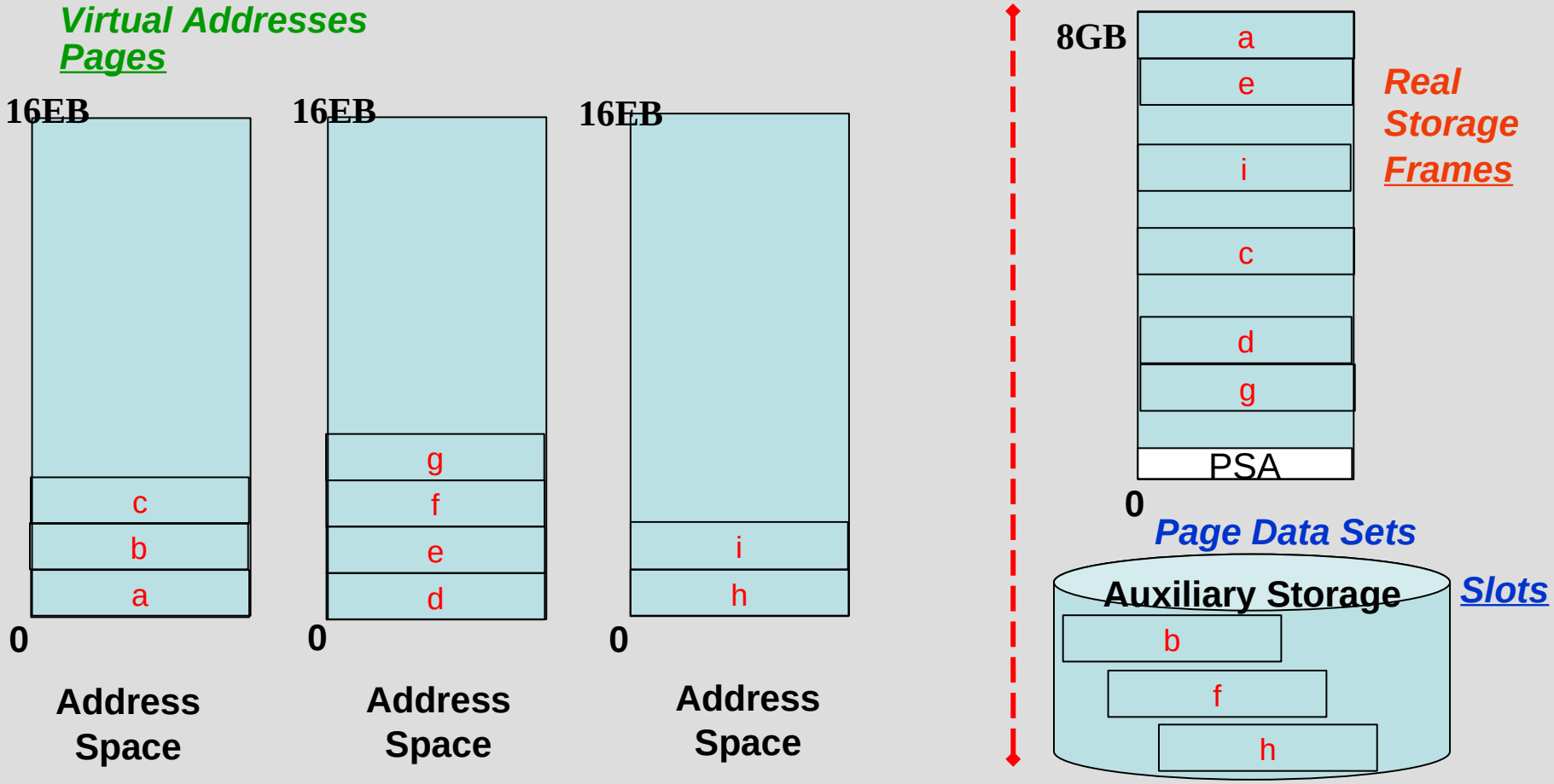
A block of virtual storage is a ***page***

A block of real storage is a ***frame***

A block of auxiliary storage is a ***slot***

A ***page***, ***frame*** and ***slot*** are all the same size: 4096 bytes (4 kilobytes – 4K)

To the programmer, the entire program appears to occupy contiguous space in real storage at all times.


Virtual Addresses
Pages
16EB
16EB
16EB
c
b
a
g
f
e
d
i
h
0
0
0
Address Space
Address Space
Address Space
8GB
a
e
i
c
d
g
PSA
0
Real Storage Frames
Page Data Sets
Auxiliary Storage
b
f
h
Slots

Page Stealing

z/OS tries to keep an adequate supply of available real storage frames on hand.

When this supply becomes low, z/OS uses *page stealing* to replenish it.

Pages that have not been accessed for a relatively long time are good candidates for *page stealing*.

z/OS also uses various storage managers to keep track of all pages, frames, and slots in the system.

Swapping

Swapping is one of several methods that z/OS uses to balance the system workload and ensure that an adequate supply of available real storage frames is maintained.

Swapping has the effect of moving an entire address space into, or out of, real storage:

- A swapped-in address space is active, having pages in real storage frames and pages in auxiliary storage slots.
- A swapped-out address space is inactive; the address space resides on auxiliary storage and cannot execute until it is swapped in.

z/OS Data Areas and Control Blocks

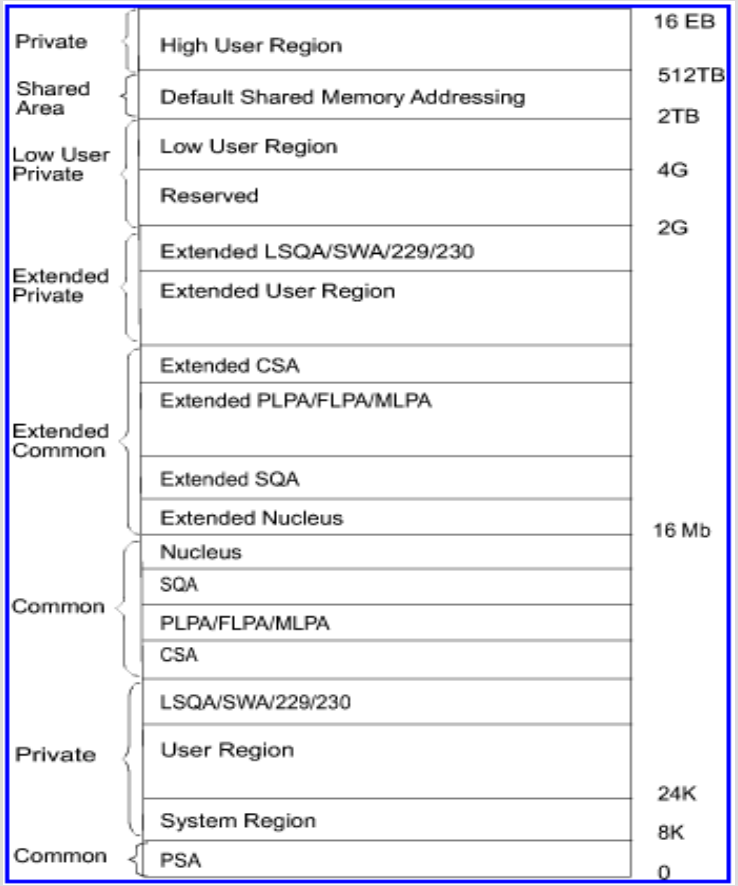
Structures for all 4K pages owned by the hardware, operating system address spaces, middleware address spaces and application address spaces such as TSO and Batch JOB

z/OS Data Areas and Control Blocks

4K pages of
system information

4K page of system
information can reside
in a ‘frame’ or ‘slot’

Some 4K pages of system
information are marked as
a permanent resident in
real storage – ‘frame’ only



Data Areas and Control Blocks

Key-Controlled protection ensures system wide integrity

A storage key is associated with each 4K-byte block of storage that is available in the configuration.

An execution key is associated with each running program

Program Status Word (PSW) Keys

- 0 system or authorized programs, can access all storage
- 1 MVS Scheduler, JES, APPC, TSO/E
- 2 WebSphere
- 5 Data Management O/C/EOV
- 6 VTAM, TCPIP
- 7 IMS, DB2
- 8 Problem Program

Data Areas and Control Blocks

MVS Storage Managers

- Real Storage is managed by RSM
- Virtual Storage is managed by VSM
- Auxiliary Storage is managed by ASM
- Dynamic Address Translation (DAT)
  converts a virtual-to-physical address
- A subpool is an area of virtual storage

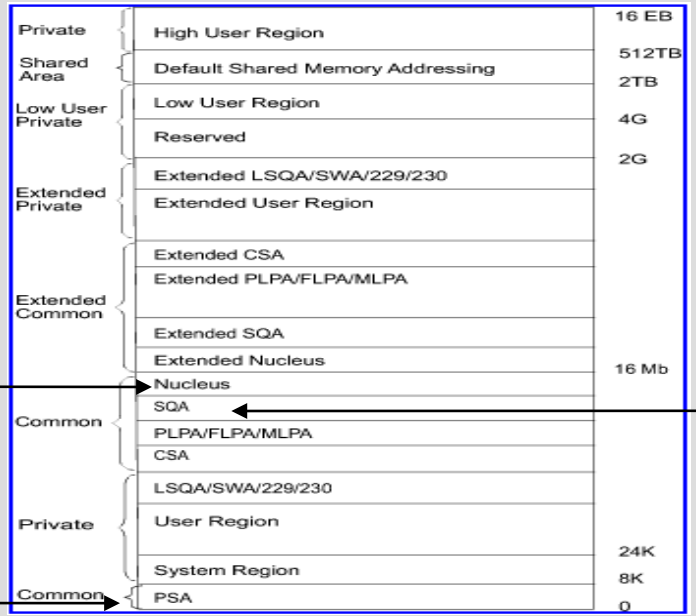
z/OS Data Areas and Control Blocks

PSA>CVT>ASVT>ASCB

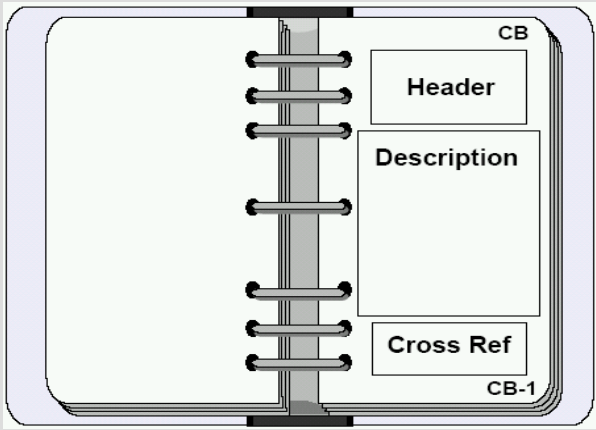
Prefixed **S**ave **A**rea

Address **S**pace **C**ontrol **B**lock

Communications **V**ector **T**able



IBM Manuals



z/OS Data Areas and Control Blocks

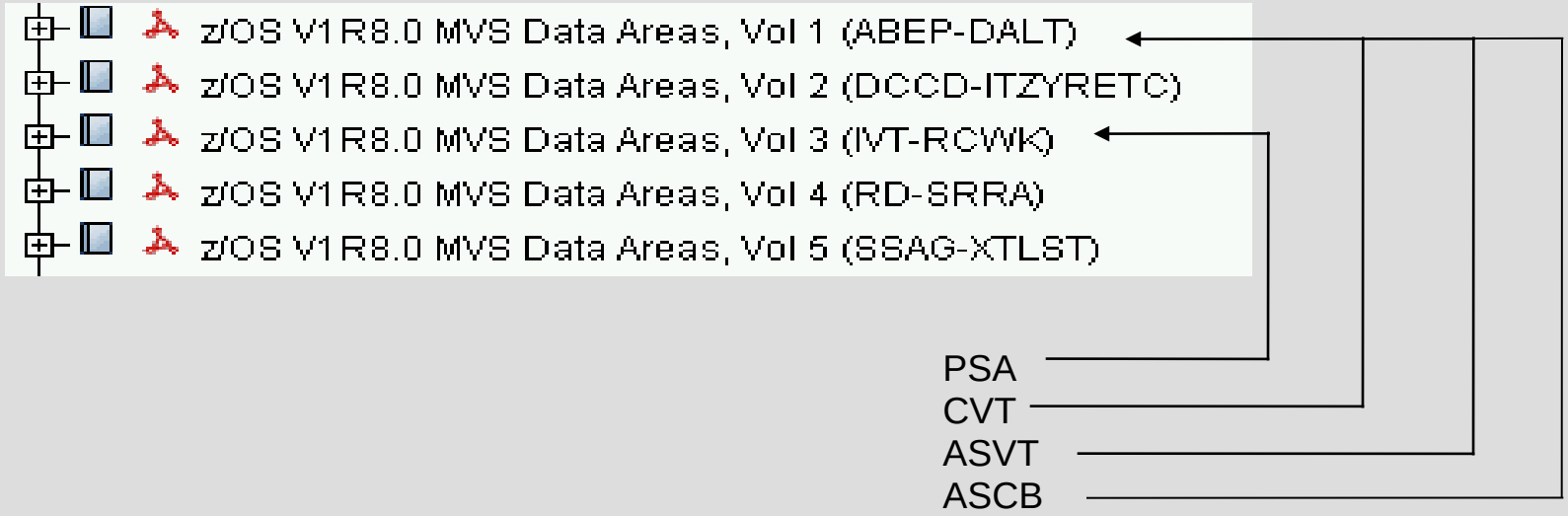
PSA is at real address location 0 << represents a physical CPU

CVT is located at the address stored in x’10’ offset in PSA

ASVT is located at the address stored in x’22C’ offset in the CVT

ASCB list is located at the address stored in x’210’ offset in the ASVT

z/OS Data Areas and Control Blocks



z/OS Control Block
Prefixed Save Area (PSA)

PSA Heading Information

Common Name:	Prefixed Save Area
Macro ID:	IHAPSA
DSECT Name:	PSA
Owning Component:	Supervisor Control (SC1C5)
Eye-Catcher ID:	None
Storage Attributes:	Subpool: 239 Key: 0 Residency: Below 16 MB line
Size:	4096 bytes
Created by:	IEAVFX00 IEAVNIP0 IEEVCPRA
Pointed to by:	The PSA maps the storage that starts at location 0 for the related processor.
Serialization:	Disablement. None needed for FLCFACL.
Function:	Maps fixed hardware and software storage locations for the related processor.

z/OS Control Block – Prefixed Save Area (PSA)

PSA Map

Offsets					
Dec	**Hex**	**Type/Value**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	0	PSA	
0	(0)	X'0'	0	FLC	""
0	(0)	CHARACTER	8	FLCIPPSW (0)	- IPL PSW
0	(0)	BITSTRING	4	FLCRNPSW	-RESTART NEW PSW (AFTER IPL) MDC001
4	(4)	ADDRESS	4		"V(IEAVRSTR)" - SECOND HALF OF RESTART NEW PSW MDC128
4	(4)	X'0'	0	IPLPSW	"FLCIPPSW" --- ALIAS
8	(8)	CHARACTER	8	FLCICCW1 (0)	- IPL CCW1
8	(8)	BITSTRING	8	FLCROPSW	- RESTART OLD PSW (AFTER IPL)
16	(10)	CHARACTER	8	FLCICCW2 (0)	- IPL CCW2
16	(10)	ADDRESS	4	FLCCVT	"V(IEACVT)" - ADDRESS OF CVT (AFTER IPL). THIS OFFSET FIXED BY ARCHITECTURE. (MDC450)
20	(14)	BITSTRING	4		- RESERVED (AFTER IPL) (MDC431)
24	(18)	BITSTRING	8	FLCEOPSW	- EXTERNAL OLD PSW
24	(18)	X'18'	0	EXOPSW	"FLCEOPSW" --- ALIAS
32	(20)	BITSTRING	8	FLCSOPSW	- SVC OLD PSW. THIS OFFSET FIXED BY ARCHITECTURE. (MDC451)
32	(20)	X'20'	0	SVCOPSW	"FLCSOPSW" --- ALIAS
40	(28)	BITSTRING	8	FLCPOPSW	- PROGRAM CHECK OLD PSW
40	(28)	X'28'	0	PIOPSW	"FLCPOPSW" --- ALIAS
48	(30)	BITSTRING	8	FLCMOPSW	- MACHINE CHECK OLD PSW
48	(30)	X'30'	0	MCOPSW	"FLCMOPSW" --- ALIAS
56	(38)	BITSTRING	8	FLCIOPSW	- INPUT/OUTPUT OLD PSW
56	(38)	X'38'	0	IOOPSW	"FLCIOPSW" --- ALIAS
64	(40)	BITSTRING	8		- RESERVED
72	(48)	DBL WORD	8	FLCCVT64 (0)	- 8-byte CVT address
72	(48)	BITSTRING	4		- 1st 4 bytes are 0
76	(4C)	ADDRESS	4	FLCCVT2	"V(IEACVT)" - ADDRESS OF CVT - USED BY DUMP ROUTINES ICB319
80	(50)	BITSTRING	4		- RESERVED
84	(54)	BITSTRING	4		- RESERVED - FLCTRACE DELETED DUE TO SYSTEM TRACE REDESIGN.
88	(58)	BITSTRING	4	FLCENPSW	-EXTERNAL NEW PSW
92	(5C)	ADDRESS	4		"V(IEAQEX00)" - SECOND HALF OF EXTERNAL NEW PSW
92	(5C)	X'58'	0	EXNPSW	"FLCENPSW" --- ALIAS

z/OS Control Block
Communications Vector Table (CVT)

CVT Heading Information

Common Name:	Communications Vector Table
Macro ID:	CVT
DSECT Name:	CVT(when DSECT=YES is coded and PREFIX=YES is not coded)
	CVTFIX(when DSECT=YES and PREFIX=YES is coded)
	CVTMAP(or name user coded in label field of CVT invocation)
	CVTVSTGX(DSECT name of virtual storage extension)
	CVTXTNT1(DSECT name of OS-OS/VS common extension)
	CVTXTNT2(DSECT name of OS/VS1-OS/VS2 common extension)
Owning Component:	Common Macros (SC101)
Eye-Catcher ID:	CVT Offset: 96 Length: 4
Storage Attributes:	Subpool: Nucleus Key: 0 Residency: Below 16M line
Size:	Prefix: 256 bytes CVT: 1280 bytes Virtual storage address extension: 80 bytes OS - OS/VS common extension: 12 bytes OS/VS1 - OS/VS2 common extension: 132 bytes
Created by:	IEAVCVT
Pointed to by:	FLCCVT field of the PSA data area (location X'10') FLCCVT2 field of the PSA data area CVTSMEXT points to the Virtual address storage extension OS/VS - OS/VS extention is pointed to by CVTEXT1 OS/VS1 - OS/VS2 extention is pointed to by CVTEXT2
Serialization:	Based on the individual fields being referenced.
Function:	The CVT provides the means by which non-nucleus-resident routines may refer to information in the nucleus of the control program. It contains addresses of other control blocks and tables used by the control program routines.

z/OS Control Block
Address Space Vector Table (ASVT)

ASVT Heading Information

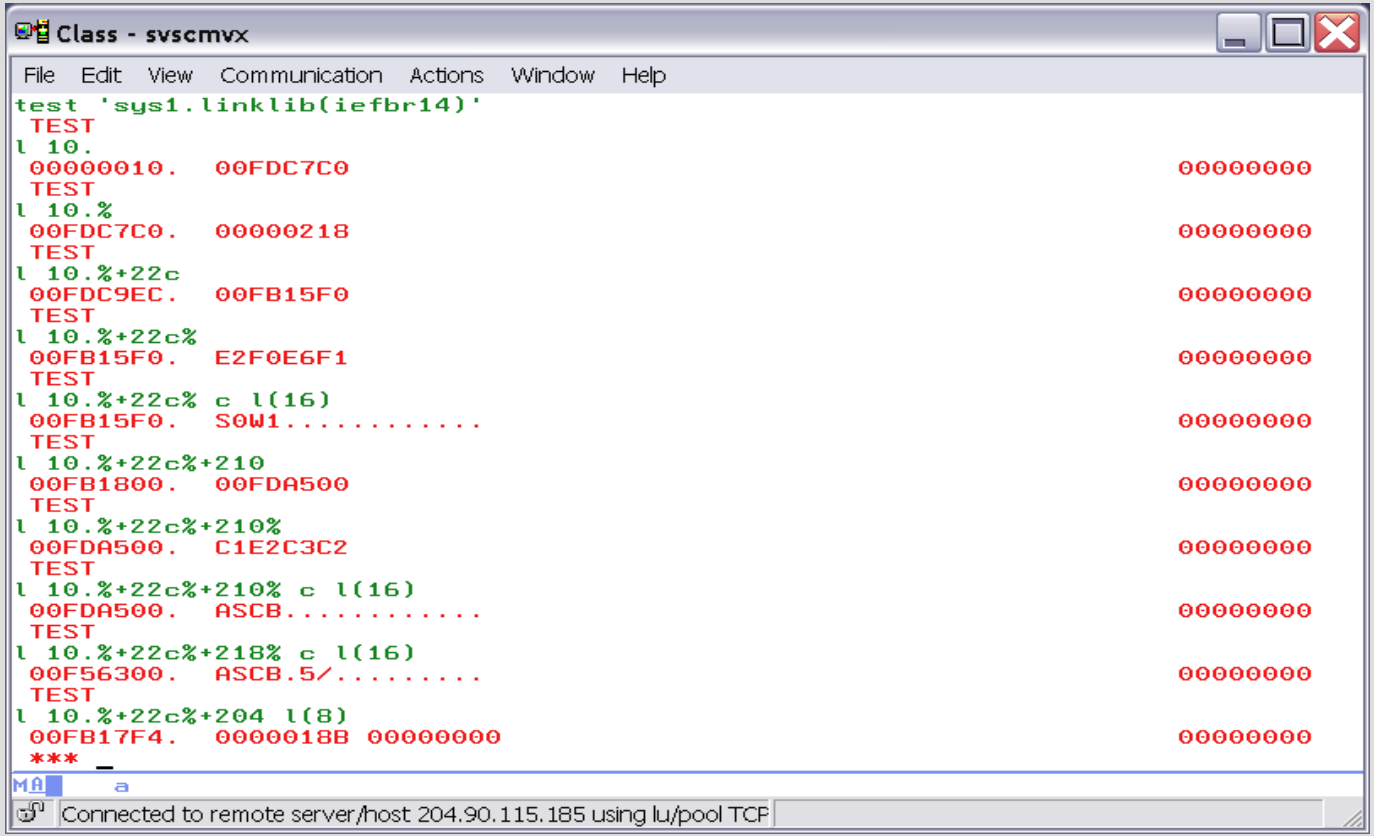
Common Name:	Address Space Vector Table
Macro ID:	IHAASVT
DSECT Name:	ASVT
Owning Component:	Supervisor Control (SC1C5)
Eye-Catcher ID:	ASVTASVT Offset: 512 Length: 4
Storage Attributes:	Subpool: 245 Key: 0 Residency: Below 16M
Size:	Offset of ASVTEND minus offset of ASVTBEGN plus four times the value of ASVTMAXU.
Created by:	IEAVNP09
Pointed to by:	CVTASVT field of the CVT data area
Serialization:	General CMS lock and dispatcher lock
Function:	Mapping for the Address Space Vector Table

z/OS Control Block
Address Space Vector Table (ASVT)

ASVT Map

Offsets					
Dec	**Hex**	**Type/Value**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	0	ASVT	
0	(0)	CHARACTER	472	ASVTPRFX	RESERVED FOR FUTURE EXPANSION
472	(1D8)	DBL WORD	8	ASVTBEGN (0)	- BEGINNING OF ASVT
472	(1D8)	ADDRESS	4	ASVTREUA	ADDRESS OF ASVTREUS BITS
476	(1DC)	ADDRESS	4	ASVTRAVL	ADDRESS OF FIRST AVAILABLE REUSABLE ASID SLOT
480	(1E0)	SIGNED	4	ASVTAAV	NUMBER OF FREE SLOTS ON THE ASVT AVAILABLE QUEUE.
484	(1E4)	SIGNED	4	ASVTAST	NUMBER OF FREE SLOTS ON THE START/SASI QUEUE.
488	(1E8)	SIGNED	4	ASVTANR	NUMBER OF FREE SLOTS ON THE NON-REUSABLE REPLACEMENT QUEUE.
492	(1EC)	SIGNED	4	ASVTSTRT	ORIGINAL SIZE OF START/SASI QUEUE.
496	(1F0)	SIGNED	4	ASVTNONR	ORIGINAL SIZE OF NON-REUSABLE REPLACEMENT QUEUE.
500	(1F4)	SIGNED	4	ASVTMAXI	- ORIGINAL MAX USERS COUNT AS INPUT TO IEAVNP09. OWNERSHIP - SUPERVISOR CONTROL SERIALIZATION - NIP RIM PROCESS
504	(1F8)	BITSTRING	8		- RESERVED. WAS ASVTRSHD/DSHD
512	(200)	CHARACTER	4	ASVTASVT	- ACRONYM IN EBCDIC -ASVT-
516	(204)	SIGNED	4	ASVTMAXU	- MAXIMUM NUMBER OF ADDRESS SPACES
520	(208)	SIGNED	4	ASVTMDSC	- MAXUSER DEFICIT SLOT COUNT. ASVTMDSC = ASVTMAXI - ASVTAAV - NUMBER OF ACTIVE A.S. INCREMENTED WHEN WE TRY TO TAKE A REPLACEMENT SLOT BUT THERE ARENT ANY. DECREMENTED WHEN NON-ZERO AND A NONREUSEABLE ASID BECOMES REUSEABLE AND WE ADD A SLOT TO THE MAXUSER POOL WHEN AN ADDRESS SPACE BECOMES REUSEABLE.
524	(20C)	ADDRESS	4	ASVTFRST	- ADDRESS OF FIRST AVAILABLE ASVT ENTRY (MDC300)
		1...		ASVTAVAI	"X'80'" - BIT ONE IF ASID IS AVAILABLE AND ZERO IF ASID IS ASSIGNED MDC002
528	(210)	ADDRESS	4	ASVTENTY	- ENTRY FOR EACH POSSIBLE ASID. IF ADDRESS SPACE ASSIGNED, ENTRY CONTAINS ADDRESS OF ASCB. IF NOT ASSIGNED, ENTRY CONTAINS EITHER ADDRESS OF NEXT AVAILABLE ASID OR ZEROS WITH HIGH-ORDER BIT ON IF LAST ENTRY. (MDC301) IF THE ADDRESS SPACE IS MARKED NON-REUSABLE, THE ENTRY CONTAINS THE ADDRESS OF MASTER'S ASVT ENTRY WITH THE HIGH BIT ON.

z/OS Control Block
PSA > CVT > ASVT > ASCB



z/OS Control Block
Address Space Control Block (ASCB)

ASCB Heading Information

Common Name:	ADDRESS SPACE CONTROL BLOCK
Macro ID:	IHAASCB
DSECT Name:	ASCB
Owning Component:	SUPERVISOR CONTROL (SC1C5)
Eye-Catcher ID:	ASCB Offset: 0 Length: 4
Storage Attributes:	Subpool: 245 Key: 0 Residency: Below 16M
Size:	384 bytes
Created by:	IEAMSWCB, IEAVEMRQ
Pointed to by:	CVTASCBH and CVTASCBL fields of the CVT data area PSAANEW field of the PSA data area PSAAOLD field of the PSA data area (Master's ASCB) ASVTENTY field of the ASVT data area ASCBFWDP, ASCBBWDP and ASCBTRQP fields of the ASCB data area ASMASCBP field of the ASMVT data area JSELASCB field of the JSEL data area LCTASCBA field of the LCT data area LDAASCB field of the LDA data area LWAPASCB field of the LWA data area PCBASCB field of the PCB data area RSMASCB field of the RSMHD data area SMCAASCB field of the SMCA data area SRBASCB field of the SRB data area SSENASCB and SSETASCB fields of the SSOB data area TCASASCB field of the TCAST data area TQEASCB field of the TQE data area TSBASCBA field of the TSB data area TVCSASCB field of the TVCS data area TWAASCB field of the TWAR data area UCMASCB field of the UCM data area OUCBASCB field of the OUCB data area WEBHASCB field of the WEB data area WEBLSQP field of the WEB data area
Serialization:	Serialization of the ASCB is dependent on the field being referenced. Some serialization techniques used here are local lock, compare and swap (CS), compare double and swap, and global intersect.
Function:	Contain information and pointers needed for Address Space Control. The ASCB is non-swappable.

z/OS Control Block
Address Space Control Block (ASCB)

ASCB Map

Offsets					
Dec	Hex	Type/Value	Len	Name (Dim)	Description
0	(0)	STRUCTURE	0	ASCB	
0	(0)	DBL WORD	8	ASCBEGIN (0)	- BEGINNING OF ASCB
0	(0)	CHARACTER	4	ASCBASCB	- ACRONYM IN EBCDIC -ASCB-
4	(4)	ADDRESS	4	ASCBFWDP	- ADDRESS OF NEXT ASCB ON ASCB READY QUEUE
8	(8)	ADDRESS	4	ASCBBWDP	- ADDRESS OF PREVIOUS ASCB ON ASCB READY QUEUE

z/OS Control Block
Address Space Control Block (ASCB)

Offsets					
Dec	Hex	Type/Value	Len	Name (Dim)	Description
12	(C)	ADDRESS	4	ASCBLTCS	- TCB and preemptable-class SRB Local lock suspend service queue. Serialization: ASCB CML promotion WEB lock.
16	(10)	DBL WORD	8	ASCBSUPC (0)	- SUPERVISOR CELL FIELD (MDC309)
16	(10)	ADDRESS	4	ASCBSVRB	- SVRB POOL ADDRESS. THIS OFFSET FIXED BY ARCHITECTURE. (MDC310)
20	(14)	SIGNED	4	ASCBSYNC	- COUNT USED TO SYNCHRONIZE SVRB POOL. THIS OFFSET FIXED BY ARCHITECTURE. (MDC311)
24	(18)	ADDRESS	4	ASCBIOSP	- POINTER TO IOS PURGE INTERFACE CONTROL BLOCK (IPIB) (MDC308)
28	(1C)	BITSTRING	4	ASCBWQLK (0)	WEB QUEUE LOCK WORD SERIALIZATION: COMPARE AND SWAP OWNERSHIP: SUPERVISOR CONTROL
28	(1C)	BITSTRING	2	ASCBR01C	RESERVED, MUST BE ZERO
30	(1E)	SIGNED	2	ASCBWQID	LOGICAL CPU ID OF THE PROCESSOR HOLDING THE WEB QUEUE LOCK OWNERSHIP: SUPERVISOR CONTROL
32	(20)	ADDRESS	4	ASCBR020 (0)	Reserved as of z/OS 1.11
32	(20)	ADDRESS	4	ASCBSAWQ_PREZOS11	- ADDRESS OF ADDRESS SPACE SRB WEB QUEUE SERIALIZATION: WEB QUEUE LOCK OWNERSHIP: SUPERVISOR CONTROL Not set as of z/OS 1.11
		1...		ASCBURRQ_PREZOS11	"X'80'" - SYSEVENT USER READY REQUIRED SERIALIZATION: WEB QUEUE LOCK OWNERSHIP: SUPERVISOR CONTROL Not set as of z/OS 1.11
36	(24)	SIGNED	2	ASCBASN (0)	- SAME AS ASCBASID
36	(24)	SIGNED	2	ASCBASID	- ADDRESS SPACE IDENTIFIER FOR THE ASCB
38	(26)	BITSTRING	1	ASCBR026	- RESERVED
39	(27)	BITSTRING	1	ASCBSRMFLAGS	- SRM flags Ownership: SRM Serialization: SRMLOCK
		1...		ASCBVCMOVERRIDE	"X'80'" - This bit indicates that this address space should not follow the standard SRM management in an VCM=on environment. Instead of trying to assign the work this address space to the same affinity node for cache efficiency concerns, assign this work to any affinity node, ignore any cache concerns. Ownership: SRM
		.1..		ASCBBROKENUP	"X'40'" - This bit indicates that this address space has been broken up by SRM. Ownership: SRM
		..1.		ASCBVCMGIVEPREEMPTION	"X'20'" - This bit indicates that this address space should get full preemption. Ownership: SRM
		...1		ASCBVCMGIVESIGPANY	"X'10'" - This bit indicates that this address space can SIGP any waiting CPUs to process its work. Ownership: SRM
40	(28)	BITSTRING	1	ASCBLL5	- FLAGS, SERIALIZATION - LOCAL LOCK
		..1.		ASCBS33	"X'20'" - STAGE II EXIT EFECTOR HAS SCHEDULED AN RQE OR IQE AND STAGE III EXIT EFFECTOR SHOULD BE INVOKED

Summary

- Each Batch Job, TSO and Started Task are a separate address space

- Types of Address Spaces
  1. Batch Jobs – identified by JOBID JOB#####
  2. Started Tasks – identified by JOBID STC#####
  3. Time Sharing Tasks – identified by JOBID TSU#####

  where ##### is a uniquely assigned number

- Address Space is a contiguous range of virtual addresses divided into blocks of 4K pages. The pages are stored in both real and auxiliary storage. Paging is the movement of pages between real and auxiliary storage.

Summary

- A PAGE is a 4K area of processing storage
- A PAGE is also considered to be a 4K block of virtual addresses
- All processing storage is contained in PAGES
- A 4K FRAME of central storage can hold a PAGE
- A 4K SLOT of auxiliary storage can hold a PAGE
- Movement of a PAGE between a FRAME and a SLOT is called PAGING

Summary

- All programs operate with an assigned protection key
- All discrete storage areas have an assigned protection key
- System integrity is maintained through a requirement for program and storage area keys to match
- Program operating with key 0 can access any discrete storage areas regardless of the assigned storage area protect key

Summary

- Virtual storage is managed by Virtual Storage Manager (VSM)
- Real storage is managed by Real Storage Manager (RSM)
- Auxiliary storage is managed by Auxiliary Storage Manager (ASM)
- System workload is managed and prioritized by Work Load Manager (WLM)

Professional Manuals and Documentation

Unit summary

Having completed this unit, you should be able to:

- Describe an address space
- Describe virtual storage
- Describe paging
- List 3 types of address spaces
- List 3 types of memory storage
- Describe system integrity using key-controlled protection