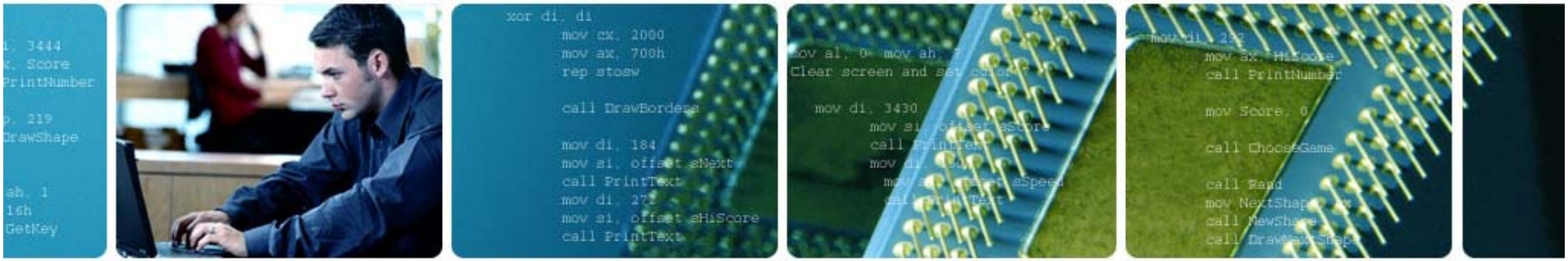
Introduction to the new mainframe

Chapter 3: z/OS Overview



Chapter 3 objectives

Be able to:

- **Give examples of how z/OS differs from a single-user operating system.**
- **List the major types of storage used by z/OS.**
- **Explain the concept of virtual storage and its use in z/OS.**
- **State the relationship between pages, frames, and slots.**
- **List several defining characteristics of the z/OS operating system.**
- **List several software products used with z/OS to provide a complete system.**
- **Describe several differences and similarities between the z/OS and UNIX operating systems.**

Key terms in this chapter

- address space
- addressability
- auxiliary storage
- dynamic address translation (DAT)
- frame
- input/output (I/O)
- middleware
- multiprocessing
- multiprogramming
- page / paging
- page data set
- program product
- real storage
- slot
- swap data set
- UNIX
- virtual storage
- z/OS

What is z/OS?

The most widely used mainframe operating system

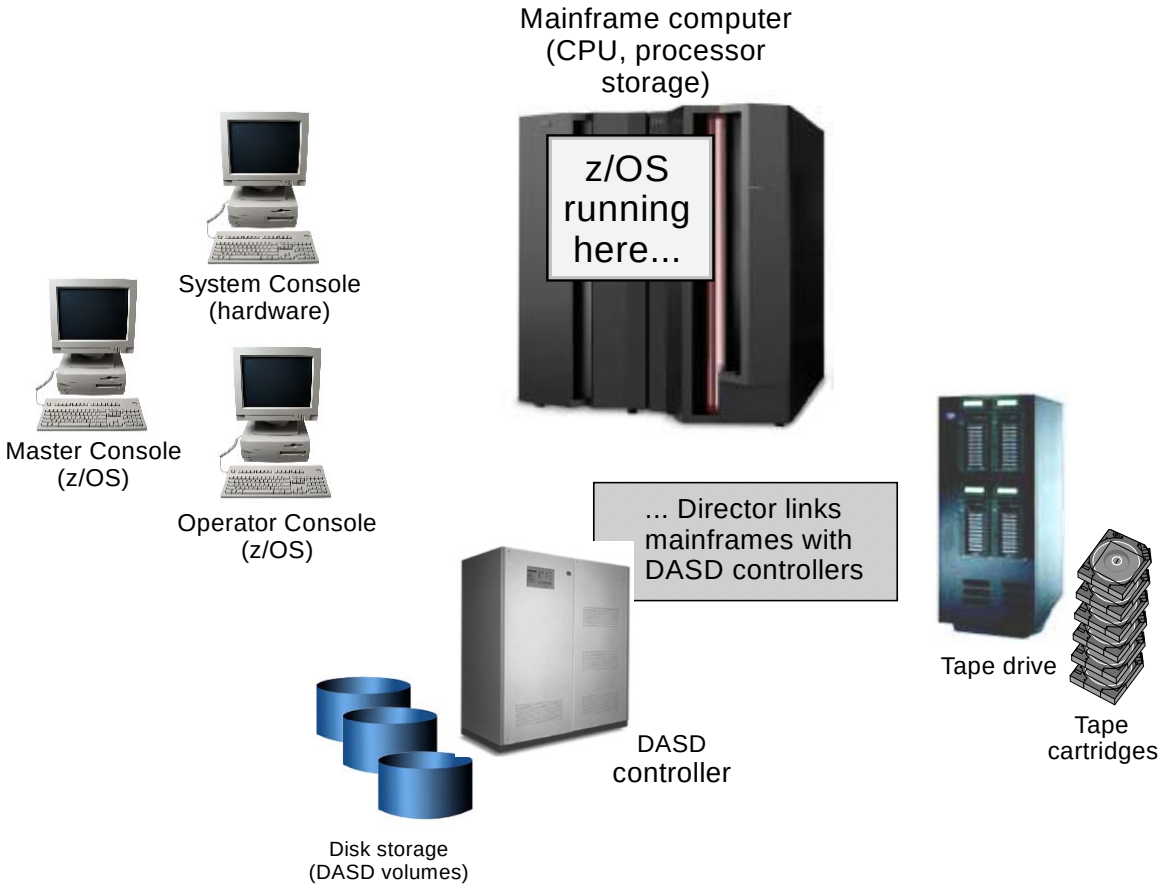
64-bit operating system

Ideally suited for processing large workloads for many concurrent users

Designed for:

- **Serving 1000s of users concurrently**
- **I/O intensive computing**
- **Processing very large workloads**
- **Running mission critical applications securely**

Hardware resources managed by z/OS



Overview of z/OS internals

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

Use of the program status word (PSW)

Techniques of multiprogramming and multiprocessing

Information about the system, resources, and tasks is contained in control blocks

Management of physical storage:

- **Real storage**
- **Auxiliary storage**
- **Virtual storage**

Virtual storage concepts

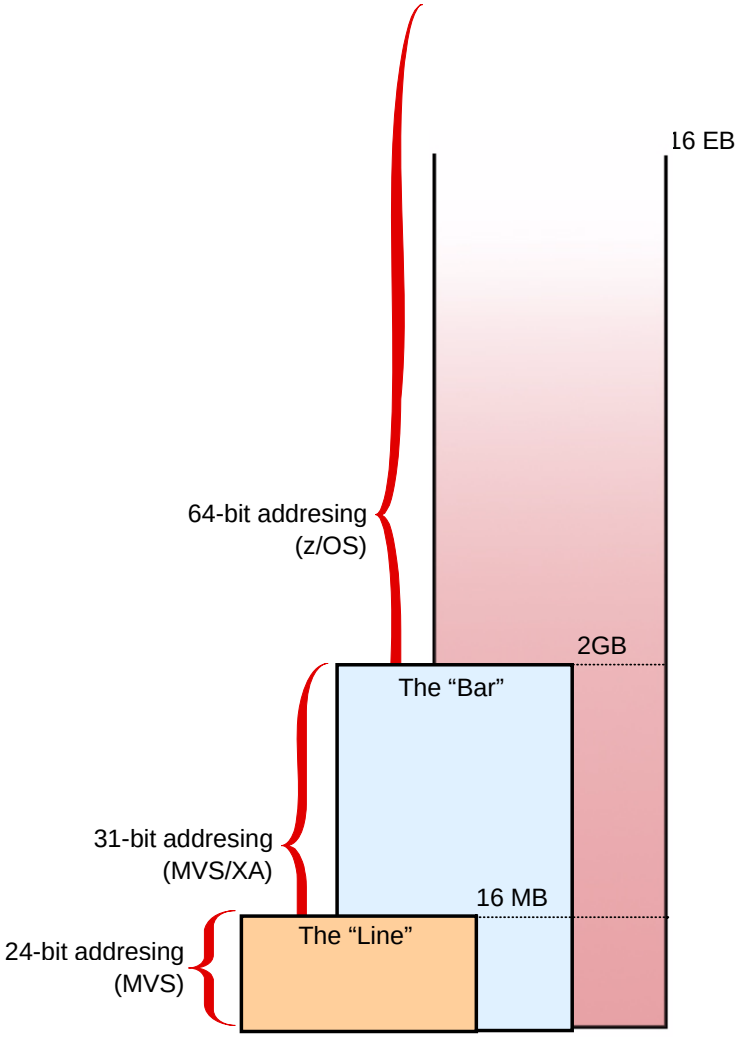
Virtual storage **is an "illusion" created through z/OS management of real storage and auxiliary storage through tables.**

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

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)

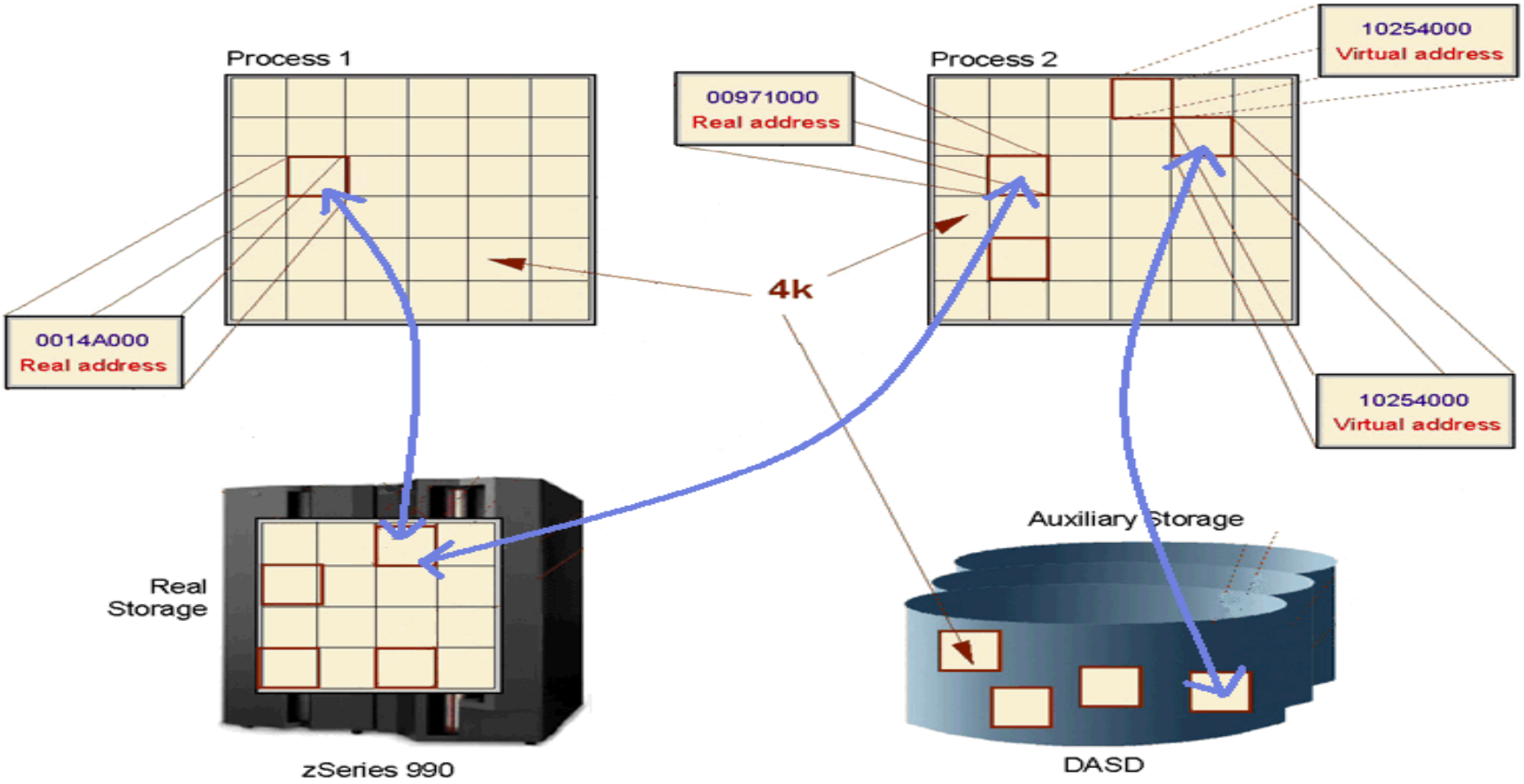
The address space concept



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 a page of information

How virtual storage works (continued…)



Pages, Frames, and Slots

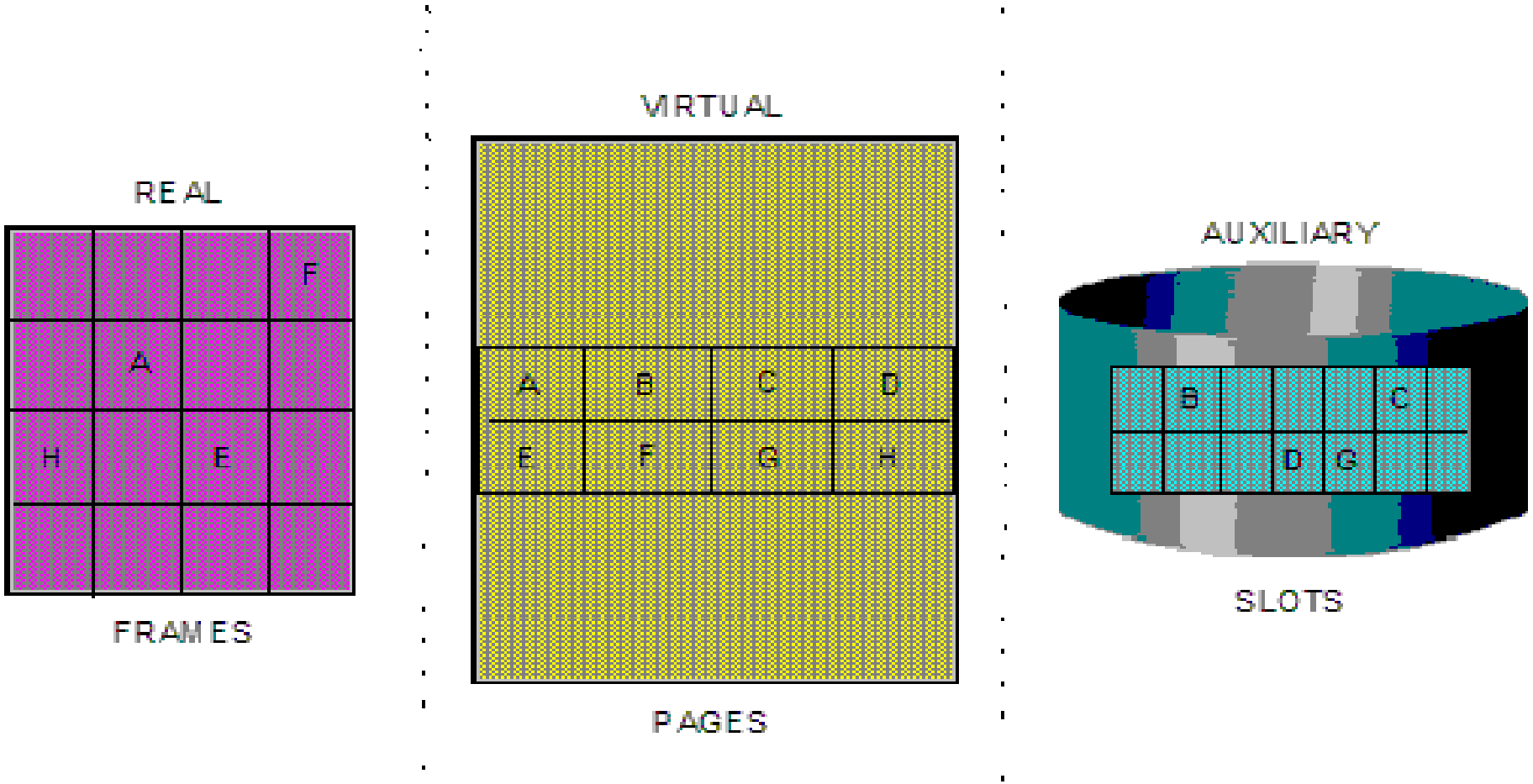
The pieces of a program executing in virtual storage must be moved between real and auxiliary storage:

- **A block of real storage is a *frame*.**
- **A block of virtual storage is a *page*.**
- **A block of auxiliary storage is a *slot*.**

A page, a frame, and a slot are all the same size: 4096 bytes (4 kilobytes).

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

Pages, Frames, and Slots (continued)



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.**

Brief history of z/OS addressability

1970 - System/370 defined storage addresses as 24 bits in length.

1983 - System/370-XA extended the addressability of the architecture to 31 bits.

2000 - z/Architecture extended the addressability to 64 bits.

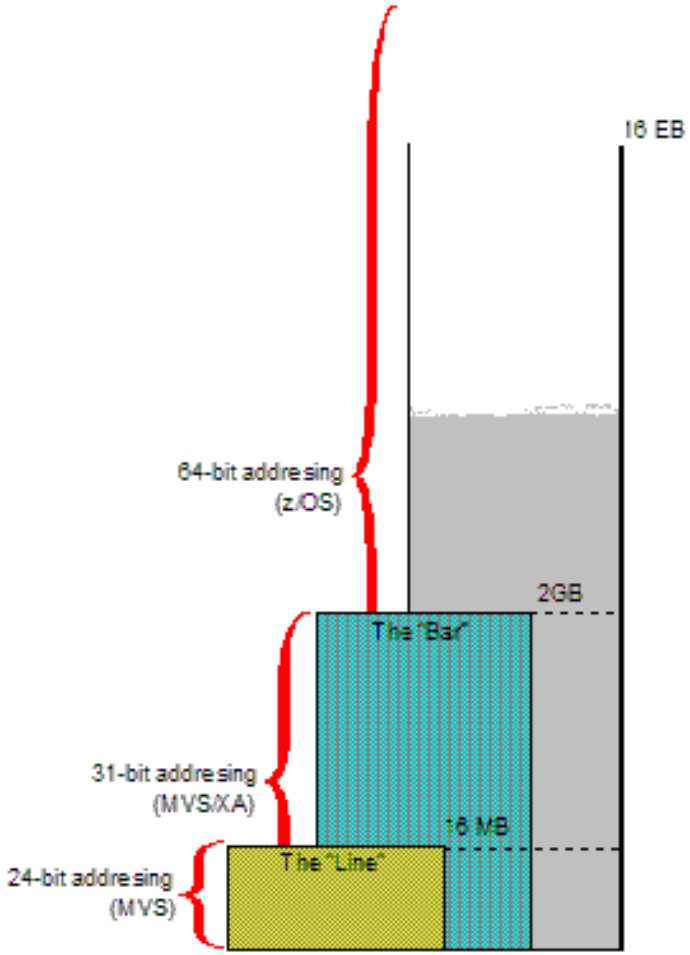
Addressing value of a byte

Bits	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	
Bit count	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	
Bit value	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**	**(binary, power of 2)**

In summary:

- **One byte can address 256 locations**
- **Three bytes can address 16 million locations**

Brief history of z/OS addressability (continued…)

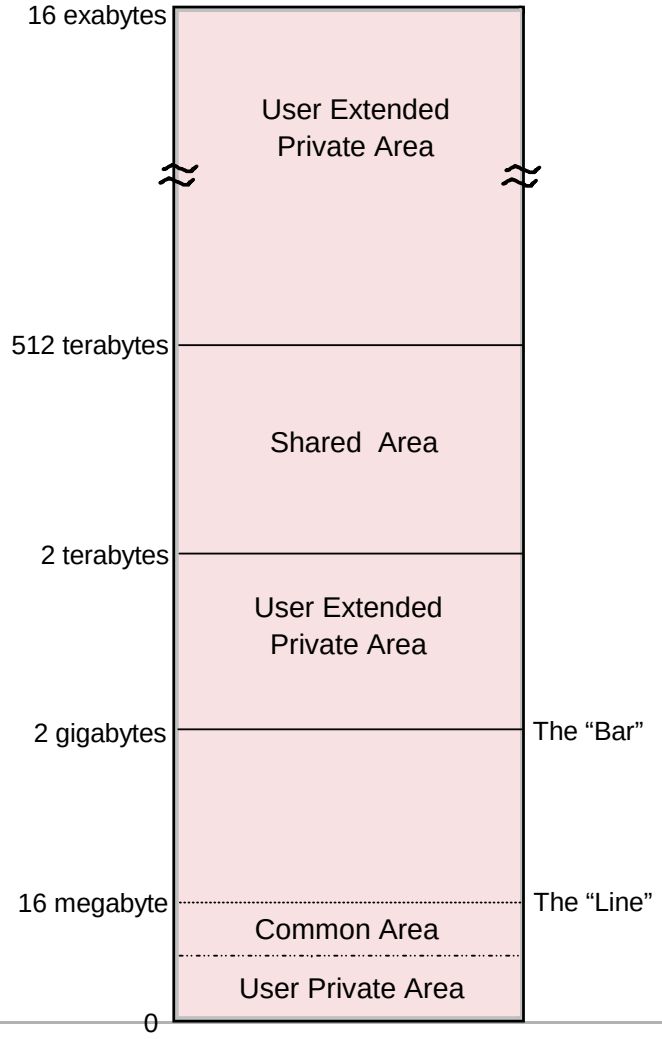


What's in an address space?

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

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

64-bit address space map



z/OS 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.**
- **Subsystem address spaces for major system functions and middleware products such as DB2, CICS, and IMS.**
- **TSO/E address spaces are created for every user who logs on to z/OS**
- **Address spaces for every batch job that runs on z/OS.**

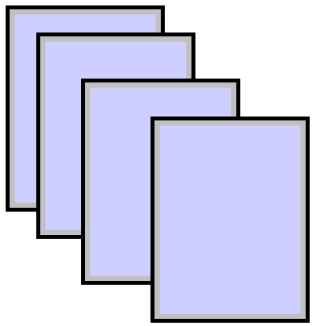
How is peripheral storage managed?

- Management of peripheral storage devices involves file allocation, placement, monitoring, migration, backup, recall, recovery, and deletion.
- A typical z/OS production system includes both manual and automated processes for managing storage.
- A user or program can directly control many aspects of z/OS storage use.
- The primary means of managing storage in z/OS is through DFSMS.

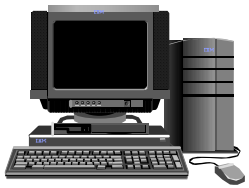
Summary of z/OS facilities

- Address spaces and virtual storage for users and programs.
- Physical storage types available: real and auxiliary.
- Movement of programs and data between real storage and auxiliary storage through paging.
- Dispatching work for execution, based on priority and ability to execute.
- An extensive set of facilities for managing files stored on disk or tape. Operators use consoles to start and stop z/OS, enter commands, and manage the operating system.

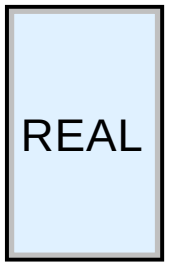
Summary of z/OS facilities (continued…)



Address spaces

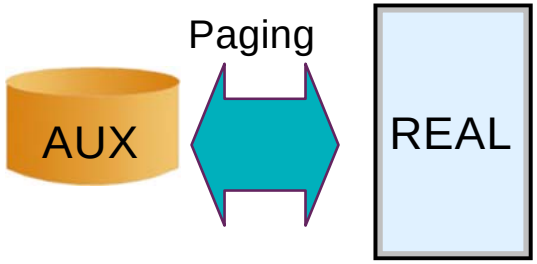


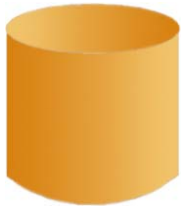
Operator communication



Physical storage

Reliability, availability, and serviceability





Data integrity

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.

Program products for z/OS

A z/OS system usually contains additional program products (priced software) that are needed to create a practical working system:

- **security manager**
- **database manager**
- **compilers**
- **utility programs**
- **vendor products**

Middleware for z/OS

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**

A brief comparison of z/OS and UNIX…

Quite a few concepts are common to both:

- Boot the system versus IPL the system
- Files versus data sets
- Editors → vi, ed, sed, and emacs (UNIX) versus ISPF (z/OS)
- telnet or rlogin (UNIX) versus TSO logon (z/OS)

Summary

- z/OS, the most widely used mainframe operating system, is ideally suited for processing large workloads for many concurrent users.
- Virtual storage is an illusion created by the architecture, in that the system seems to have more storage than it really has.
- Each user of z/OS gets an address space containing the same range of storage addresses.
- z/OS is structured around address spaces, which are ranges of addresses in virtual storage.
- Production systems usually include add-on products for middleware and other functions.