- An address space, literally defined as the range of addresses available to a computer program, is like a programmer's map of the virtual storage available for code & data.

- Various techniques, like storage protect key and supervisor state requirements, provide protection that is almost like a wall around an address space, and this wall is basically a good thing from the point of view of the work going on inside that individual address space.

- Any data that the requesting task shares must be placed in common areas of the address space.

- Cross memory provides the mechanism that stores data outside the memory of an address space. This data is a fancy way to describe scheduling an SRB which is a service request block that a task can schedule to request that some service take place in the same address space or another address space.

- SRB is a service request block, the point where the SRB completes cannot be predicted.

- The system memory cross memory communication can be either a wall around an address space, and this wall is basically a good thing from the point of view of the programming problems.

- Data spaces and hiperspaces are data-only spaces that can hold up to 2 gigabytes of data and provide isolation for the data they contain in 4KB blocks.

- The virtual lookspace facility (VLF) is a set of z/OS services providing a high-performance alternate path method of retrieving named objects from DASD on behalf of a user program or a data-in-virtual object.

- Virtual I/O (VIO) provides an alternative to the virtual lookaside facility (VLF) and is less expensive than VLF.

- VIO requires no modification to existing programs to use i/o operations.

- Requires no modification to existing programs that use an access method with EXCP. Either use storage management subsystem (SMS) to make global request to use VIO or use JCL for an individual program.

- Programs must change to use system macros or data window services. LHL programs can NOT use hiperspaces directly. They can use hiperspaces through data window services.

- Programs can use data spaces and hiperspaces to: 1. Isolate data from other tasks in the address space. Data spaces and hiperspaces do not include system areas, the cost of creating and deleting them is less than that of an address space. Because data spaces and hiperspaces do not include system areas, the cost of creating and deleting them is less than that of an address space. 2. Isolate data from other tasks in the address space. Data spaces and hiperspaces do not include system areas, the cost of creating and deleting them is less than that of an address space. 3. Isolate data to improve performance and availability. Data spaces and hiperspaces do not include system areas, the cost of creating and deleting them is less than that of an address space.

- Programs can use data spaces and hiperspaces to: 1. Isolate data from other tasks in the address space. Data spaces and hiperspaces do not include system areas, the cost of creating and deleting them is less than that of an address space. 2. Isolate data from other tasks in the address space. Data spaces and hiperspaces do not include system areas, the cost of creating and deleting them is less than that of an address space. 3. Isolate data to improve performance and availability. Data spaces and hiperspaces do not include system areas, the cost of creating and deleting them is less than that of an address space.

- Use VIO when you want to improve the performance of an existing program, but you do not want to make large changes. (template to set up VIO)

- For more information about how to use VIO, see z/OS MVS JCL.

-hour time, extended addressability can mean learning new programming techniques, or new ways of applying existing techniques.

- Asynchronous cross memory communication is a fancy way to describe scheduling an SRB which is a service request block that a task can schedule to request that some service take place in the same address space or another address space.

- Any data that the requesting task shares must be placed in common areas of the address space.

- Access Registered Address Space Control (AR ASC) mode, a program can move, compare, or perform operations on data in other address spaces or in data spaces.
Because of changes in the architecture that supports the Multiple Virtual Storage (MVS) operating system, there have been changes in the addresses to the 64-bit address space:
- The address space of the 1970s began at address 0 and ended at 16 megabytes. This address space provided 2-gigabyte addresses.

In the early 1980s, XA (extended architecture) introduced an address space that began at address 0 and ended at 2 gigabytes. This address space provided 31-bit addresses.
- To maintain program compatibility, MVS provided two addressing modes (AMODEs):
  - Programs that run in AMODE 24 can use only the first 16 megabytes of the address space.
  - Programs that run in AMODE 31 can use the entire 2 gigabytes.

As of z/OS release 1.2, the address space begins at address 0 and ends at 16 exabytes, an incomprehensibly high address and the architecture that creates this address space provides 64-bit addresses.

- The special area below the 2-gigabyte address has not changed; all programs in AMODE 24 and AMODE 31 continue to run without change.

- In some fundamental ways, the address space is much the same as the XA address space.
  - It includes a virtual line at the 16-megabyte address.
  - The 64-bit address space also includes the virtual line at the 16-megabyte address; in addition, it includes a second virtual line called the bar that marks the 2-gigabyte address.

- The bar separates storage below the 2-gigabyte address, called below the bar, from storage above the 2-gigabyte address, called above the bar.

- The area above the bar is intended for data; no programs run above the bar.

- There is no area above the bar that is common to all address spaces (at this time), and no system control blocks exist over all address spaces.

**NOTE:** IBM reserves an area of storage above the bar for special uses to be developed in the future. You can set a limit on how much virtual storage above the bar each address space can use. This limit is called the what applications are granted access to large pages. The key factors to consider when granting access to large pages:
- 1. Available large frame area where long-running memory-intensive applications benefit most from using large pages.
- 2. Assigned to a subspace
- 3. Not eligible to be assigned to a subspace.

- **MEMLIMIT Decision Tree**

The storage key is defined by the program; for an unauthorized program, the storage key at the time of issuing the MEMLIMIT command is the program's PSW key.

 You can specify whether you want the memory object to be fetched protected or not. There is no change key support for virtual storage above the bar.

- The owner of a private memory object is the TCB of the program that creates the private memory object, or a TCB with which the creating program assigns ownership. If an SRB creates a private memory object, the SRB must assign ownership of the private memory object to a task.

- A memory object is system-owned. The cross-memory resource owner (CMRO) TCB of the address space owns the shared interest in the shared memory object.

When a program creates a memory object, it provides an area in which the system returns the memory object's low address.

You can think of the address as the name of the memory object. After creating the memory object, the program can use the storage in the memory object as it used storage in the 2-gigabyte address space.

- **Programs obtain storage above the bar in “chunks” of virtual storage called memory objects.**
  - The system allocates a memory object as a number of virtual segments; each segment is a megabyte in size and begins on a megabyte boundary.
  - A memory object can be as large as the memory limits set by your installation and as small as one megabyte.

- **Other attributes of a memory object include the following characteristics:**
  1. The storage key is defined by the program; for an unauthorized program, the storage key at the time of issuing the MEMLIMIT command is the program's PSW key.
  2. You can specify whether you want the memory object to be fetched protected or not. There is no change key support for virtual storage above the bar.
  3. The owner of a private memory object is the TCB of the program that creates the private memory object, or a TCB with which the creating program assigns ownership. If an SRB creates a private memory object, the SRB must assign ownership of the private memory object to a task.

**Why would you use virtual storage above the bar?**

- The program needs more virtual storage than the first 2-gigabyte address space provides.
- Before z/OS v1.3, all programs in AMODE 31 or AMODE 24 were unable to use data above the bar.
- To use virtual storage above the bar, a program must request storage above the bar, be in AMODE 64, and the new zArchitecture assembler instruction.
- As of z/OS v1.5, the following enhancements for 64-bit virtual storage have been added:
  1. 64-bit shared memory support
  2. Multiple guard area support for private high virtual storage
  3. Default shared memory addressing area between 2 terabytes and 512 terabytes

**Why would you use virtual storage above the BAR in the 64-bit address space?**

- The program needs more virtual storage than the first 2-gigabyte address space provides.
- Before z/OS 1.2, a program’s need for storage beyond what the former 2-gigabyte address space provided was sometimes met by creating one or more data spaces or hyperspaces and designing a memory management schema to keep track of the physical memory associated with these spaces. Sometimes programs wrote to zero and used complex algorithms to manage storage, reallocate and reuse areas, and check storage availability.
- With the 16-exabyte address space, these kinds of programming complexities are unnecessary because a program can potentially use as much virtual storage as it needs, while containing the data within the program’s home address space.

**Memory management above the 2 GB bar**

- A memory object is a contiguous range of virtual addresses that are allocated by programs as a number of application pages or page sets.
- Memory management above the 2 GB bar is based on the assumption that a memory object is a contiguous range of addresses...