International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677● Website: www.ijsat.org● Email: editor@ijsat.org

Technical Insights into the Implementation of z/OS Memory and Address Spaces

Chandra Mouli Yalamanchili

chandu85@gmail.com

Abstract

z/OS is an advanced operating system designed for IBM mainframes that excel in enterprise-level scalability and reliability. IBM's memory management is one of the vital factors that make IBM z/OS 's massive processing possible.

This paper explores different concepts regarding different types of memory and how IBM mainframes use memory and storage to provide massive processing scale.

This paper also explores a coding example to illustrate the memory allocation and de-allocation from program perspective.

Keywords: IBM Mainframe; z/OS; Mainframe memory management; z/OS memory paging; z/OS storage; z/OS address spaces.

Introduction

z/OS is the operating system developed by IBM for mainframes to support enterprise-level computing that supports unparalleled scalability and performance. The z/OS is capable of multiprogramming and multiprocessing, as explained below, allowing it to support multiple users running multiple programs simultaneously. [1]

- Multiprogramming is the capability to execute many programs concurrently. [1]
- Multiprocessing is the capability to execute the operation of two or more processors while sharing various resources. [1]

These multiprogramming and multiprocessing capabilities need a large amount of memory, leading to complex memory management requirements. [1]

This paper explores different types of memory and different memory management features like virtualization, paging, and other components that come into play as part of memory management.

For the purposes of this paper, memory refers to the storage utilized by the processor as part of task execution, encompassing different control blocks, working storage, and data fetched as part of I/O for processing, analogous to RAM in traditional PC architectures.

Types of memory

Like other computers, mainframes use two types of physical storage: real and auxiliary. [1]

- Real or central storage - This is the physical memory on the mainframe processor. Access to this storage is much faster and synchronous as it's closely integrated with the processor.[1]
- Auxiliary or paging storage - This is the physical storage external to the mainframe processor, such as disk drives and tape drives. Access to this storage is slower compared to real storage as the processor must go through an I/O request to access this storage.[1]
- Virtual storage - The mainframe combines real and auxiliary storage to form virtual storage. Instead of physical storage, each user or process can only access virtual storage. [1]

Through virtual storage, z/OS can support many users simultaneously by creating an illusion of much bigger memory availability, while the mainframe might have much less real storage installed. z/OS achieves this by keeping only the active portions of each program in real storage to execute it quickly, and it keeps the rest of the program and data in page data sets that reside in auxiliary storage. z/OS relates the auxiliary storage to central storage by using a series of tables and indexes. [1]

Virtual storage concepts

Address space - The range of contiguous virtual addresses assigned by the operating system to a user or a process is called an address space. z/OS address space is analogous to the process in UNIX. The user can start multiple tasks within address space using task control blocks, like UNIX threads.[1]

Address spaces provide isolation for data and errors. Private areas in one address space are not accessible to other address spaces. Similarly, errors in one address space won't impact other address spaces. This isolation of address spaces helps bring more stability to the mainframe ecosystem.[1]

Inter-address space communication can happen through two different options, as mentioned below:

1. SRB (Service Request Block) - Asynchronous process to initiate a process in another address space.[1]
2. Cross-memory (XM) services and access registers - Synchronous process to access another's address spaces directly. The issuing program must have special authority controlled by the authorized program facility (APF) to use this service.[1]

Dynamic Address Translation (DAT) - DAT is the process that translates virtual address during storage reference into real address. Different storage faults (type, region, segment, page) can occur while looking for real storage using translation look-aside buffers, leading to z/OS bringing the respective page from auxiliary storage. DAT uses hardware and software to manage the programs efficiently and other read-only data across different address spaces to avoid duplicate copies of data. [1]

The picture below depicts the DAT in action:

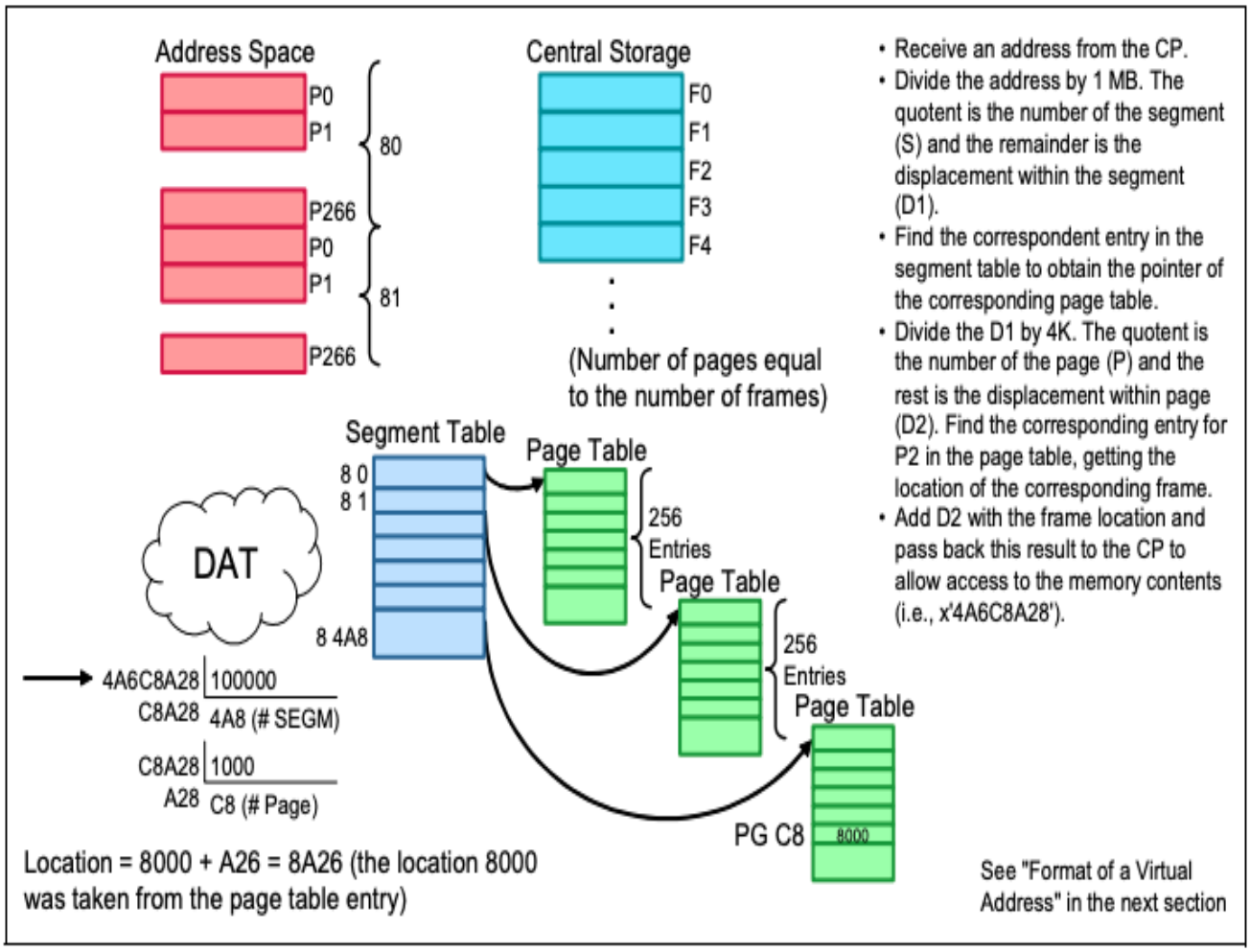

Figure 1: Depiction of how DAT process resolves virtual address to central storage address. [1]

How virtual storage addressing works

To manage central, auxiliary, and virtual storage, z/OS uses different storage building blocks to track which data is available and where and how to access it.

- Frame - Equally divided central/real storage block with a unique address.[1]
- Slot - Equally divided auxiliary storage block with a unique address.[1]
- Page - z/OS divides the program into pieces as frames or slots; these program pieces are called pages. [1]

The example below shows how z/OS builds virtual storage by performing a paging operation to pull the program fragments from real and auxiliary storage.

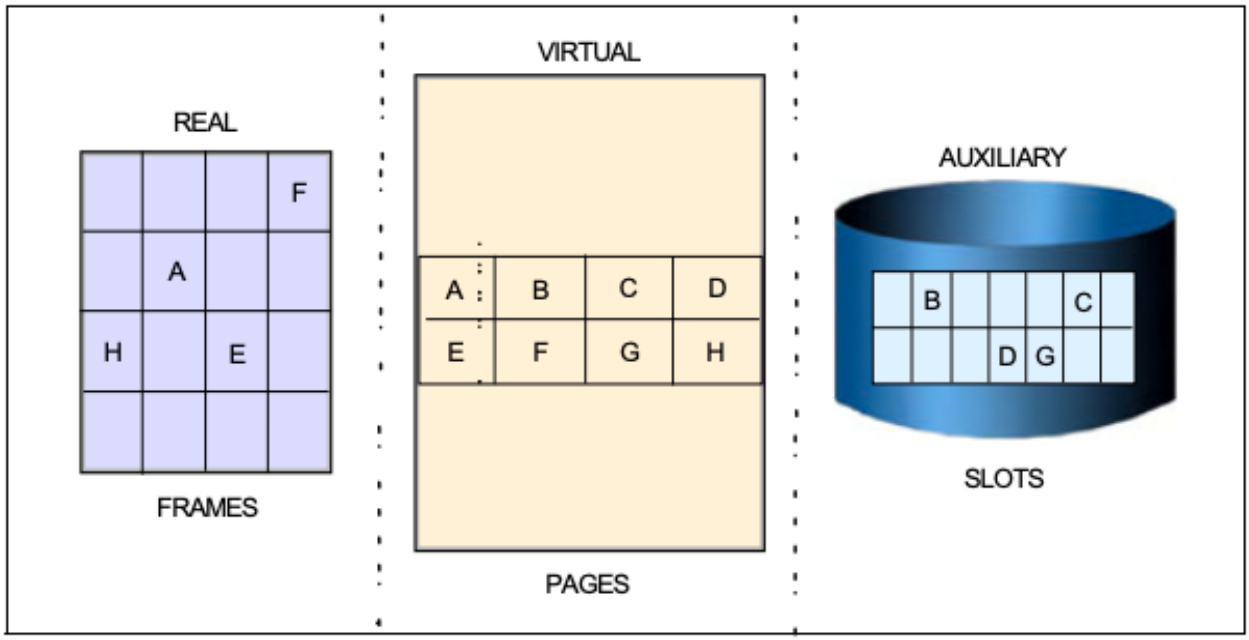

Figure 2: Depiction of how different fragments of a program are pulled into virtual storage from real and auxiliary storages for execution. [1]

Paging

As part of program execution, z/OS moves the required pages into central storage or out to auxiliary storage as needed. This movement of pages between auxiliary storage slots and central storage frames is called paging. Below are different types of paging activities that can happen:

- Paged-in - Contents of the page are copied from auxiliary storage to central storage. [1]
- Paged-out - Contents of the page are copied from central storage to auxiliary storage. z/OS uses the 'least used' algorithm to select the pages that can be paged out. [1]
- Page Stealing - When the supply of frames is low on central storage, z/OS takes an inactive frame assigned to an active user and makes it available to other work. [1]
- Swapping - Swapping transfers all the address space pages between central and auxiliary storage. [1]

Virtual Address

z/OS manages address spaces in units of various sizes as mentioned below:

- Page - 4KB units of virtual storage.
- Segment - 1 MB units of virtual storage. A segment is a block of sequential virtual addresses spanning megabytes, beginning at a 1 MB boundary. For example, A 2 GB address space consists of 2048 segments.

- Region - 2 - 4 GB units of virtual storage. A region is a block of sequential virtual addresses spanning 2 - 8 GB, beginning at a 2 GB boundary. For example, a 2 TB address space consists of 2048 regions.

These three units determine the virtual address: bits 0 - 32 are the region index (RX), bits 33 - 43 are the segment index (SX), bits 44 - 51 are the page index (PX), and bits 52 - 63 are the byte index (BX) as shown below. [1]

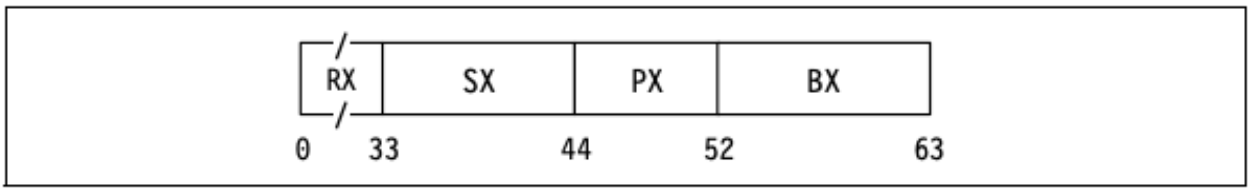

Figure 3: Showing the breakdown of virtual address. [1]

The RX part of the virtual address splits further into three indexes and depending on which of the indexes is a significant part, the respective virtual address can address 4TB (4096 regions), 8 PB (4 million regions), or 16 RB (8 billion regions). [1]

The below picture depicts how real and auxiliary storage combine to create the illusion of virtual storage.

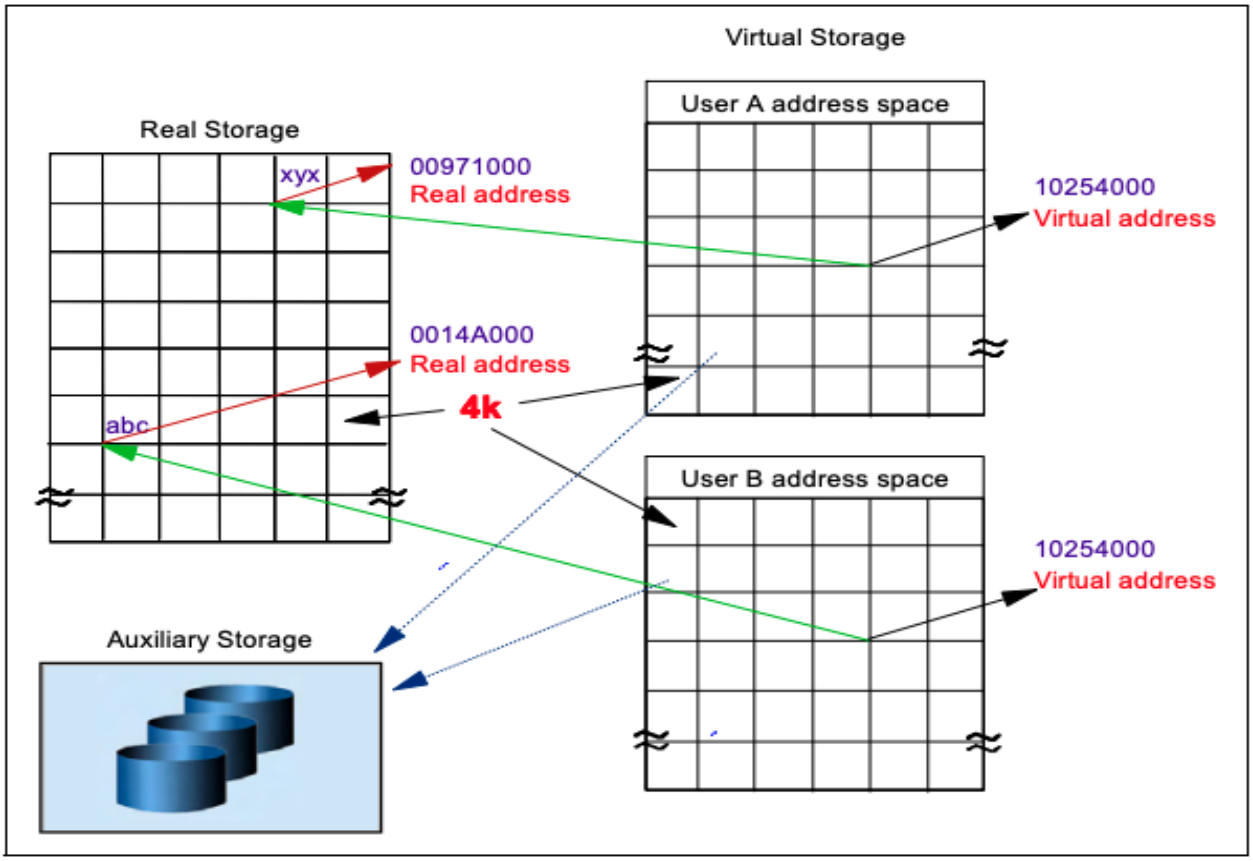

Figure 4: Illustration of real and auxiliary storages forming virtual storge. [1]

Storage managers

All storage types have their storage managers, as explained below:

- Real Storage Manager (RSM) - RSM keeps track of the contents of central storage. It also manages the paging activities such as page-in, page-out, page stealing, and swapping. [1]
- Auxiliary Storage Manager (ASM) - ASM uses the system's page data sets to keep track of auxiliary storage slots. ASM works with RSM during paging activities to locate the proper central storage frames and auxiliary storage slots.
- Virtual Storage Manager (VSM) - VSM handles the requests from user programs to obtain or free the virtual storage. VSM also keeps track of the virtual storage map for each address space.

Addressability

The program's ability to reference the storage available within the address space is called addressability. IBM started with 24-bit addressability back in 1970 in the initial S/370 architecture, which eventually expanded to 31-bit and then 64-bit addressability.[1]

Due to the mainframe's backward compatibility feature, even the current mainframe supports 24-bit addressability. Depending on the program's addressability, address spaces of the respective sizes will be allocated.[1]

Below is the depiction of address spaces that are possible for each addressability.

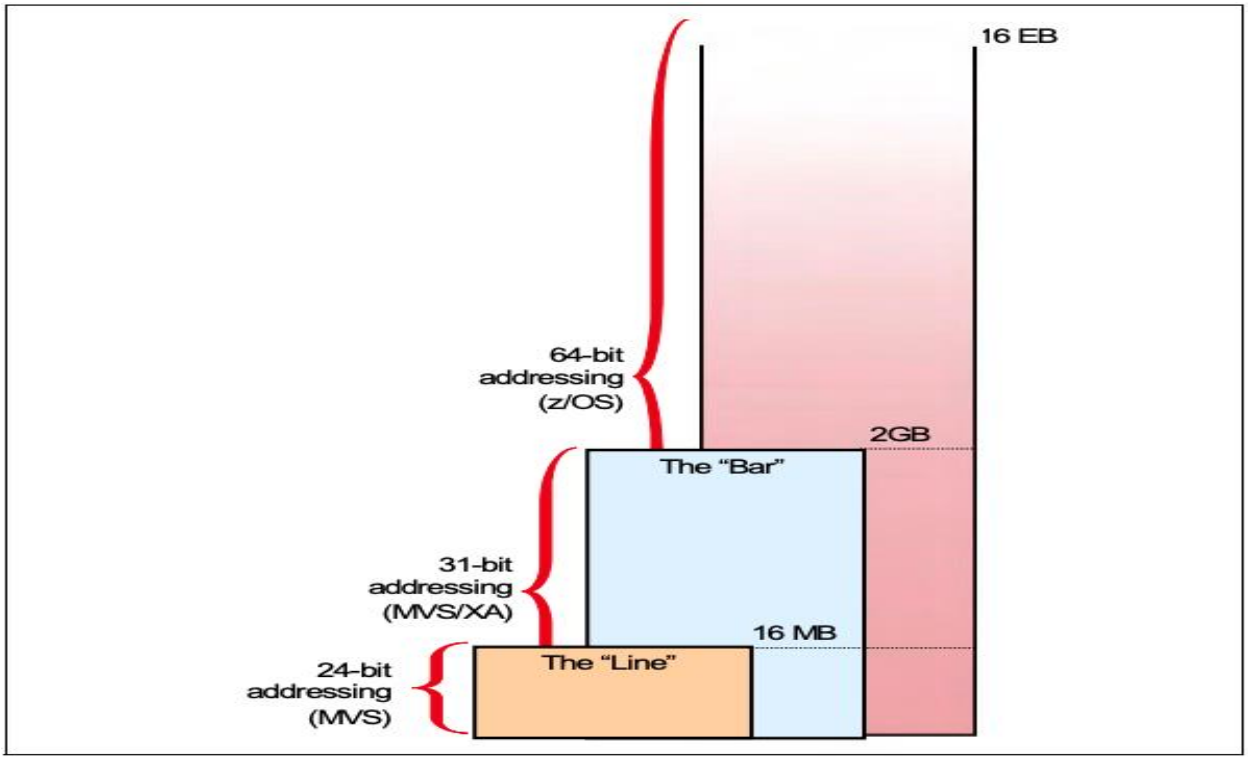

Figure 5: Showing different addressability's and the maximum possible allocations for the respective address space. [1]

Some z/OS programs depend on real addresses, and some require these real addresses to be less than 16 MB (24-bit addressability). RMODE and AMODE options in the program help us to manage these dependencies.[1]

- Residence Mode (RMODE) – Specifies whether the program must reside in storage below 16 MB. Programs with RMODE(24) reside below 16 MB (below the line), while programs with RMODE(31) can reside anywhere in the virtual storage.[4]
- Addressing Mode (AMODE) - Defines the addressing mode in which the program executes. [4]

Storage areas in any 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 available addresses, an address space includes system code, user code, and data. Thus, not all mapped addresses are available for user code and data.[1]

The picture below depicts major storage areas in each address space.

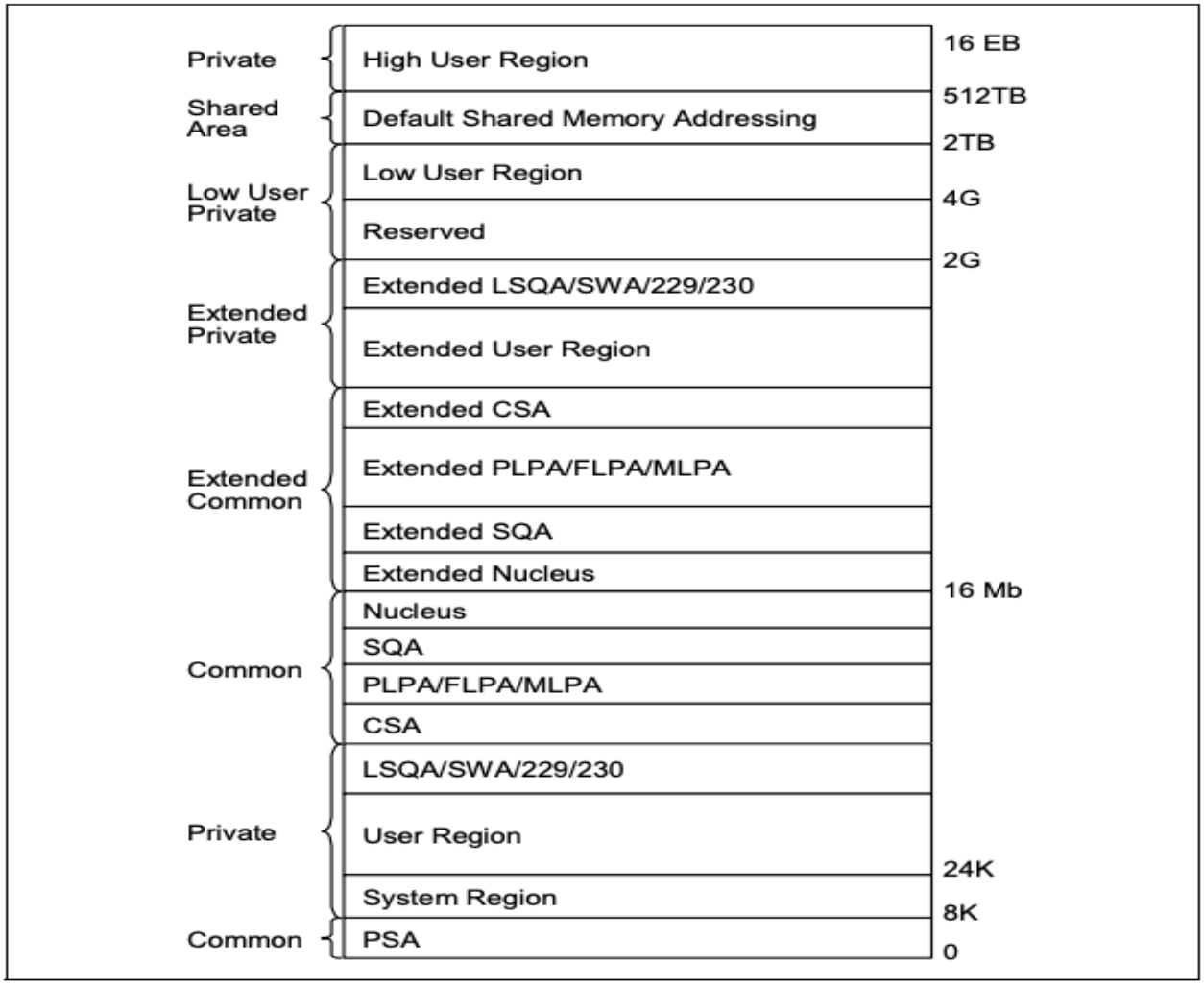

Figure 6: Storage areas in an address space. [1]

As depicted in the diagram, some storage areas, like Nucleus, contain operating system control programs to which the application or user program will only have read access. We can also see some common or shared areas used by multiple address spaces.[1]
Code Example

Below is a basic example of anassembler program doing memory allocation and deallocation:

```
ALLOCATE  CSECT
        STM   R14,R12,12(R13)        SAVE REGISTERS
        LR    R12,R15
        USING ALLOCATE,R12
        LA    R15,64                 REQUEST 64 BYTES
        GETMAIN R,LV=(R15),SP=0      ALLOCATE 64 BYTES FROM SUBPOOL 0
        ST    R1,SAVEAREA            STORE THE ALLOCATED MEMORY ADDRESS
        ...                          (PROCESS MEMORY HERE)
        FREEMAIN R,LV=(R15),A=(R1)   RELEASE THE 64 BYTES
        L     R14,12(R13)
        LM    R14,R12,12(R13)        RESTORE REGISTERS
        BR    R14                    RETURN CONTROL
SAVEAREA  DS    F                    SAVE AREA FOR MEMORY ADDRESS
        END
```

Explanation of code example

1. **GETMAIN (Memory Allocation):**
    - The **GETMAIN** macro requests a memory block (in this case, 64 bytes) from **Virtual Storage**.
    - z/OS allocates memory from a specified **sub-pool** managed by the **Virtual Storage Manager (VSM)**.[1][2]
    - The allocated block is part of the **program's address space**, typically mapped in virtual storage.[1]
2. **FREEMAIN (Memory Deallocation):**

- The **FREEMAIN** macro releases the allocated memory block, making it available for other tasks.[1][3]
- This deallocation updates control blocks the VSM uses to track free and allocated memory.[1]

Conclusion

IBM mainframes, specifically z/OS, remain a powerful platform for enterprises. IBM's memory management is one of the main reasons for IBM's highly scalable architecture, which can handle multiple users simultaneously while processing massive volumes of data.

The illusion of virtual storage created by combining the real or central storage and the auxiliary storage enables the IBM mainframe to handle massive workloads concurrently. By implementing several paging, swapping, and storage isolation features and using different storage managers, z/OS provides the address space level isolation, thereby ensuring high stability and isolation of issues in a particular address space.

References

1. IBM Corporation, Introduction to the New Mainframe: z/OS Basics. IBM Redbooks, 2012.https://www.redbooks.ibm.com/redbooks/pdfs/sg246366.pdf
2. IBM Corporation, GETMAIN – Allocate virtual storage.Retrieved November 2020 fromhttps://www.ibm.com/docs/en/zos/2.1.0?topic=hsp-getmain-allocate-virtual-storage
3. IBM Corporation, FREEMAIN – Free virtual storage. Retrieved November 2020 from https://www.ibm.com/docs/en/zos/2.1.0?topic=hsp-freemain-free-virtual-storage
4. IBM Corporation, Defining MVS residence and addressing modes.Retrieved November 2020 fromhttps://www.ibm.com/docs/en/cics-ts/5.1?topic=programs-defining-mvs-residence-addressing-modes