Infosys Technologies Limited

Education and Research Department

MVS

Feb 2000

Document No.	Authorized By	Version.Revision	Signature / Date
ER/CORP/CRS/OS01/002	S. YEGNESHWAR	2.0	

Modification Log

Version. Revision	Date	Author(s)	Description
1.00	**01/03/96**	**Vasundara.N.**	Original Version
2.00	**01/02/2000**	**Subrahmanya. S.V.** **Ram Mohan Bharadwaj** **Eugene Xavier.P.**	Included new topics such as ESCON, SYSPLEX etc. More illustrations have been given on introduction to Mainframe computers. More explanation has been given on MVS Internals, Recovery and Security. Corrected both spelling as well as grammatical mistakes. Exercises have been included at the end of each Unit. Incorporated comments from Samuel Raj, Rakesh Agarwal and Kamalakannan.

Review Record

Version. Revision	Date	Reviewer(s)	Description	Remarks by the Author
2.00	**01/02/2000**	Samuel Raj, Rakesh Agarwal Kamalakannan	To include new topics such as ESCON, SYSPLEX, MVS Internals, Recovery and Security	Included new topics such as ESCON, SYSPLEX etc. More illustrations have been given on introduction to Mainframe computers. More explanation has been given on MVS Internals, Recovery and Security. Corrected both spelling as well as grammatical mistakes. Exercises have been included at the end of each Unit. Incorporated comments from Samuel Raj, Rakesh Agarwal and Kamalakannan.

Course Description and Design

Course number	OS03	Course name	MVS
Authors	Subrahmanya S.V. Ram Mohan Bharadwaj Eugene Xavier.P.		(Introductory Concepts)
Pre-requisites :	Must have attended the course "CHSSC"		
Estimated effort for preparation	About 15 days		
Estimated course duration	03 days (03 hrs of Lecture + 05 hrs of lab)		

Course objectives
(course objectives state the major outcomes of the training and indicate how to measure the expected performance)

Sl#	Objective	Demonstrable knowledge/skills
1	To introduce participants to the Mainframe Computing System both Hardware and Software.	Ability to understand the overview and major components of an IBM Mainframe system, such as Processor Complex, DASD, ESCON, SYSPLEX etc.
2	To illustrate the constituents of the internals of MVS.	Ability to understand the various OS components of MVS such as its memory management, job management, task management, I/O management, process management and data set management.
3	To illustrate the TSO and ISPF commands and screens.	Ability to understand and use various commands in TSO for interactive program development and job related activities. Also, to use ISPF environment for software development under Mainframe system
4	To illustrate the use of VSAM and non-VSAM data sets.	Ability to understand and use data sets such as VSAM and non-VSAM.
5	To illustrate participants regarding the issues like performance, recovery, security in MVS	Ability to appreciate the issues of performance, recovery and security and how they are implemented in MVS
6	To introduce various parameters related to the performance enhancement.	Ability to identify the importance of parameters to enhance the performance.

Sources consulted

1. Robert. H. Johnson , *MVS – Concepts and Facilities*, McGraw-Hill Book Company, 1989.
2. Doug Lowe, MVS JCL - *Mike Murach & Associates*, 1994.
3. *MVS/DFP - IBM Manual*
4. Doug Lowe , *MVS TSO PART1 CONCEPTS AND ISPF*, Mike Murach & Associates, 1991.
5. Jay Ranade, Hirday Ranade, *VSAM Concepts, Programming, and Design* - McGraw-Hill Book Company.
6. *IBM On-line Manuals - IGG3L100, IGG3U100 and IGG3V400. On MVS.*

Session Plan (The session plan gives the day-wise break up of the topics to be covered along with the estimated time required to deliver each unit)			
Sl#	Unit name	Unit objectives	Lecture Time
1	Introduction to Mainframe Computing.	To explain the various aspects the hardware and software of mainframes especially IBM right from its evolution. To Introduce the participants to TSO/ISPF	03 hrs
2	Introduction to MVS internals.	To explain the major components such as Virtual Memory Management, Job Management, Task Management, Process Management and I/O management.	03 hrs
3	Data set management, Performance, Recovery and Security.	To illustrate the usage of data sets such as VSAM and non-VSAM. To convey the issues of performance, recovery and security and how they are implemented in MVS	03 hrs

CONTENTS

1. Introduction to Mainframe Computing

1.1 Classification of Computers

Computer systems used for business purposes can be divided in to three classes: microcomputers, minicomputers and main frame computers. Though these divisions are loosely based on the size of the computer systems, there are no hard and fast rules for deciding exactly where one category ends and the next begin. Hence the largest minicomp uter systems are often larger than the smallest mainframe computers.

The "size" of a computer system is dependent on the size of a computer's hardware configuration, the nature of its applications, and the complexity of its system software. This helps us to classify a system as a microcomputer, minicomputer or mainframe.

Irrespective of size, all computers consist of two basic types of components viz., processors and input/output (I/O) devices. Fig.1.1 illustrates these components. The processor consists of three parts: the Central Processing Unit, or CPU, main storage and device controllers. The CPU executes instructions, main storage stores instructions and data processed by the CPU and device controllers let the CPU and main storage connect to I/O devices.

Input / Output devices fall into two classes: those that provide input and output to the system, such as terminals and printers and those that provide secondary storage, such as tape and disk drives.

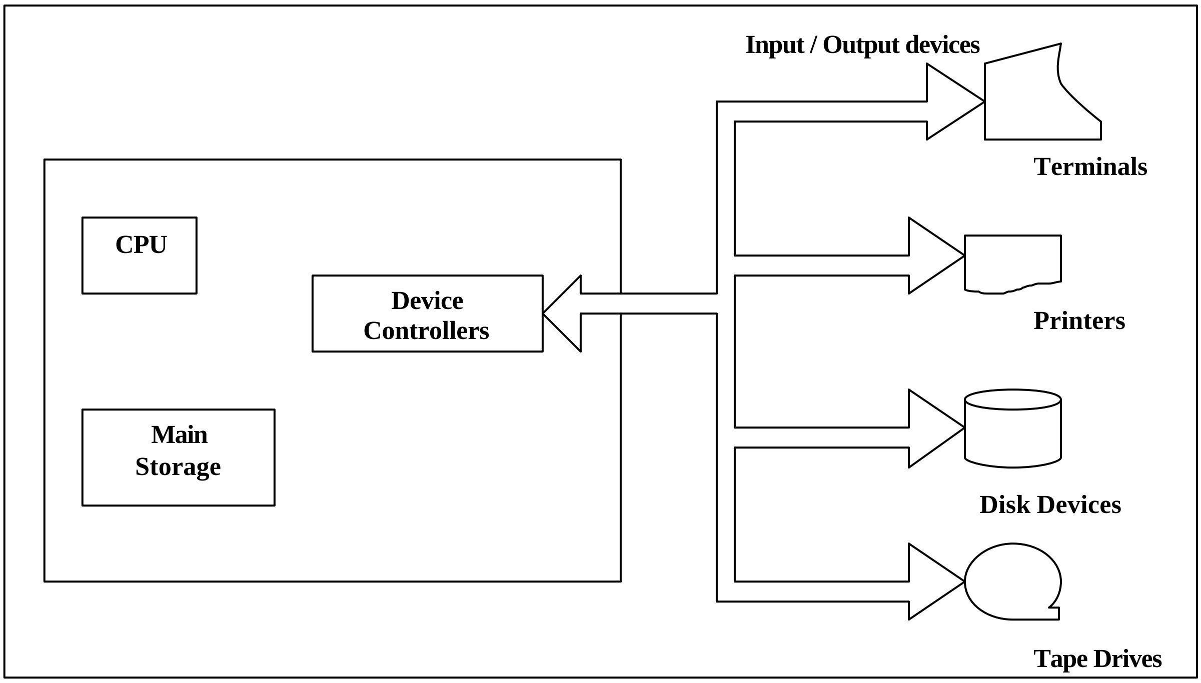


Fig.1.1 The basic components of a modern Computer system

Though all computer systems consist of three basic components, the way those components are combined for a particular computer system varies depending on the system's requirements.

1.1.1 Microcomputer or Personal Computer

Microcomputer is primarily intended for stand-alone use by an individual. Microcomp uters, are small, single-user systems which provide a simple processor and just a few input/output devices. Fig. 1.2 shows a configuration of a typical microcomputer. This system consists of a processor with 1MB of main storage, a keyboard, a display monitor, a printer, and a diskette drive with a capacity of 1.2 MB and a 40 Mb hard disk.

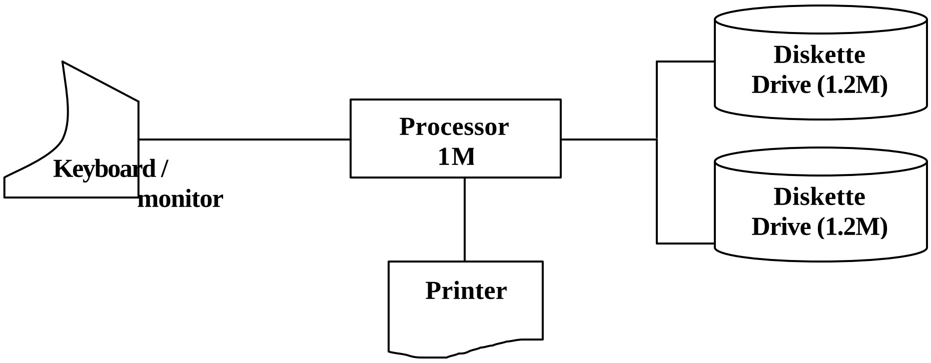


Fig.1.2 Configuration of a microcomputer

1.1.2 Minicomputer

Unlike microcomputers, most of the minicomputers provide more than one terminal so that several users can use the system at a time. Such systems are referred to as “multi-user systems”. Fig.1.3 illustrates a minicomputer.

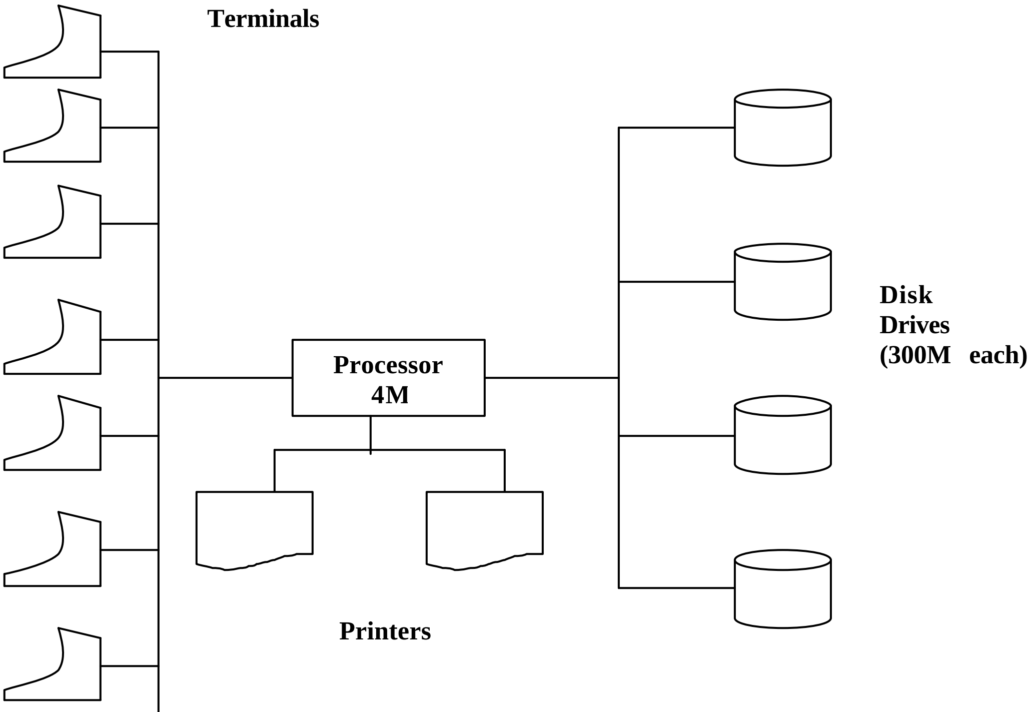


Fig. 1.3 Configuration of a minicomputer

A minicomputer is functionally intermediate between a microcomputer and a mainframe computer. Compared to mainframe, a minicomputer consists of less storage, less processing power and lower speed. The minicomputer configuration shown in fig.1.3 has 4MB of main storage, 8 terminals, two printers and four disk drives totaling 1200MB.

1.1.3 Mainframe Computer

A mainframe computer consists of the same basic types but has more I/O devices and larger storage capacities. It is a computer with extensive capabilities and resources to which other computers may be connected so that they can share facilities.

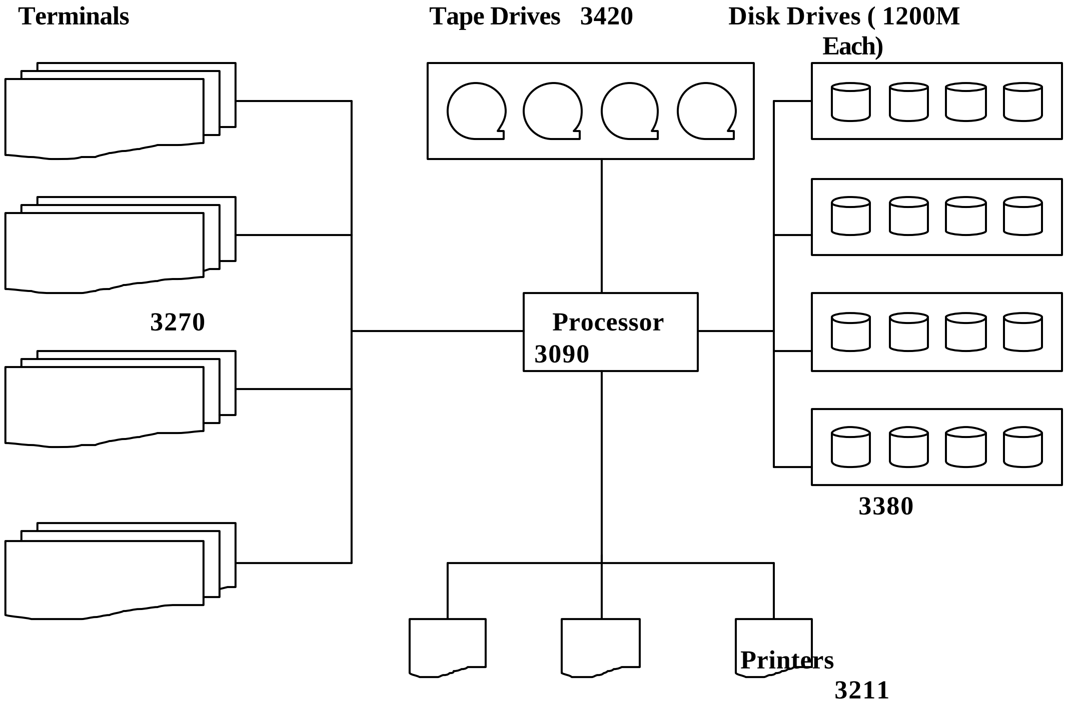


Fig. 1.4. Configuration of a mainframe computer

Fig. 1.4 shows the configuration of a mainframe computer which includes 16 disk drives, four tape drives, three printers, and a large number of terminals. The processor's main storage is 512 MB, and the total disk capacity is nearly 40 billion bytes. The most popular form of mainframe computer is the System/370 family. Today the largest member of the family can be configured with more than 2 billion bytes of memory storage.

The processing power is another aspect of its size. Generally large computers are used to process large amount of data and supports large number of users.

Another distinction among computers is the scope of the system software that is required to manage the computer system's resources so that application program can perform useful work.

1.2 History of Mainframe

In the 1960's, IBM had three categories of computers.

- Scientific computers (IBM 704, 709,7040-44 and 7094) allowed very large (or small) numbers to be represented and manipulated.

- Decimal Computer (IBM 7070 and 7080) were primarily designed to perform calculation with dollars and cents.
- Character Computers (IBM 1401, 1410 and 7010) were character oriented computers, designed mainly for character data handling.

You are required to have three computers if you want to solve a problem from three domains viz., scientific, business and text data processing. None of the applications are portable from one machine to another.

It was observed that major share of the investment went to software development. Development of MVS has taken into account this factor by providing upward compatibility, for code, written for the earlier systems. A program written for one model of the System/360 will run on the next version of the hardware.

In 1964, IBM announced the System/360 architecture. The System/360 unified, machine architectures. The System/360 models had binary, decimal, floating point (including, single and double precision), character, and word orientation-all in a single profitable computer.

In 1972, IBM introduced a new range of machines called System/370 family. This family was the mainstay of IBM for many years and underwent continuous improvement, both in terms of hardware and software. This was later followed by the XA, 390 and ES/9000 range of machines, which is the current offering in the mainframe range.

1.3 MVS Evolution

1.3.1 Operating System

An operating system is a program that acts as an intermediary between user of a computer and the computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs. The operating system controls and coordinates the use of hardware among various application programs for various users. The operating system is viewed as a resource allocator. A computer system has many resources (hardware and software) that may be required to solve a problem. CPU time, memory space, file storage space, I/O devices and so on. The operating system acts as the manager of these resources and allocates them to specific programs and users, as necessary for the latter's tasks. Since there may be many possible conflicting requests, resources are allocated to operate the computer system efficiently and fairly.

The primary goal of an operating system is thus to make the computer system convenient for use, the secondary goal is to use the computer hardware in an efficient manner.

Components of OS

- Process Management
- Main Memory Management
- Secondary Storage Management
  - Free Space Management
  - Disk Scheduling
- I/O System Management
  - A buffer caching system
  - A general device driver interface
  - Drivers for specific hardware
- File Management
  - The creation and deletion of files
  - The mapping of files onto secondary storage
  - The back up of files on stable storage
- Protection
- Networking

1.3.2 History of an Operating System

Early systems

Initially, there was only computer hardware. Early computers were (physically) very large machines run from the consoles. Programming was very difficult since programs were written in machine language. The programmer would write a program and then operate directly from the console. First, the program would manually be loaded into memory, from the panel switches (one instruction at a time), paper tapes or punched cards. Then the appropriate buttons would be pushed to set the starting address and to start the execution. As the program ran, the programmer/operator could monitor its execution by the display lights on the console. If errors were discovered, the programmer could halt the program, examine the contents of memory and registers, and debug the program directly from the console. Output was printed or was punched onto paper tapes or cards for later printer.

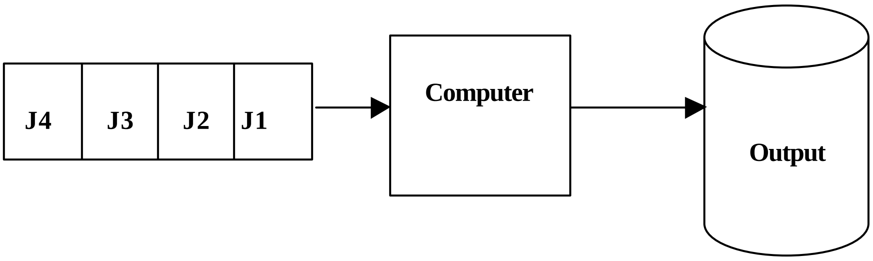


Fig. 1.5 Batch Processing

An important aspect of this environment was its hands-on interactive nature. The programmer was also an operator of the computer system. Operating a computer was relatively simpler than writing a program. The advent of high level languages (COBOL, FORTRAN, etc.) had simplified programming prospects but addition of more sophisticated hardware to the system, made the operating part extremely difficult.

In early systems, operating system was very simple. It did the loading and exe cution of the program from the job queue. Job queue was built in the beginning, i.e., before starting of the computer. This was the beginning of Batch Operating System.

1.3.3 The MVS Operating System

MVS is an evolutionary system. It has evolved over the architecture of Batch Processing concept. This means it takes a job from the job queue, and executes it. MVS is one of the most complex software written so far. Performance and ease of use, are the two factors that influenced the evolution of the MVS. The differences between Batch Processing and Interactive Processing have been given in table1.1.

Batch Processing	Interactive processing
Jobs are submitted for execution by the processor at later time.	Jobs submitted are executed immediately.
Absence of any interaction from user.	User interaction is a usual thing.
Performance measure is throughput.	Performance measure is response time.
User interaction is simulated by means of system files.	This may provide batch processing. This processing is also known as background processing.
Snap shot of the output is used for debugging.	Interactive debugging may be provided.

Table1.1 Batch Processing Vs Interactive Processing

1.3.4 Primary Control Program (PCP)

In 1966, IBM introduced the first OS for its System/360 range of machines, called Primary Control Program (PCP). This operating system was such that only one program was resident in memory at any given point of time. The memory map looked like the one shown in figure. 1.6.

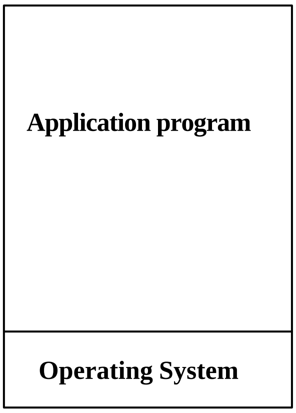


Fig. 1.6 Illustration of PCP

1.3.5 Multiprogramming with Fixed Number of Tasks (MFT)

The biggest problem with PCP was that processor cycles were wasted whilst the application program made a request for I/O. To improve processor utilization, another OS called Multiprogramming with Fixed number of Tasks (MFT) was introduced.

Multiprogramming implies that a number of programs are resident in memory at any given point of time. Of these, one program is running on a processor. When a program makes a request for I/O its state is saved and the program is suspended. Control is given to another program and it continues from where it left off. In this fashion, ultimately all programs run to completion after many cycles of running and sleeping.

In MFT, many fixed partitions were created at machine startup. The sizes of the partitions could vary. Each program would be loaded into a free partition of appropriate size.

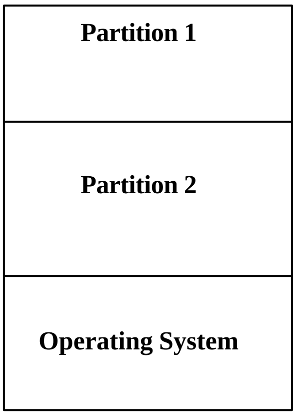


Fig. 1.7 Illustration of MFT

1.3.6 Multiprogramming with Variable Number of Tasks (MVT)

In MFT, a program had to wait until a partition of suitable size was available even though the available memory might be enough to accommodate it. A variant called Multiprogramming with Variable number of Tasks (MVT) was introduced. In this, the regions were not frozen at startup, but were dynamically made available to the program upon request. Fig.1.8 below illustrates this concept by showing the memory map at two different instants of time.

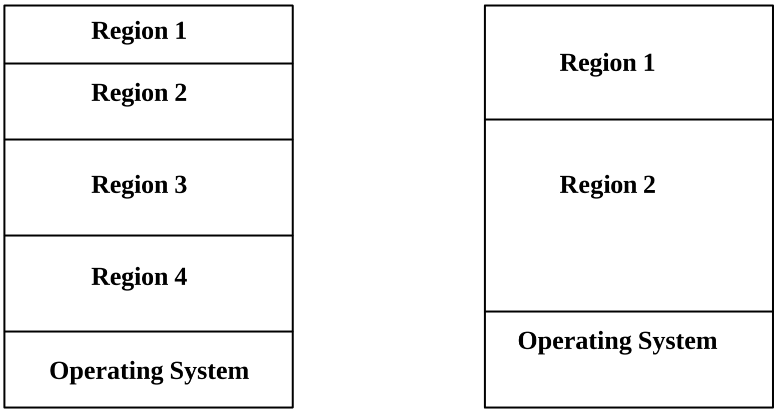


Fig. 1.8 Illustration of MVT

1.3.7 Operating System / Virtual Storage (OS/VS 1)

A program can be loaded in a contiguous space in central memory. All the operating systems above suffered from the limitation that the maximum size of any program was restricted by the size of Central Storage. Also, another problem is that of unusable memory areas due to memory fragmentation.

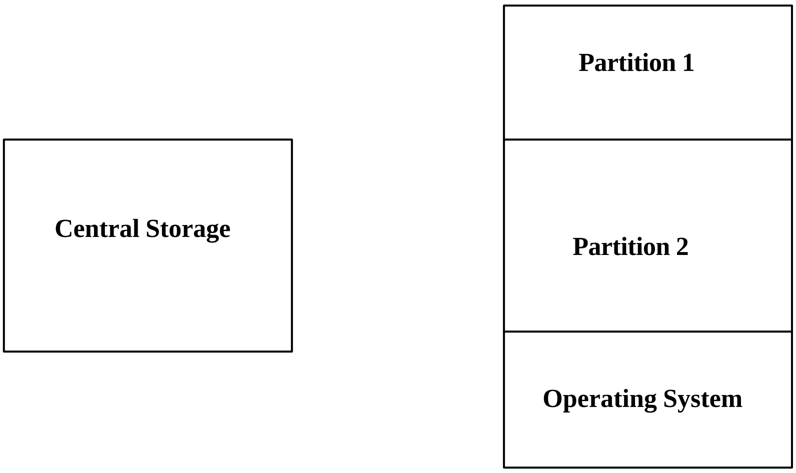


Fig. 1.9 Illustration of OS/VS1

To overcome these problems, IBM introduced Operating System/Virtual Storage 1 (OS/VS1) which was an enhancement on MFT. This is like MFT, but the partitions are created in virtual address than in real address. In this OS, the operator assigned virtually contiguous partitions. This virtual storage is back up in Central Storage and on auxiliary disk storage. The operating system takes care of bringing in portions of the program from disk into Central Storage as needed.

1.3.8 Single Virtual Storage (SVS)

Another variant of OS/VS1 called OS/ VS2 Single Virtual Storage (OS/VS2 SVS) was released. This was an enhancement of MVT with the addition of virtual storage. This freed the operating system, as did MVT, from the problem of locating a suitable partition to run the program.

In SVS, like in OS/VS1, the programs are relocatable, i.e. they can be placed anywhere in memory.

1.3.9 Multiple Virtual Storage (MVS)

With the growth of OS code, the available area for execution of user programs reduced. This problem was solved with the introduction of MVS in 1974. In this OS, each user is given a separate address space of 16 Mb. System code is common to all the users and maps onto each address space (An address space is simply the complete range of addresses - and as a result, the number of storage locations that can be accessed by the computer. The maximum size of the computer's address space is limited by the number of bits used to represent an address).

Additionally, Some system functions were taken out of the common area and moved into separate address spaces to increase the available virtual memory for each use. Separate segment and page tables for each address space ensured the integrity amongst users.

Communication across Address spaces was achieved

- through the common area
- utilizing Cross Memory Services (CMS)

		Common Area		
System Address Space 1	**System Address Space 2**	**User 1 Address Space**	**User 2 Address Space**	**User 3 Address Space**
		Common Area		

Fig. 1.10 Address Spaces in System/370

Common area is Virtual Storage shared by all address spaces in MVS. Cross Memory Services (CMS) is a set of control structures and macros within MVS, which invoke DAS instructions that mange cross-memory environment.

1.3.10 MVS/Extended Architecture (MVS/XA)

In 1983, an improvement to the hardware architecture of the System/373/6/963/6/960 called the System/370 - Extended Architecture (System/370-XA) was introduced. This architecture was supported by a new release of MVS (called MVS/XA). The main improvements in this were:

- ❖ The enhancement of address space size from 16 Mb to 2 Gb by the provision for 31 bit addresses (the 32nd bit is used for distinguishing between 24 and 31 bit addresses) and the common area was moved to around the 16 Mb line (shown in the fig.1.11 below). The XA/ESA provides two sections of each area one below the 16-MB line and one above the 16-Mb line. The private area resides below the 16Mb line and the extended private area resides above the 16MB line. With this enhancement,old programs (24 bit) will be relieved of tight memory constraints and will be able to allocate more buffer to their file operations. This now allows to allocate more buffer for file read.

User Area 1	**User Area 2**	**User Area 3**	**User Area 4**	**User Area 5**
◀		***Extended Private Area***		▶
Common	**Common**	**Common**	**Common**	**Common 16M**
Area	**Area**	**Area**	**Area**	**Area**
User Area	**User Area**	**User Area**	**User Area**	**User Area**
◀		***Private area***		▶

Fig 1.11 Address Spaces

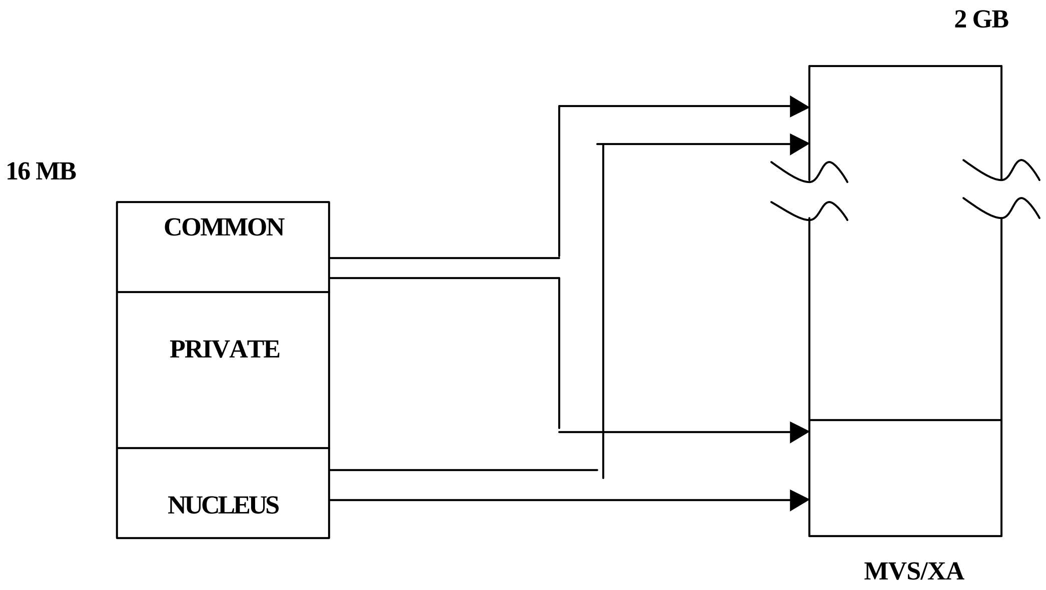


Fig 1.12 Virtual Storage difference

(The conversion from MVS/370 to MVS/XA moves some areas from below the 16-megabyte line to above the 16-megabyte line. Private Virtual Storage also has a part below the line and a part available above line).

- The provision of a channel subsystem to free the processor from I/O devices.
- Expanded Storage.

- Closely associated with Central Storage. It can be thought of as an extension of Central Storage. It is cheaper RAM, much faster than DASD access and slower than Central Storage.
- Expanded storage differs from Central Storage in terms of addressing. An address generated for expanded storage points to a 4096-byte block or frame of expanded storage, whereas an address generated for Central Storage points to a separated byte of Central Storage (byte address).

Operating system	Hardware	Average # of jobs running concurrently	Storage Types	Virtual Storage limits
PCP	System 360	1	Main	Virtual Storage not available
MFT/MVT	System 360	10-20	Main	Virtual Storage not available
SVS	System 370	30 - 50	Main Auxiliary	16 megabytes * Programs share address space
MVS/SP MVS/ Base	System 370	100s	Main Auxiliary	16 megabytes * Each program has its own address space
MVS/XA	System 370/XA	1000s	Main Expanded Auxiliary	2 gigabytes
MVS/ESA	ESA/370	1000s	Main Expanded Auxiliary	2 gigabytes * Addition of 2 new data storage areas - Data space - Hiperspace

Table 1.2 MVS Evolution

- Another difference is that the MVS software maintains information in a control block for each frame of expanded storage. That information is used to preserve data integrity and for LRU management of expanded storage. For Central Storage the hardware reference and change bits are used for these functions.

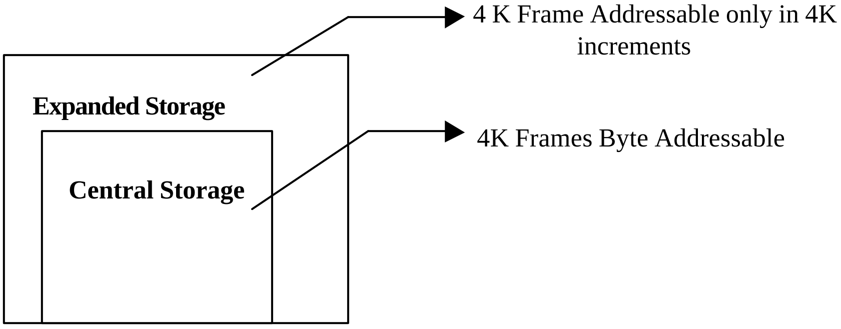

4 K Frame Addressable only in 4K increments
Expanded Storage
4K Frames Byte Addressable
Central Storage

Fig. 1.13 Central and Expanded Storage

- Data is moved between Central and expanded storage without performing I/O operations.
- Data cannot move directly from expanded storage to DASD or vice versa. It has to move out through Central Storage from expanded storage.

1.3.11 MVS/Enterprise Systems Architecture (MVS/ESA)

In 1988, IBM introduced MVS/ Enterprise Systems Architecture (MVS/ESA) and System 370/ESA as a further enhancement to MVS/XA. In MVS/ESA, the main additions are: -

Both Data space and Hiperspace provide application with the option to reduce or eliminate I/O operations. Highly read, active (as opposed to update-intensive data) is a good candidate to load into processor storage.

The ability of the program to store data in a data space whole size can be up to 2 Gb in size. The program can access up to 7999 data spaces. This means that each user can access 16000GB space. The Advanced Address Space Facility (AASF), an ESA/370 hardware facility, allows the program to access multiple address spaces and data spaces concurrently.

The differences between Address space and Data space is summarized in table 1.3.

ADDRESS SPACE	DATA SPACE
Contains instruction and data.	Can contain only data. Even if program s loaded in data space, it is considered as data.
Common areas and nucleus is mapped on Address space.	None of common areas and nucleus is mapped on data space.
An application can have one address space.	An application can have access up to 7999 data space, each the size of 2GB.

Table 1.3 Address space Vs Data Space

- Consider using Data space only when

- Data can be manipulated directly and is not structured in 4K blocks
- One copy of the data can be shared among multiple address space.
- Program will be written in Assembler language.

- Consider using Hiperspace only when

- Data can be structured in 4K blocks.
- Data will be used by only the owner's space.
- Program will be written in high-level languages such as COBOL, PL/1 and FORTRAN.

The differences between data spaces and hiperspaces are summarized in table 1.4.

	DATA SPACE	HIPERSPACE
Access	Access Registers	MVS System Services
Addressability	Byte	4Kb Blocks
Storage	Central, Expanded, Auxiliary	Expanded, Auxiliary
Language Support	Assembler	Assembler and High Level

Table 1.4 Data space Vs Hiper space

1.4 Characteristic Features of Mainframe Operating Systems

1.4.1. Virtual Storage

In most Computer Systems, the processor's main storage is among the most valuable of the systems resources. Hence modern mainframe computer operating systems provide sophisticated services to make the best use of the available main storage. Among the most important of these services is virtual storage. It is a method by which means processor will be able to work on non-contiguous memory in central memory. In Von Neumann architecture Program needs to be loaded in contiguous memory. In this method the processor feels the addresses generated by PWD are contiguous, it is a translation scheme that translates the address in to real address.

Virtual Storage is a technique that lets a processor simulate an amount of main storage that is larger than the actual amount of real storage. For example a processor that has 212 M bytes of real storage might use virtual storage to simulate 2GB bytes of main storage. To do this, the computer uses disk storage as an extension of real storage. Other data and instructions can be placed temporarily on disk storage, and recalled into main storage when needed.

1.4.2. Multiprogramming

Multiprogramming means that the computer lets more than one program execute at the same time, i.e., at any given moment, only one program can have control of the CPU. Some processing operations like reading data from an input device - take much longer than others. As a result, most programs that run on mainframe computers are idle a large percentage of the time, waiting for I/O operations to complete. If programs were run one at a time on a mainframe computer, the CPU would spend most of its time waiting. Multiprogramming simply reclaims the CPU during these idle periods and lets another program execute.

1.4.3. Spooling

It means Simultaneous Peripherals Operations On Line. A significant problem which must be overcome by multiprogramming systems is sharing access to input and output devices for the programs that execute together. For example, if two programs executing at the same time try to write output to a printer, the output from both programs is intermixed in the printout. One way to avoid this problem is to give one of the programs complete control of the printer, which defeats the purpose of multi programming because the other program has to wait until the printer is available.

To provide shared access to printer devices, spooling is used. Spooling manages printer output for applications by intercepting printer output and directing it to a disk device instead. When the program finishes, the operating system collects its spooled print output and directs it to the printer. In a multiprogramming environment, the operating system stores the spooled output separately on the disk so that it can print each program's output separately.

1.4.4. Batch processing

Work is processed in units called jobs. A job may cause one or more programs to execute in sequence. One of the problems that arise when batch processing is used is managing how work flows through the system. To manage this in the multi-user system, the Job Entry Subsystem (JES) processes each user's job in an orderly fashion.

1.4.5. Time - sharing

In a time-sharing system, each user has access to the system through a terminal device. Instead of submitting jobs that are scheduled for later execution, the user enters commands that are processed immediately. Hence this is sometimes called "Interactive Processing", as it allows users to interact directly with the computer. Time-share processing is called "Foreground Processing" and batch job processing is called "Background Processing".

1.5. System/370 Hardware Concepts

1.5.1. Processor Complex

CPU: The Central Processing Unit (CPU) is where instructions are executed. There are several components in a CPU.

Registers: Registers are special purpose areas in a computer, which hold addresses or data. There are several types:

Program Status Word (PSW) : The Program Status Word (PSW) is a special hardware register that works in conjunction with control registers to govern exactly how each instruction is executed. Each Central processor will be governed by its own PSW. The system software on behalf of problem, program manipulates the contents of the PSW. It is equivalent to Program Counter.

General-purpose registers: There are 16 general-purpose registers numbered from 0 to 15. Each register is a full word in size-four bytes, 32 bits. The registers contain a) Data, such as counter b) Address, such as the address of a program or index pointer to list of data. When it contains the address it is als o referred as base register.

Floating point registers: There are four floating-point registers. They are numbered zero, two, four and six. Each register is a double word in size-8 bytes, 64 bits. The floating point registers are not the same as the general purpose registers. Floating point registers load from or store to single word or double words.

Control registers: There are 16 control registers (CR), which are used by MVS to control address spaces and pass information to the hardware.

Access registers: There are 16 access registers, which are used to access Data spaces.

Arithmetic Logic Unit (ALU): The ALU contains the circuits necessary for performing arithmetic operations. ALU accesses and stores information in the registers.

High speed buffers: The high-speed buffer is also called "cache". The purpose of the CPU cache is to speed up processing instructions. CPU cache is used to "pipe line" or overlap instruction and data manipulation. In other words it makes the processor "faster" than the raw speed.

Computer Systems Controller: The Computer Systems Controller is responsible for all access outside the CPU. The following are the functions of the Computer Systems Control section.

- Coordinate all references to Central Storage for both instructions and data.
- Oversee the execution of instructions. There are two sub functions to instruction processing:

The first part of instruction execution called I mode, is to cause the instruction to be fetched from Central Storage. The second part of instruction execution, called E Mode, is to decode the instruction and what "operands" are required. If the instruction needs information from the registers, then that data is made available. If the instruction needs information from or written to Central Storage, then the access is made.

Physical Storage Hierarchy

CPU	Fastest	Most expensive	Nanoseconds
Registers			Nanoseconds
High speed buffer			Nanoseconds
Central Storage			Nanoseconds
Expanded Storage			Microseconds
Disk controller cache			Milliseconds
Auxiliary Storage (disk)			Milliseconds
Auxiliary Storage (tapes)	Slowest	Least expensive	milliseconds

Channels: The channels are processors, slower than the main processor, used to control movement of data from outside the Processor Complex to inside the Process Complex.

1.5.2 Uniprocessing Vs Multiprocessing

Uniprocessor: A Uniprocessor has one processor executing tasks, In a Uniprocessor, a single process has access to storage and to the channel subsystem.

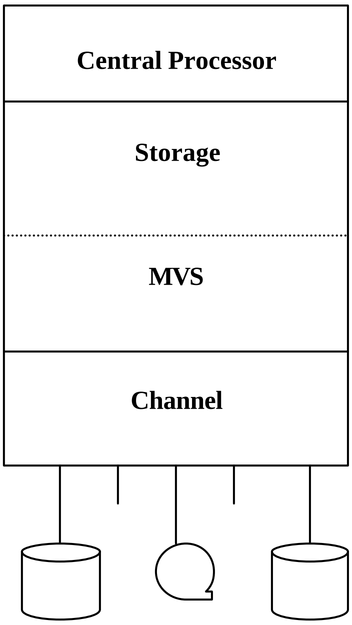


Fig. 1.17 Schematic of Uniprocessor

Multiprocessor

The term refers to the ability to have more than one task executing at the same time, not just two or more tasks running in the Processor Complex at the same time, but two instructions executing at the same point of time.

Non-Partitionable Multiprocessors

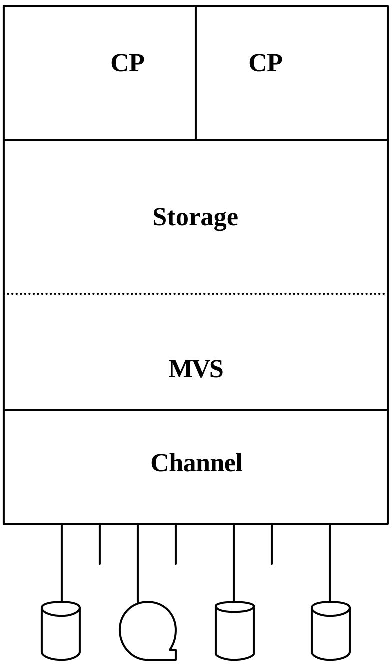


Fig.1.18 Schematic of Non-Partitionable Multiprocessor

In a tightly coupled multiprocessor two or more processors have access to storage and the channel subsystem, and execute instructions for a single MVS system. Processors that are not physically partitionable can execute independently for system availability but cannot form independent systems.

Partitionable Multiprocessors

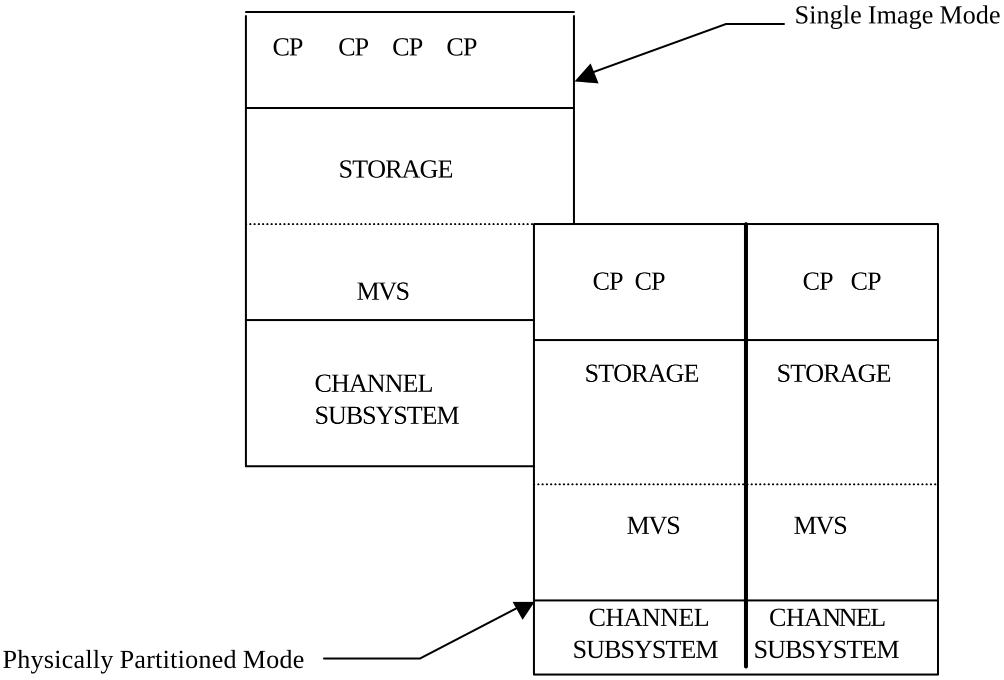


Fig. 1.19 Schematic of Partitionable Multiprocessor

Partitionable multiprocessors can operate in single image or physically partitioned mode. In single image mode all processors share all of storage and all of the channel subsystem and execute instructions for one MVS system. Each of the partitioned system requires independent operating systems: either or both may be MVS.

PR/SM - Logically Partitioned Mode

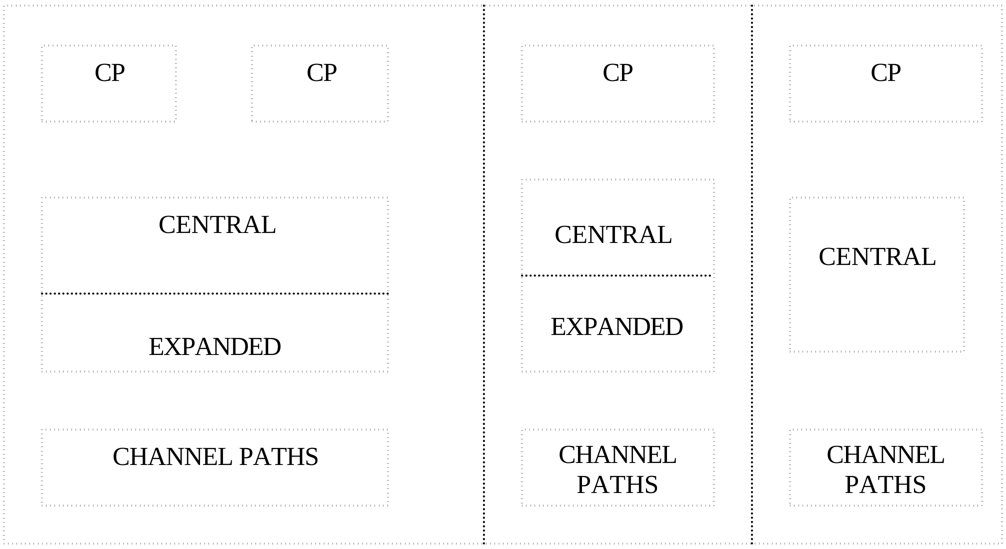


Fig. 1.20 Logical Partitioning

The 3090 E/S models have a feature called Processor Resource/System Manger (PR/SM), which allows the processor to be divided into logical partitions.

- Logical Partitions (LPAR) mode allows multiple logical partitions of Processor, Central Storage, Expanded Storage and Channel paths.

- Each logical partition has its own operating system.

1.5.3. Central Storage Management

- **Hardware Addressable - ROM**

Read Only Memory (ROM) contains the instructions called Microcode, which processes the system / 370 instructions. Microcode as such is a set of microinstructions in the hardware that tells the hardware how to execute the instructions designated in the processor complex. Microcode is installed in a part of the Process Complex. That is not addressable by the operating system (MVS) or the application programs. It is an alternative to electronic circuitry to implement function in Processor Complex.

- **User addressable: Central Storage**

Central Storage is allocated in 4-Kb blocks. Each block of Central Storage is referred to as "fixed pages", which are pages that some task has had assigned to it and then issued an MVS supervisor call to request that the page be "Fixed" in storage. For example, a "fixed page" it is used to store part of the MVS list of tasks or jobs that are executing. These pages are "long term fixed pages".

Another type of page fixing is for I/O operations. If a program has a 32-Kb block to read, then 32Kb of Central Storage is fixed, the I/O is started, and once the I/Os are completed, the 64-Kb pages are reset to "pageable". This is an example of "short-term fixed pages".

- **System Addressable: Auxiliary Storage**

When MVS runs but of Central Storage pages, the auxiliary storage viz., DASD space is used for "paged - out" pages. The MVS routines, which control DASD page space, are called Auxiliary storage Manager.

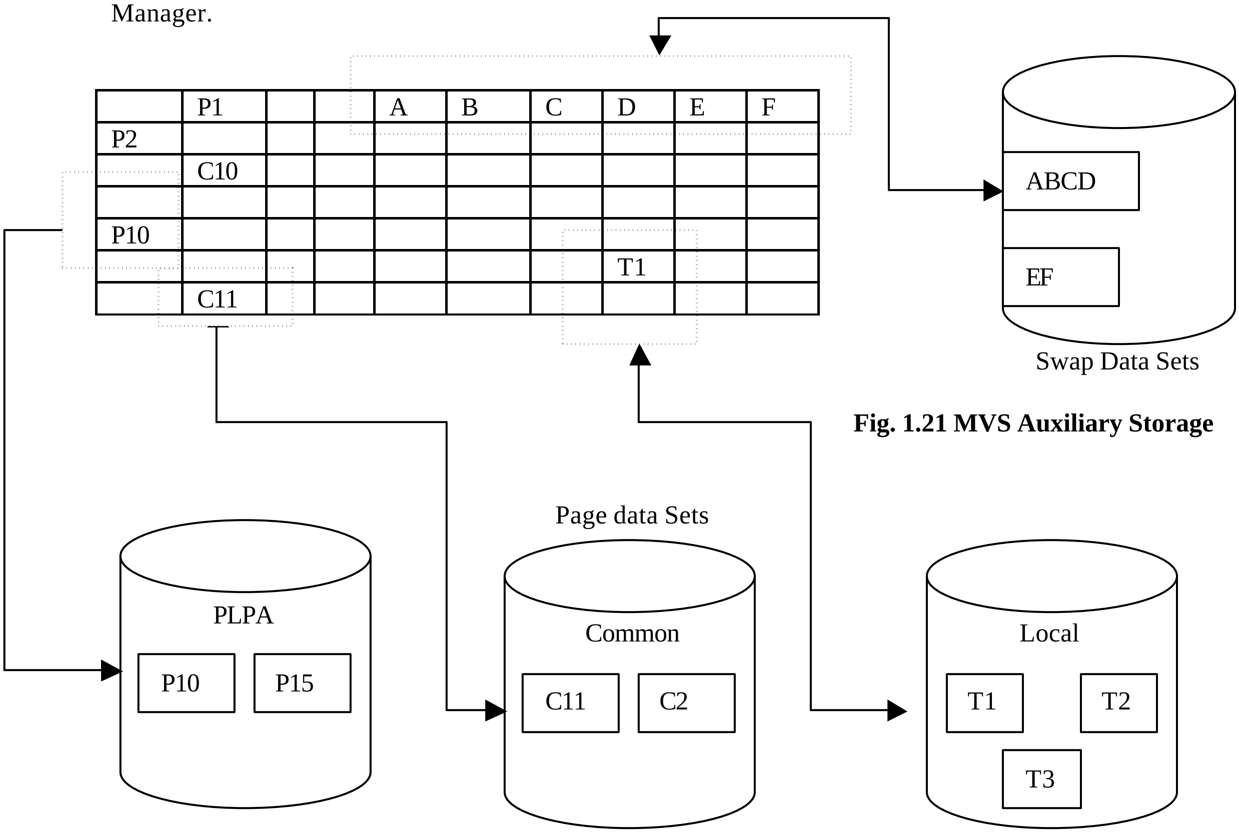


Fig. 1.21 MVS Auxiliary Storage

Types of DASD page space:

a) *Page data sets* contain pages or blocks of pages assigned to a task. There are three types of page data sets:

- The Pageable Link Pack Area (PLPA) data set contains only the pages of the MVS modules, which are in the "link pack area". Only are PLPA data set is allowed.
- The common data contains only the MVS data area pages, which are common to all address spaces. Only one common data set is allowed.
- The local data set contains the pages of any address space, which are not in Central Storage. More than one local data set is allowed.

b) *Swap data sets* contain blocks of pages, which are assigned to a task.

♦ **System Addressable- Expanded Storage**

A kind of internal storage called expanded storage is an optional storage that provides more resources to the operating environment. Expanded storage is a large repository for page which are not needed now and have a good probability of requested soon.

1.5.4 Central Storage Protection

Protection of Central Storage is necessary to keep one application from intentionally or accidentally harming another application - including MVS and its subsystems. Central Storage protection is implemented by assigning attribute to Central Storage page frames.

(a) Storage key Protection

Each block of Central Storage has an additional ridden - byte associated with it called the "Storage Protection Key". It is similar to the parity bit but is another byte altogether and is one byte per 2048 bytes and not one bit per byte like the parity bit. The storage key has 8 bits. The left most four bits are allocated to the protection key. Thus with 4 bits, we have storage keys from 0 to 15.

If all the four bits are zero, that block has a storage key of 0. Key 0 is used to protect MVS storage. Thus the total number of non-zero protection key is 15. Key o is used to project MVS storage. Thus the total number of non-zero protection keys is 15.

A storage protection key value is associated with each Central Storage page frame and MVS associates each page with a specific task. How is it done? The Program Status Word (PSW) contains a protection key as part of the PSW. For MVS/370, MVS/XA and MVS/ESA, bits 8-11 contain a storage key anytime the CPU attempts to store data into Central Storage key. Anytime the CPU attempts to store data into Central Storage, the storage keys are compared.

"Protection Key" in the "current PSW" is matched against the "Storage Key" of the block before a write takes place. The write is allowed if the keys match or if the protection key in the PSW is zero. A protection exception will result in a program error interrupt if the keys do not match and the protection key in the PSW is not zero. There are two types of PSW. The BC mode is used for non-VSAM operating systems. The EC mode is used for MVS.

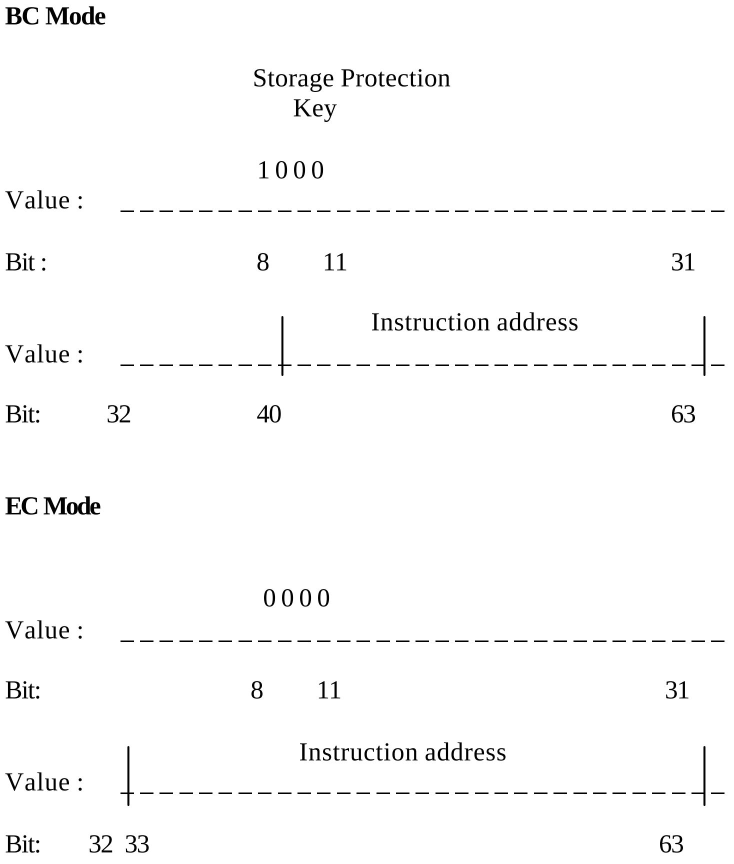


Fig.1.22 Program Status Word

(b) Segment Protection

Figure 1.23 shows another type of storage protection called Segment Protection. MVS/370 virtual storage is divided into 64Kb segments. MVS/XA and MVS/ESA Virtual Storage is divided into one-megabyte segments. If the program or system has not obtained storage from the segment, or if the segment is not included as part of the common area, then that entire 64-Kb segment is unavailable to the program. A segment may be marked "read only" to prevent either application programs or systems programs from modifying areas that should be read only.

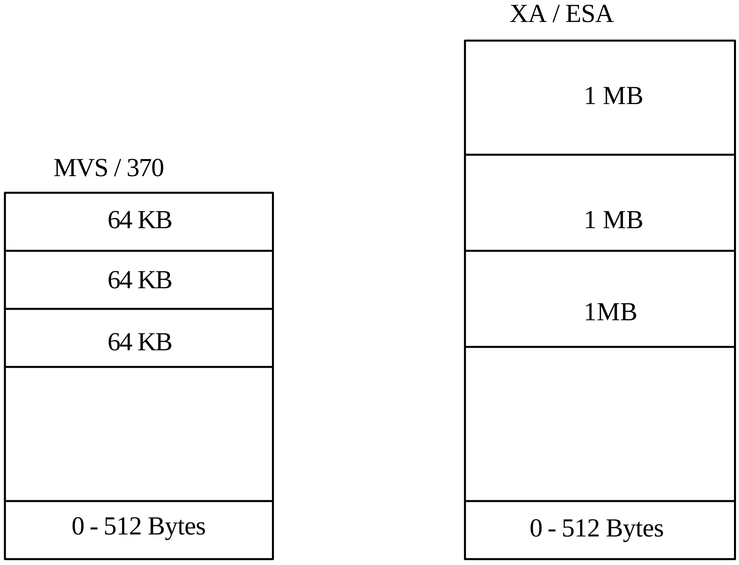


Fig. 1.23 Segment Protection and LAP

(c) Low Address Protection (LAP)

Low Address Protection is designed to protect Central Storage address from x'00000200' (the first 512 bytes) from damage. These addresses are important because this area is used by the hardware to control timer interrupts. The hardware stores the address of the program interrupted and loads the address of the MVS routines that will handle the interrupt.

(d) Page Protection of System Areas

Page protection is implemented in MVS/XA and MVS/ESA to protect system areas that are not protected by segment protected by low address protection. Page protection is in action to storage protection keys. Page correction in MVS/XA and MVS/ESA helps prevent accidental or intentional changing of Central Storage pages by marking specific pages as being "read-only".

1.6 ESCON Architecture

The ESA/390 architecture includes IBM Enterprise Systems Connection (ESCON) architecture. ESCON is a set of interrelated hardware and software products and services that provide fiber optic cabling, transmission and reception. It provides dynamic connectivity using a switched point-to-point topology. ESA/390 Processors can be ordered with groups of ESCON channels as well as with groups of traditional channels now called parallel channels. Certain control units - 3174, 3490, and 3990 - can be made ESCON - capable so that they can be directly connected into ESCON channels or connected via a switch known as ESCON Director (ESCD). Other control units continue to connect through parallel channels. The IBM 9034 ESCON converter allows attachment of Non - ESCON control units to ESCON channels. The IBM 9035 ESCON converter allows attachment of ESCON - capable 390 control units to parallel channels.

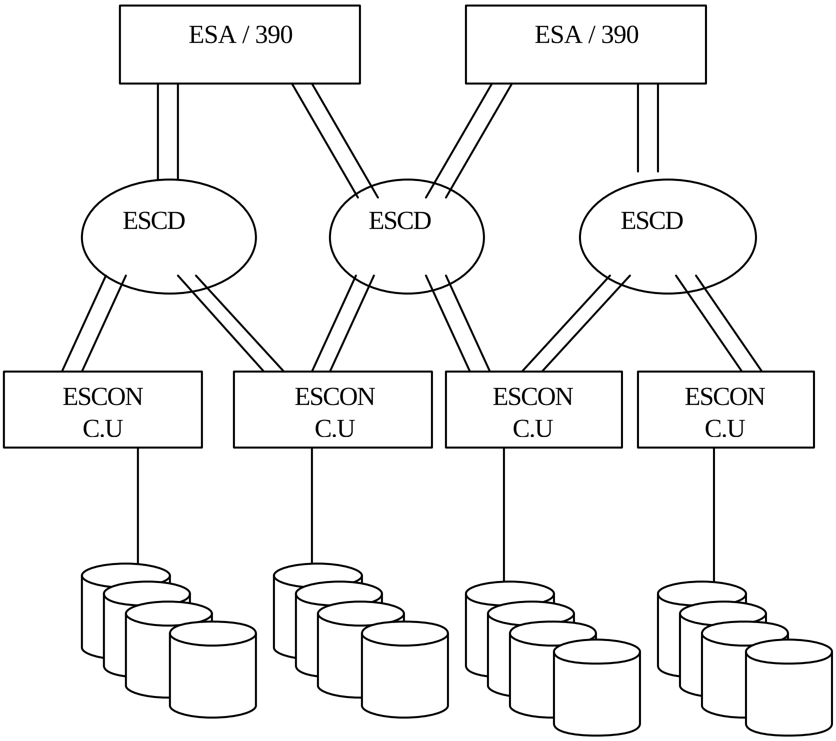


Fig. 1.24 ESCON Architecture

1.7. SYSPLEX

SYSPLEX stands for "Systems Complex." MVS/ESA version 4 introduces the concept of SYSPLEX.

MVS/ESA provides substantial systems management enhancements for coordinating and controlling data processing across multiple MVS systems. Each MVS system may be running on a separate processor, or in a PR/SM partition, or as a guest system under VM. A software component of MVS known as cross - system coupling facility (XCF) that provides the communication services between MVS systems. Various MVS functions will begin to use XCF in multisystem environments - Global Resource Serialization (GRS), console operations, and TSO/E.

1.8 MVS Software Services

The MVS operating system provides a number of services for application programs. Services that are provided are for the following areas:

a) Dynamic Address Translation (DAT): MVS constructs the control blocks in virtual storage required by DAT.

b) Multiprocessing: MVS is designed to operate on a Processor Complex, which has multiple CPUs. Many of the services provided by MVS support multiprocessing (more than one instruction being executed simultaneously).

c) Multiprogramming: MVS supports Multiprogramming (many tasks appearing to use resources of the Processor Complex) - both tightly coupled and loosely coupled.

d) ENQUE and Locking: Multiprogramming and multiprocessing environments require solutions to communications problems with two or more tasks operating at the same time. ENQUE and locking services solve these problems.

e) Timing: MVS provides timing services to the applications that need alarm clock to tell them that the current time and "wake them up" at times in the future.

f) System Operator Communication: Application needs to notify the MVS operator of information. The MVS system itself is one of the largest users of operator communication.

g) Address Space Management: MVS manages task creation, control, and its interruption processing using dispatcher services. The term "dispatch" is used to describe MVS operating system modules turning control of the CPU to a task.

h) Batch Job management and spooled input and output: As data center users submit work to the MVS system, the JES subsystem controls these resources.

i) Data set management and access methods: There are several types of data sets in the MVS architecture. The MVS Access Method selected to build the data set determines the type of data set.

j) Program management: Programs written by IBM and supplied as part of the IBM operating system provide all of these services. Additional Programs are supplied by the data center. Applications have their own common "program libraries".

k) Communication software and hardware: The task of providing and servicing terminal access to MVS is provided by the communications software and hardware supported under MVS.

1.9 Subsystems and Facilities

A *subsystem* is a software product that operates in its own address space under the control of MVS. Within an address space, a subsystem may provide services that duplicate services that duplicate the multiprogramming facilities provided by MVS. Every MVS installation has a *primary subsystem* that is the job entry subsystem, JES2 or JES3. It maintains its own queues.

1.9.1 TSO and ISPF

TSO (Time Sharing Option) is an MVS component that lets terminal users access MVS facilities. TSO does this by treating each terminal user as a job. When you logon to TSO, TSO creates a JCL stream and submits to JES2/JES3 for processing. Each TSO user is given a unique address space and can allocate data sets and invoke programs just as a batch job can. The TSO commands reside in SYS1.CMDLIB and are located into the private area of the TSO user's address space as required. The Terminal Monitor Program (TMP) is a program that accepts and interprets commands and causes an appropriate command processor to be scheduled and executed.

The TSO environment has both MVS and TSO user address spaces and MVS and TSO user data sets. The system address spaces are Master Scheduler, JES, BATCH, VTAM and TSO. TSO user spaces are created as users LOGON. The system data sets are SYS1.PARMLIB, SYS1.VTAMLST and SYS1.PROCLIB. The TSO data sets are SYS1.UADS which contains TSO user attributes, SYS1.CMDLIB which contains TSO command processor routines, service routines and utility programs, SYS1.BRODCAST which contains messages for the terminal utility programs, SYS1.HLP which contains information that is displayed whenever a TSO user enters the HELP command.

The TSO environment usually includes the ISPF (Integrated System Productivity Facility) and the Program Development Facility (PDF). The ISPF, a dialog manager, which runs under the control of TSO, provides a powerful and comprehensive environment that includes a full-screen text editor and facilities to manage background job processing. The modules and panels that make up ISPF and ISPF/PDF are usually copied to SYS1.LPALIB so that they will be mapped in the common area of every address space. We shall discuss in detail how to use ISPF's full-screen editor to create and maintain job streams and to use ISPF's background job management facilities to submit those jobs, monitor their progress, and display their output.

1.9.2 Storage Management Subsystem (SMS)

The Storage Management Subsystem, commonly referred to as SMS, is an IBM product that became available with MVS/ESA. The purpose of SMS is to provide the user with system-managed data, intended to simplify user interface for allocating and maintaining data, provide the user with a logical view of data, separated from the actual physical device characteristics and to provide centralized control of disk storage.

Placing the data set under SMS control has certain benefits. Disk allocation for new data sets is directed to the most suitable group of disk volumes under centralized disk-storage administration control. The user does not determine which volumes will be used whereas SMS does this. The management of a data set after it is created of archive, retention etc., is also under centralized administration control. The JCL is also simplified.

1.9.3 Job Entry Subsystem (JES)

The Job Entry Subsystem (JES) is the part of MVS that manages batch jobs and SYSOUT under MVS. There are two flavors. The first is JES2 and the second is JES3. The Job Entry Subsystem will be discussed in detail in chapter 2.

1.9.4 Teleprocessing

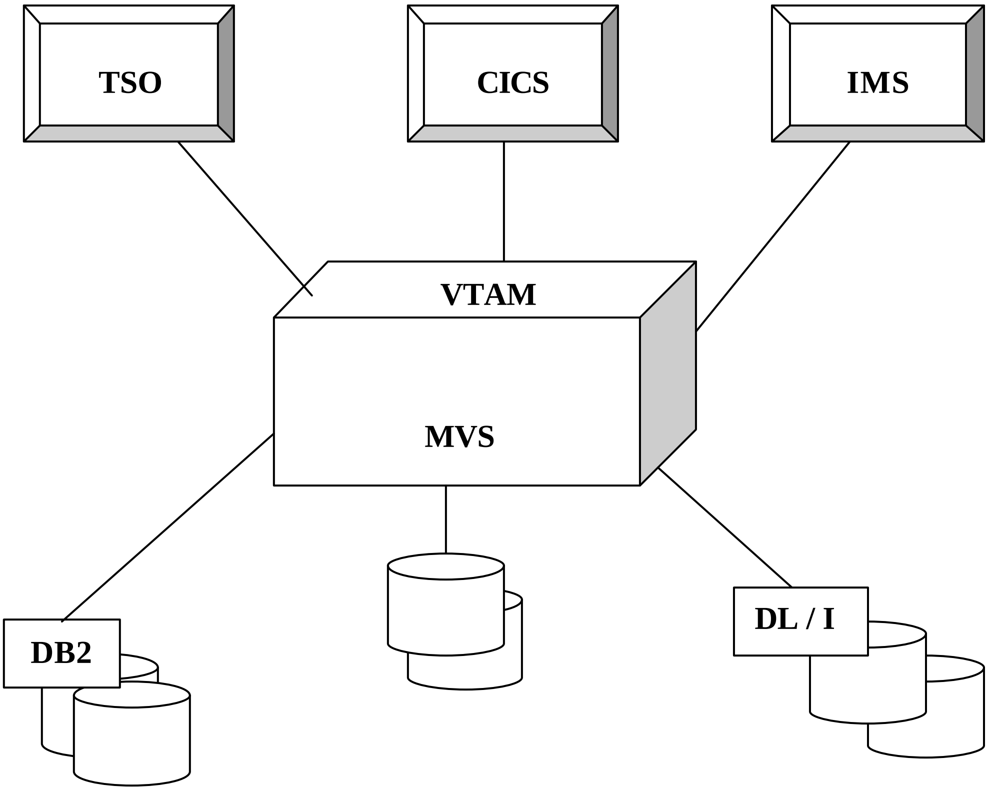


Fig.1.25 Teleprocessing System

The subsystems such as TSO, CICS and IMS interface with MVS in a teleprocessing environment and the subsystems such as IMS and DB2 Interface with MVS in a data base environment. These subsystems use many of the facilities of MVS, such as multiple address spaces and Cross Memory Services.

Teleprocessing hardware components

The component in a teleprocessing network transfers information between the application programs in the Central Storage and the devices in the network.

The communication controller routes data through the physical network. The line provides path to transmit data between modems. The terminal control unit is the interface between the modem and the attached workstation devices.

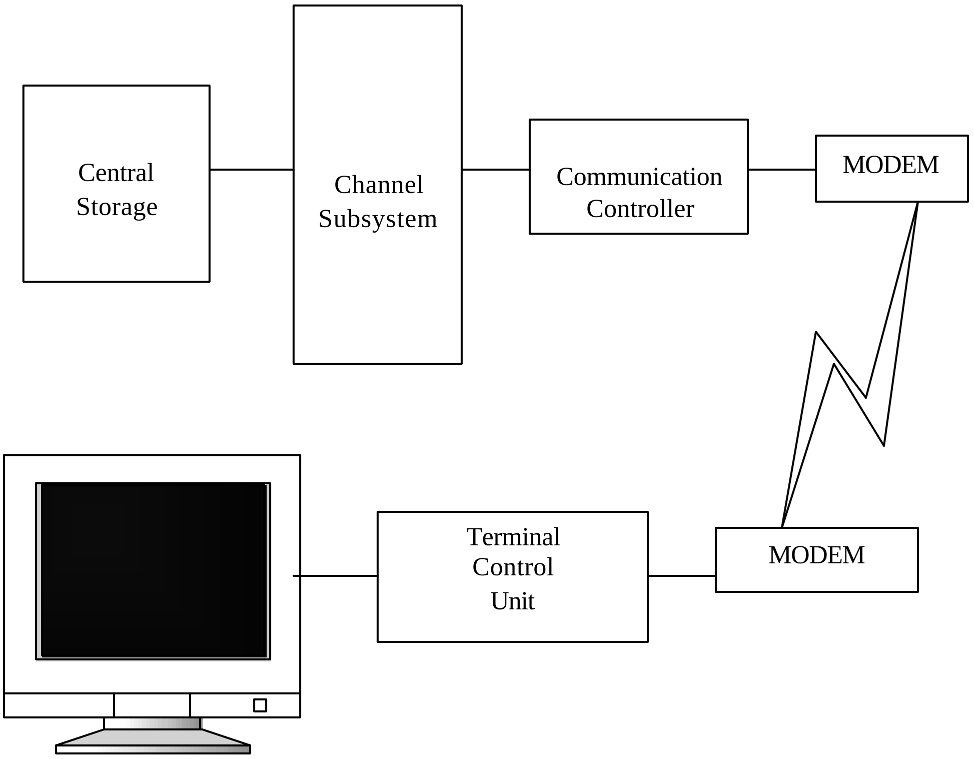


Fig.1.26 Teleprocessing Hardware Components

Teleprocessing software

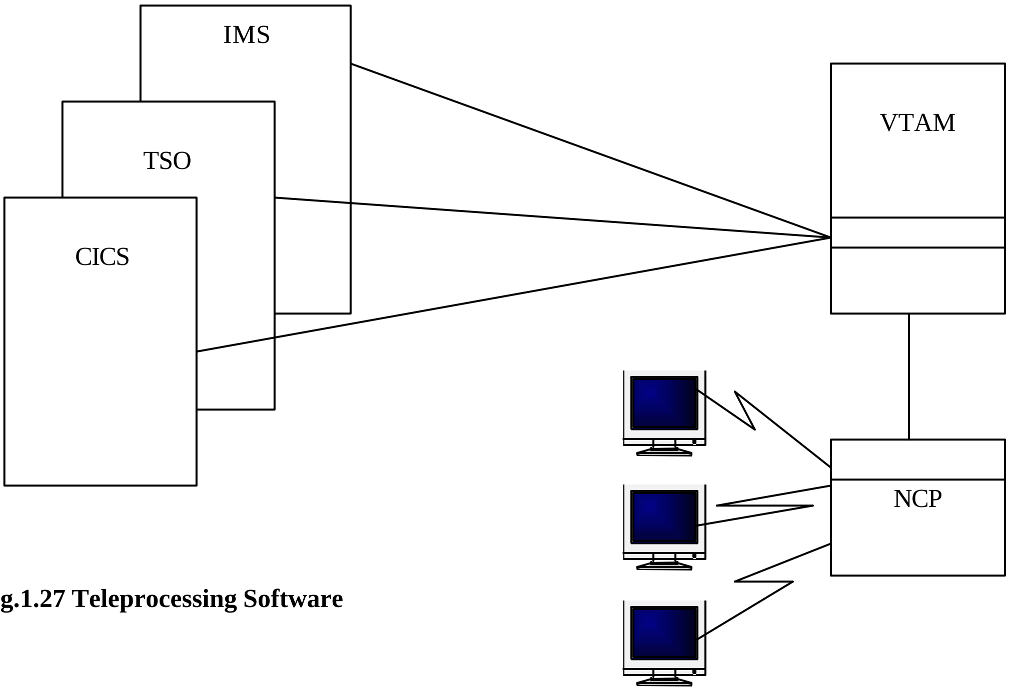


Fig.1.27 Teleprocessing Software

The software components of a teleprocessing network control how the information is transferred between the applications and the devices. Terminal users communicate with CICS, IMS and TSO via the network control program, which resides in a communications controller and the virtual telecommunications access method (VTAM) which resides in the host system.

Telecommunication access methods

For a system to support any terminal devices, local or remote, it must include a telecommunications (TC) access method.

Basic Telecommunication Method (BTAM)

The basic telecommunication method was one of the first MVT access methods for terminal access. The characteristics of BTAM are as follows.

- Device and line controls are combined. Terminals are treated as single entity.
- Each application program owns and manages its own lines and terminals.
- Neither lines nor terminals can be shared with another address space or application.
- Each program reads a message from a device into a pre-designed buffer.
- The application program issues a WRITE for output.
- No message editing is done by BTAM.
- Devices and data lines are referred to by the physical address generated by the MVS system.
- All devices and line control information must be a part of the buffer, and it must be unique to the destination terminal.

Telecommunication Access Method (TCAM)

IBM introduced TCAM as a replacement for BTAM. The improved function enabled customer network to grow. TACM handles many types of terminals. TCAM has its own control program for queuing and traffic scheduling.

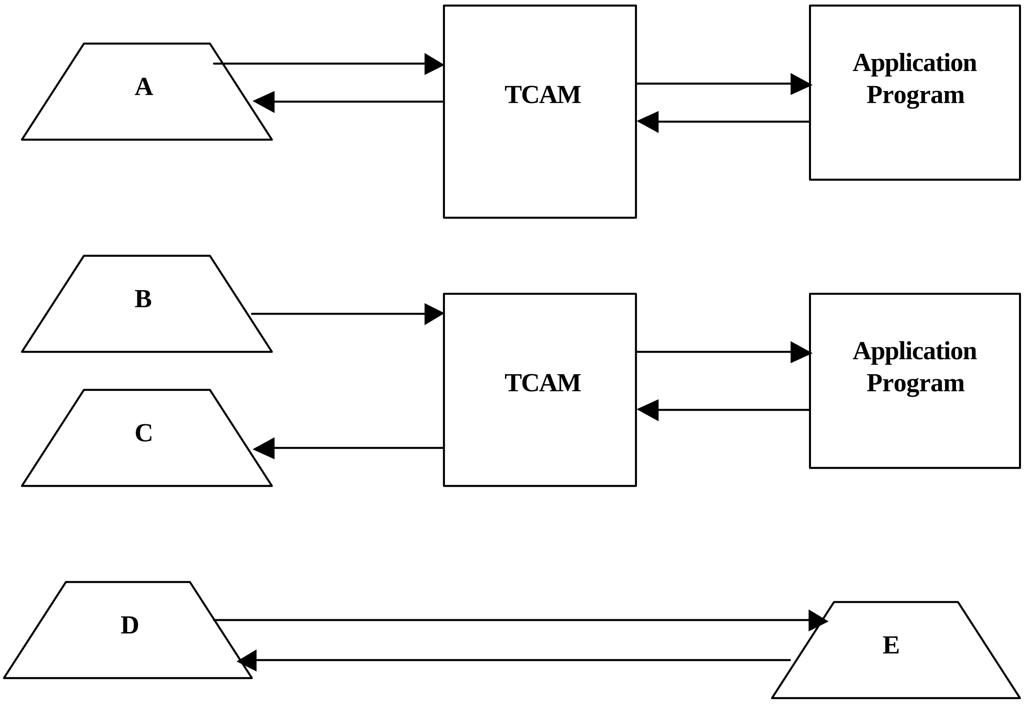


Fig. 1.28 Telecommunication Access Method

Fig.1.28 shows three general types of communication in a TCAM environment. The following are the characteristics of the TACM:

- Allows line and device sharing between applications.
- Employs a set of message queues between application and communications network.
- Permits message switching.
- The TCAM Message Control Program (MCP) performs network control functions.
- Supports application program's use of high level GET and PUT macros.

Virtual Telecommunication Access Method

MVS uses the most powerful access method viz., *Virtual Telecommunications Access Method* (VTAM), a part of the comprehensive telecommunications product called SNA, which stands for *System Network Architecture*.

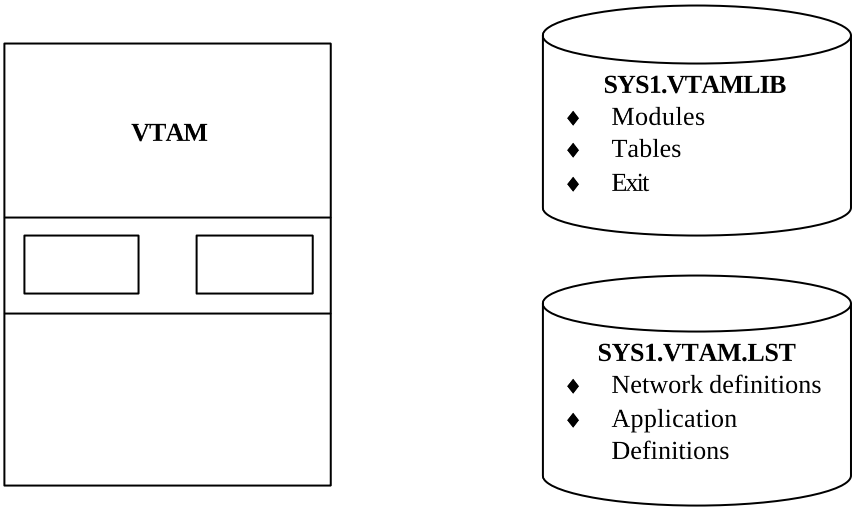


Fig.1.29 Virtual Telecommunication Access Method

VTAM is considered to be a subsystem because it runs in its own address space. VTAM is able to provide centralized control over all of the terminal devices attached to an MVS system. Each VTAM terminal device is allocated to the VTAM address spaces, communicate with those terminal devices indirectly. They issue requests to VTAM, which in turn services the request for the appropriate terminal.

There are two required data sets that control how VTAM operates: SYS1.VTAMLIB contains the VTAM load modules and user defined tables and exit routines. SYS1.VTAMLST contains the definition statement and start options for the VTAM.VTAM applications that must be defined to VTAM with an APPL statement. This allows network resources to communicate with that application through VTAM. An APPL definition must be coded for each application program.

1.9.5 CICS

CICS (Customer Information Control System) is a transaction subsystem that handles interactions between terminal users and online application programs. CICS is a subsystem of VTAM, and so any CICS system must be defined to VTAM. A connection between the user terminal and CICS is automatically established when either CICS or VTAM is started. End users identify themselves to CICS by signing on to begin an online session.

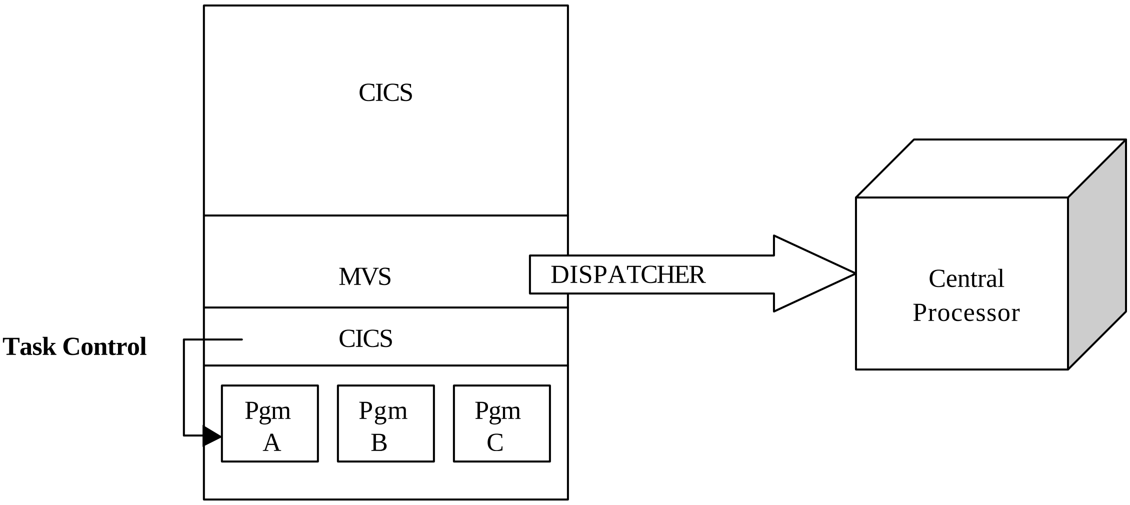


Fig. 1.30 CICS

CICS is a single task to MVS. However CICS behaves as an operating system within an operating system. CICS task control determines which CICS application will execute for this MVS task. The CICS address space is usually given favored status within the MVS system i.e., it may be non-swappable or it may be given a dispatching priority higher than most other work in the system.

CICS has the multiregion operation feature. This allows multiple CICS address spaces to communicate with each other. For example, a terminal that is logged on to CICS2 can access a program owned by CICS1.. Both CICS address spaces could be dispatched by MVS at the same time on two separate processors.

1.9.6 IMS

IMS (Information Management System) has two components: DL/I and Data Communications. The DL/I component of IMS allows users set up and maintain complex hierarchical databases that can be processed by application programs run as batch jobs. If the optional Data Communications component of IMS (IMS DC) is used, interactive application programs can be coded that use IMS databases and communicate with terminals.

Like CICS, IMS DC implements its own multiprogramming that is transparent to MVS. IMS DC multiprogramming is more like MVS multiprogramming than CICS multiprogramming, however. The IMS control region in its own address space schedules application programs for execution in dependant regions (also in separate address spaces). The control region also manages communication between the application programs, databases, and terminals.

1.9.7 DB2

A program with embedded SQL statements requires special processing to prepare it for execution. The program development process includes the steps:

- Pre-compilation
- Compilation or assembly
- Link editing
- Binding
- Execution

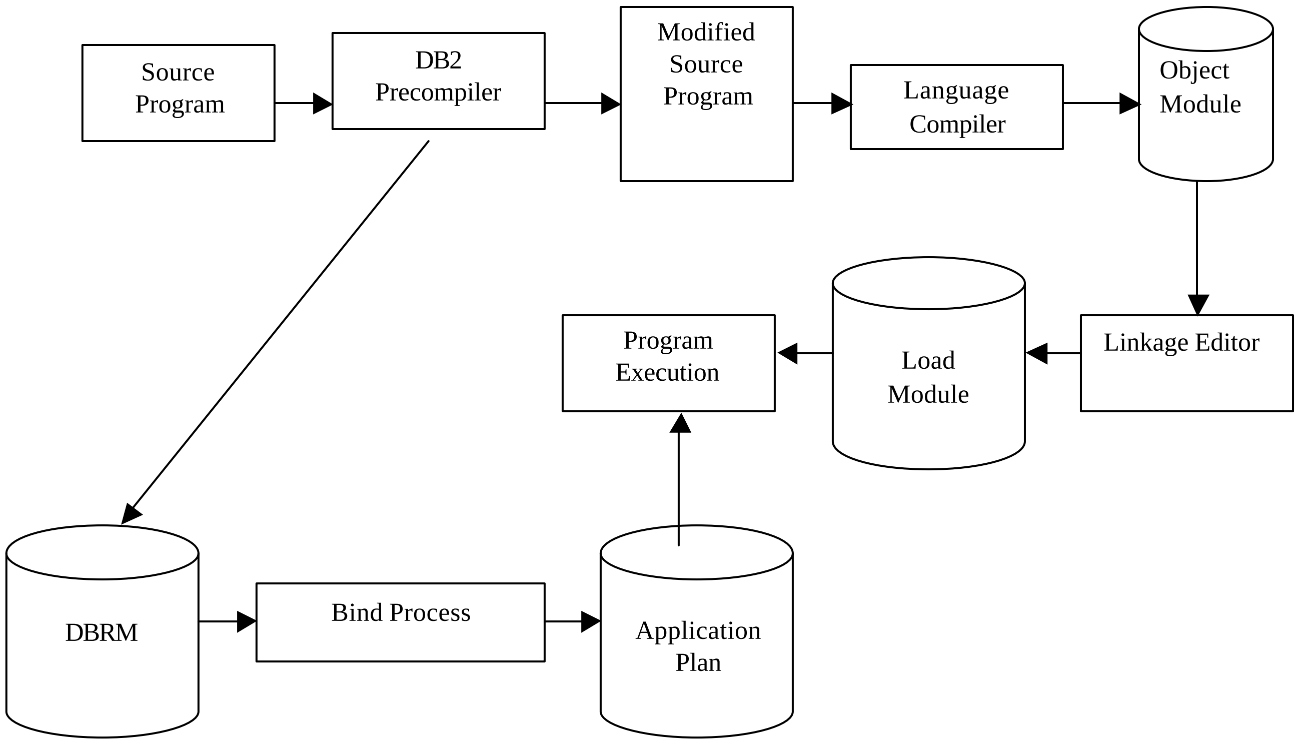


Fig. 1.31Components of DB2

Pre-compiler

Before a program can be compiled or assembled, it must be processed by the DB2 pre-compiler. The pre-compiler does the following functions:

- Converts SQL statements into host-language statements.
- Validates SQL statements.
- Detects SQL statement syntax errors.
- Produce a data base request module.

Compilation and Link Editing

A source program is compiled and assembled to produce an object module. After successful compilation, the object module is link-edited from the load module. DB2 can have multiple source modules that are each compiled and assembled separately. The individual object modules are then combined in a single link-edit step. Individual DB2 program modules communicate with each other using conventional linkage techniques.

Binding

Before a program can be executed, a process called binding must be performed. Binding establishes a linkage between the application program and the DB2 data it accesses. Binding process involves only the DBRM produced by pre-compiler.
Process performed by binding are as follows:

- Verify SQL statements

- Verify authority
- Select access paths
- Prepare an application plan

The process can be run immediately after pre-compilation itself since it works on only the DBRM.

Program Execution

After successful binding, the program is ready for execution. The program can be executed many times using the application plan. When a program is executed, DB2 checks the application plan, if there are any changes it rebinds it.

Authorization

No special authorizations are required to code, pre-compile, compile or assemble, or link-edit a DB2 program. However the user who binds or executes the program must have been granted appropriate authorizations to perform these functions. The user must have appropriate table privileges, based on the type of operations performed by the program. These can include authorization to select data, to insert, update, or delete data, or to create new tables. A user, performing bind must have specific authorization to perform a bind for the program.

1.9.8 RACF

A very important consideration in any MVS installation is to maintain security and so unauthorized users cannot access data. To provide security, most installations use a comprehensive security package called *Resource Access Control Facility* (RACF). RACF identifies both users and resources, such as data sets. When a user attempts to access a resource, RACF ensures that the user has the correct authority. RACF is not a subsystem, but rather a set of routines stored in the PLPA that are invoked by a user's address space whenever needed.

1.9.9 SMF

System Management Facility (SMF) monitors jobs as they execute and records information such as the amount of CPU time used, the amount of DASD I/O that was performed, the number of print records that were created etc. This information is recorded in special data sets so that it can be used as the basis for billing.

1.9.10 Language Translators, Linkage editor, and the Loader

Language translators are the programs that convert source programs into *object modules*. The purpose of the language translators is to reduce the programming time required to prepare a working object program. Hence the language translators print all error listings to help the programmer correct clerical errors. Also they provide debugging tools to help the programmer test the program. A language translator viz., the assembler, is supplied as a part of MVS. Other language translators, like the COBOL, PL/I, FORTRAN IV,C and Java compilers, are separate products.

The *linkage editor* program which is supplied as a part of MVS, converts object modules into executable programs called *load modules* that can be loaded into virtual storage and executed. The *loader* program is like the linkage editor except that instead of creating a permanent load module, the loader program creates a temporary load module, executes it, and then deletes it.

1.9.11 Utility Programs

Some routine processing functions such as copying files and sorting records within a file are common to computer installations. MVS provides a set of general-purpose *utility programs* or *utilities* to perform those functions. A utility program can be supplied with parameters to specify the exact processing that should be done. A basic MVS utility program viz., IEBGENER produces a copy of a data set.

1.9.12 CLISTs and REXX

The CLIST (Command List) language is an interpretive language. It allows users as well as programmers to work more efficiently with TSO. When using TSO, scripts called CLISTs can be written that perform a given task or group of tasks. To execute a single or multiple set of commands or programs, a CLIST can be invoked to perform those tasks.

Most CLISTs consist of a series of TSO commands. When such a CLIST is invoked, it issues the TSO commands in sequence. In addition to issuing TSO commands, CLISTs can perform more complex programming tasks. The CLIST language includes the programming tools that are needed to write extensive, structured applications. CLISTs can perform a variety of complex tasks, from displaying a series of full-screen panels to managing programs written in other languages.

Like programs in other high-level interpretive languages, CLISTs are easy to write and test. There is no need for compilation and link editing. A CLIST is tested by executing it, correcting any errors and then re-executing it.

Other than CLIST there is one more command language available on TSO called the REXX. The UNIX counterparts to CLISTs are called shell scripts. The CLISTs provide an extensive set of arithmetic and logical operators for processing numeric data, string handling functions for processing character data etc. CLIST allows us to structure the programs, perform I/O, define and modify variables, and handle errors and attention interrupts.

CONTROL statements are used to debug the CLISTs. CLIST errors are diagnosed using the CONTROL statement options. The diagnostic options - LIST, CONLIST, SYMLIST and MSG are used within a CLIST to display its statements, commands, and informational messages at the terminal during execution of the CLIST. The information that is displayed assists us in finding statements that contain errors. A sample CONTROL statement is

```
CONTROL  LIST  CONLIST  SYMLIST  MSG
```

CONTROL statements can be placed at the beginning of the CLIST or in any line within the CLIST that is to be tested or debugged. Each new occurrence of the CONTROL statement overrides any previous CONTROL statements. To turn off the diagnostic options we type

```
CONTROL  NOLIST  NOCONLIST  NOSYMLIST  NOMSG
```

The options that can be placed on the CONTROL statement have the following effects:

SYMLIST The CLIST will display each TSO command, subcommand, or CLIST statement at the terminal before scanning it for symbolic substitution.

LIST The CLIST will display each TSO command or subcommand at the terminal after symbolic substitution has taken place, but before execution.

CONLIST The CLIST will display each CLIST statement at the terminal after symbolic substitution has taken place, but before execution.

MSG The CLIST displays informational messages at the terminal.

1.9.13 Overview of MVS System Components and Facilities

Components of the Common Area

The common area, both above and below the 16-Mb line, consists of the following portions:

Nucleus: This is permanently resident portion of the operating system code. These modules are loaded from the system data set SYS1.NUCLEUS which resides on the system residence (SYSRES) volume and whose location is known to the Initial Program Load (IPL) program. These modules are loaded below and above the 16-Mb line (Nucleus and Extended Nucleus).

Link Pack Area (LPA): These are operating system modules, which are reentrant, i.e. they are not modified. These modules are loaded into Central Storage at IPL time and are forcibly paged out. These modules are loaded into Central Storage whenever required and as they are not modified, there is no need to ever page them out. Within the LPA, there is the LPA below the 16 Mb line and the Extended LPA (ELPA) above the 16 Mb line. Within the LPA and the ELPA, another classification is Pageable LPA (PLPA), Fixed LPA (FLPA, which is fixed in Central Storage) and the Modified LPA (MLPA, which is primarily used for test modified LPA modules). Hence, this results in a total of six categories, PLPA, EPLPA, FLPA, EFLPA, MLPA, EMPLA. All these modules are located in the system data set SYS1.LPALIB.

System Queue area (SQA and ESQA): This area contains system wide control blocks and queues.

Common System Area (SQA and ESQA): This area contains data that can be accessed by more than one address space. In addition to these components in the common area, some system components that are particular to each address space reside in the private areas of each address space. These are the Scheduler Work Area (SWA and ESWA), that contains information for use by the Master Scheduler and the Local System Queue Area (LSQA and ELSQA), which contains address space specific control blocks.

Real storage Manager address space (RASP): This address space, which is present only in MVS/ESA, handles the control blocks necessary for Data space processing.

System Trace: This address space is used for the tracing of system activity.

Global Resource Serialization (GRS): This address space contains control blocks to serialize access to shared resources.

Dump services (DUMPSRV): This address space contains buffer for storage of system generated dumps.

Catalog (CAS): This address space handles the task of catalog management.

Console address space: This address space is the interface between user and the system consoles.

Allocation (ALLOCAS): This address space contains tables to track device allocation.

System Management facility (SMF): In this address space, there is code for recording of accounting and statistical information.

Lnklst Look Aside (LLA): In MVS/XA, this address space contains in virtual storage the directory entries of SYS1.LINKLIB and its concatenations. Hence with the use of the LLA, the search time is reduced. In MVS/ESA, this address space is associated with a data space, which contains frequently referenced programs.

Virtual Look-aside Facility (VLF): This address space, which is there only in MVS/ESA, has associated data spaces in which frequently used programs, CLISTS and catalog records are stored for fast retrieval.

Job Entry Subsystem (JES): This address space contains code for the Job Entry Subsystem (JES), which is the agent responsible for the management of background jobs on the system. There are also a number of batch address spaces or initiators, in which the actual jobs run.

Virtual I/O (VIO): This address space is used for temporary data sets. It provides a one-block window in virtual storage for temporary data sets, thereby increasing performance.

Data-In-Virtual (DIV): This address space which is present in MVS/ESA only, issued by DB2. 4-Kb blocks of data are read into virtual storage and accessed directly.

Storage Management Subsystem (SMS): In MVS/ESA, there is an optional product called SMS, which simplifies to a great extent the management of data on a MVS system. This product runs in a separate address space.

Time Sharing Option (TSO): This product is one of the products used for online tasks on MVS.

Virtual Telecommunications Access Method (VTAM): This address space is responsible for communication between terminals and the host.

Customer Information Control System (CICS): This product is another one, which is used for online tasks on MVS.

1.9.14 JOB CONTROL LANGUAGE (JCL)

MVS is primarily a batch oriented system designed to give the user maximum throughput. JCL is a language, through which all of the parameters required for execution of a program are given. Any work that you want to do on MVS is to be submitted.

```
// JOBA       JOB  .................           ————→ JOB CARD

// STEP1      EXEC PGM=A ....................  ————→ FIRST STEP

// DD1        DD DSN = ..........................  ——→ DD CARD

                    ............................
                    ............................
                    ............................

//STEPN       EXEC  PGM = Z .................   ——→ Nth  STEP
```

JOB card-states that it is the start of the job. In the job card, all job-related parameters are described. EXEC card-states, which is the program to be executed. All parameters related to this step will be coded here. EXEC is also known as a step. DD card - If program needs to READ/WRITE any file, the corresponding DD card is needed, specifying the parameters for that file.

A job consists of more than one step. Each of the steps is executed sequentially one by one, when job is submitted. A submitted job is interpreted by Job Entry Subsystem (JES), and if there is no error in your JCL, it is placed in Job Queue, Element (JQE), for execution. Depending upon priority, and availability of an initiator, your job will be executed in the initiator's address space. Since each step is executed sequentially, it forms a task, and it is also a unit of work.

Review Questions

1. What are the basic components of a basic computer system?
2.Differentiate between Minicomputer, Microcomputer and Mainframe computer.
3.Differentiate between Scientific computers, Decimal computers and Character computers.
4.Differentiate between Batch Processing and Interactive Processing.
5. Differentiate between PCP and MVT.
6.State the disadvantages of PCP.
7.Differentiate between OS/VS1 and Single Virtual Storage.
8. Differentiate between SVS and MVS.
9. State the advantages of MVS.
10.What is the significance of the 16-MB line?
11.Differentiate between MVS and MVS/ESA.
12. Differentiate between Address space and Data space.
13. Differentiate between Data space and Hiperspace.
14.What are the characteristics of Mainframe operating system?
15. What is "Spooling"?
16. Define the terms " Multiprogramming " and "time sharing".
17. What are the components of Processor Complex in System/370 architecture?
18. Distinguish between Uniprocessing and Multiprocessing.
19. What are partitionable Multiprocessors?
20.What are the types of DSAD page space?
21. Distinguish between Page data sets and Swap data sets.
22. What is the significance of Segment Protection?
23. What do you mean by Low Address Protection?
24. How is Page Protection of System Areas taken care of in MVS/XA and MVS/ESA?
25.What is ESCON? State its advantage.
26. What is SYSPLEX? State its advantage.
27.Distinguish between multiprocessing and multiprogramming.
28.What do you mean by a subsystem?
29. What are the advantages of SMS?
30.What are the advantages of JES?
31. What are the types of JES?
32. What do you understand by TSO and ISPF?
33. Compare and contrast the types of Telecommunication Access Methods.
34.What are the types of Telecommunication Access methods.
35. What are the two data sets required for VTAM to operate?
36.How does CICS help MVS?
37. What are the components of IMS?
38. What are the different components of DB2?
39.What do you mean by "Binding" in DB2?
40.What is the advantage of RACF?
41. What are the various functions of SMF?
42.What are language translators?
43. What is the function of a linkage editor?
44. What are Utility Programs?
45. Name a Utility Program.
46. What is the function of IEBGENER?
47.What are REXX and CLISTs?
48. What are the different options that are available on CLISTs?
49.What is the use of the Control Statements
50. State the usefulness of JCL.

2. Introduction to MVS Internals

2.1 Introduction

2.1.1 Real and Virtual Storage

Real Storage consists of main, expanded and auxiliary storage. The CPU stores in any of these three as pages. Main storage operates at the speed of CPU itself and so access is faster. Main storage is of specific size. Expanded storage is an extension of main storage. DASD corresponds to the expanded storage. The data should be in main storage for execution.

Types of Real Storage

Storage type	Speed of Retrieval	Cost	Size	Components
Main	Fastest	Most expensive	Big	Solid-state devices
Expanded	Fast	Less Expensive	Bigger	Solid-state devices
Auxiliary	Slow	Least Expensive	Biggest	DASDs

Understanding how the MVS/ESA operating system manipulates data is essential to understanding how to improve the efficiency of the programs. The primary benefit of ESA is its ability to make the amount of I/O required to access data. Knowing how the processor stores data in the storage areas allows us to use it to its fullest potential. Paging and buffering are the activities that the processor performs to make data available to the program.

Virtual Storage is the concept used to create the appearance that an entire program is in main storage when in fact only parts of it are. Other parts are in auxiliary and expanded storage.

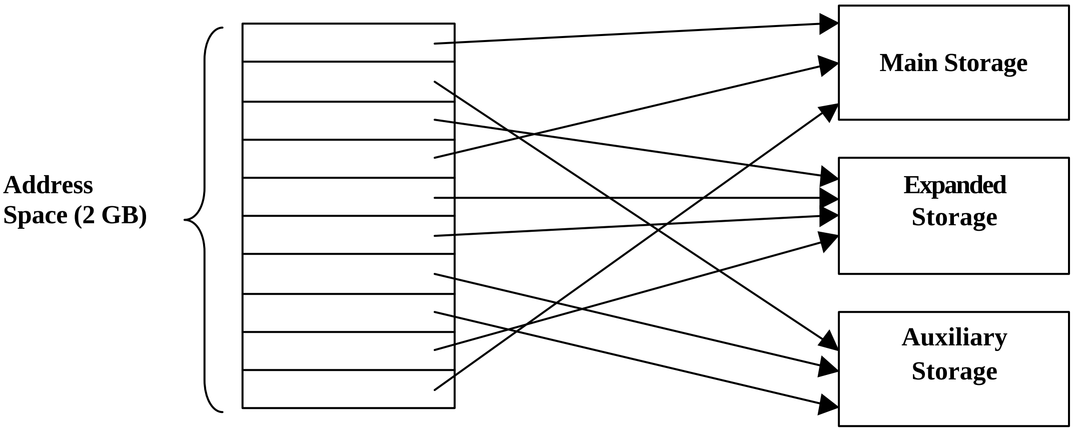


Fig.2.1Virtual Storage
(Each partition in the Address space is 4K size)

Virtual Storage allows us to execute programs that are larger than the space available in main storage. Virtual Storage is what the application programmer can access. Each MVS task - address space - has its own copy of Virtual Storage.

2.1.2 Address Spaces

An *address space* is a virtual address if it does not correspond to real address. This range of addresses is a contiguous area that is available for the program to access program instructions, data areas, I/O buffers, etc. With MVS, each job has its own address space, and each address space could be as large as 16 MB or 2 GB (MVS/XA and MVS/ESA). As MVS starts up, it creates a single address space called the MASTER address space and several "Operating System" address spaces. Some of the address spaces are CATALOG (to manage VSAM catalog requests), CONSOLE (to manage communication to OS consoles in the computer room), and GRS (to manage queuing resources across address spaces and across multiple computers).

2.1.3 Virtual Storage Sizes

Virtual Storage may be the same size as, or different from, the size of Central Storage. Virtual address translation has already been covered in the CHSSC course under memory management. The reader is advised to refer to the references listed in the CHSSC course material. The concept of Virtual address translation is felt important as it describes the importance of the region parameters in JCL. The virtual storage is basically equal to the addressability of the machine.

2.2 Multiple Virtual Storage (MVS)

Multiple Virtual Storage (MVS) simulates more storage, but it also uses real storage to simulate several address spaces, each of which is independent of the others. The real storage and areas of DASD storage called *page data sets* are used in combination to simulate several virtual storage address spaces. To refer to a particular byte of virtual storage under MVS, the address, which identifies a specific byte of storage within a 16 Mb or 2 GB address space and the address space to which the address applies must be known.

MVS can simulate unlimited number of address spaces. Though an MVS system can support more than one address space at a time, the CPU can access only one of them at a time. Each background job or user is given its own address space. So each job or user can access up to 16 MB or 2 GB of virtual storage independently of any other job or user to another, MVS transfers control of the CPU to the other job's or user's address space. Then the CPU can access instructions and data in that address space until MVS is ready to pass control to a job or user in yet another address space.

2.2.1 Paging

Paging is the physical movement of single 4K pieces of information from the main storage to expanded or auxiliary storage and back again. MVS divides virtual storage into 4K (4096 bytes)sections called *pages*. Data is transferred between real and DASD storage one page at a time. Hence real storage is divided into 4K sections called *page frames*, each of which can hold one page of virtual storage. Similarly, the DASD area used for virtual storage, called a page data set, is divided into 4K (4096 bytes blocks) *page slots*, each of which holds one page of virtual storage.

When a program refers to a storage location that is not in real storage, a *page fault* occurs. Then a *page-in* is said to occur, whereby MVS locates the page that contains the needed data on DASD and transfers it into real storage. Sometimes the new page can overlay data in a real storage page frame. Or the data in a page frame has to be moved to a page data set to produce space for the new page. This is called *page-out*.

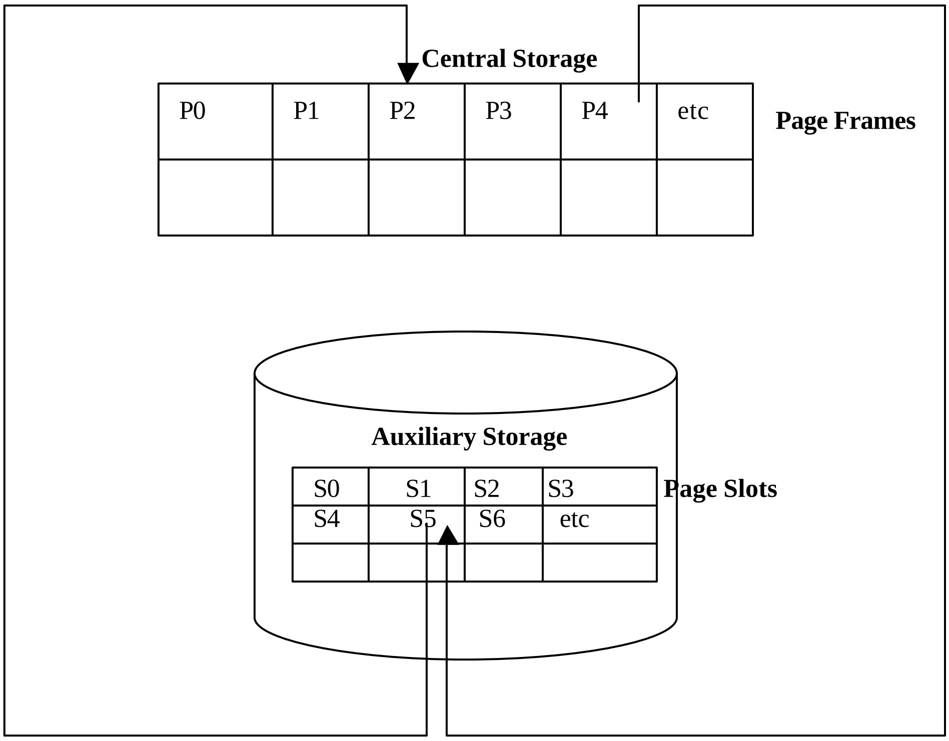


Fig. 2.2 Central Storage Vs Virtual Storage

If	**then**
a 4K piece (page) of data is requested	the system looks in main storage to satisfy the request.
requested data is in main storage	the request is satisfied.
requested data is not in main storage	a page fault occurs and the system looks in expanded storage for the data.
requested data is in expanded storage	it is paged to main storage.
requested data is not in main storage or expanded storage	a page fault occurs and the system pages data into main storage from auxiliary storage.

2.2.2 Expanded Storage

System/390 or ESA processors now include a special type of memory called *expanded storage*. Expanded storage improves the efficiency of virtual storage operations by acting as a large buffer between real storage and the page data sets. So, when a virtual storage page must be paged out, the processor moves the page's contents to expanded storage. This transfer occurs at CPU speeds rather than at DASD speeds, so the operation is instantaneous.

2.2.3 Paging Hierarchy

Programs and data are moved between main, expanded and auxiliary storage based on frequency of use. Programs and data must be in main storage to execute.

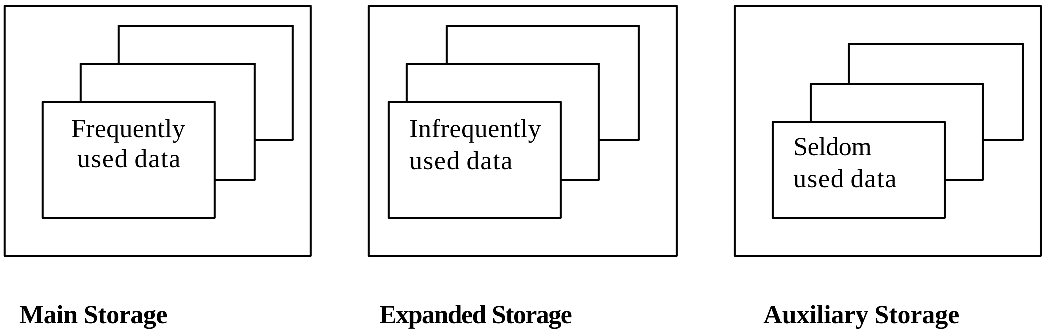


Fig. 2.3 Paging hierarchy

Paging allows many jobs to run at the same time and improves performance for less cost. Address space is created by system for user's program and data. Address space is being used by OS program code and OS data elements. Only a portion of it can be used for application data.

2.2.4 Swapping

Swapping is the physical movement of all 4K pieces of information for a single job from main storage to expanded or auxiliary storage and back again. MVS periodically transfers entire address spaces in and out of virtual storage so that they are temporarily unavailable for processing, which is called swapping.

When an address space is "swapped out", its critical pages (the ones that contain the tables that keep track of the location of each virtual storage page for the address space) are written to a special data set called a *swap data set*. Later, when the system can accommodate the job again, the address space is "swapped in" so that it can be processed again. Therefore swapping is usually done to alleviate some type of constraint situation. MVS/ESA is able to reduce the I/O operation and so there is an enhancement in processing.

Auxiliary Storage Manager (ASM) which is one of the MVS/370 and MVS/XA subsystems transfers Virtual Storage pages between Central Storage and auxiliary storage either by paging or swapping. ASM is called by the Real Storage Manager (RSM) and by the Virtual Block Processor (VBP).

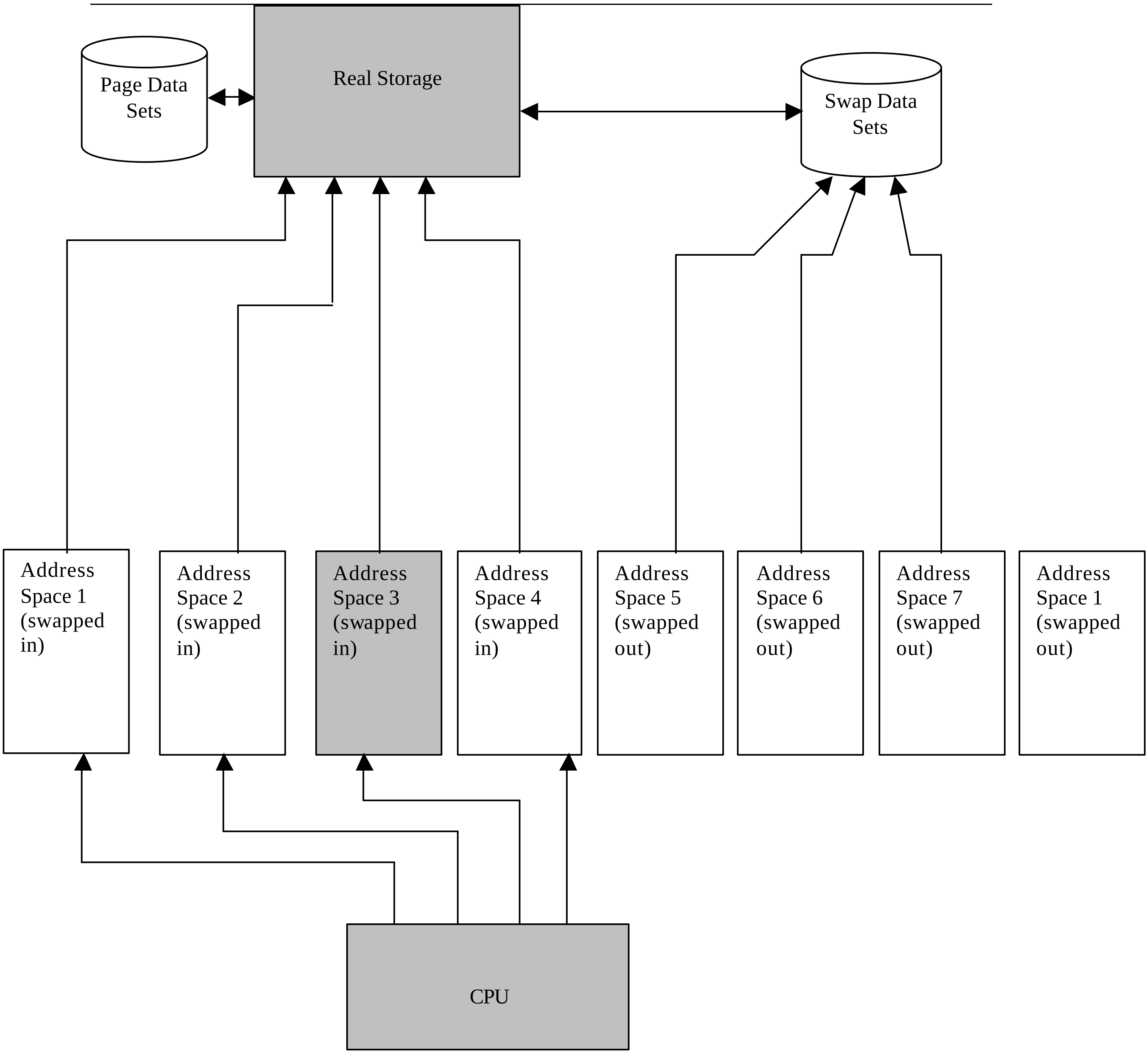

Page Data Sets
Real Storage
Swap Data Sets
Address Space 1 (swapped in)
Address Space 2 (swapped in)
Address Space 3 (swapped in)
Address Space 4 (swapped in)
Address Space 5 (swapped out)
Address Space 6 (swapped out)
Address Space 7 (swapped out)
Address Space 1 (swapped out)
CPU

Fig. 2.4 The concept of swapping in MVS

2.2.5 Program Modes

A program within an address space can run is either *real mode* or *virtual mode*. These modes indicate whether or not a program is subject to the paging or swapping processes. Paging and swapping operate only for programs that run in virtual mode. Programs that operate in real mode are not paged or swapped.

Some parts of the operating system are responsible for managing virtual storage and they cannot be subjected to paging process. These parts of the operating system are expected to be always resident in real storage. So they are non-pageable or non-swappable. Similarly, certain programs that communicate directly with channel devices or those have certain time dependencies, should run in real mode.

2.3 Address Space Organization under MVS/XA and MVS/ESA

Recent System/370 family processors provide 31-bit addressing rather than 24-bit addressing. Programs that are written in 24-bit addresses should be able to run properly on processors that provide 31-bit addresses.

In order to provide the compatibility and also to reap the benefits of 31-bit addressing, processors that use 31-bit addresses and the operating systems that support them provide a full 2GB address space, but they include special provisions for addresses below 16MB.

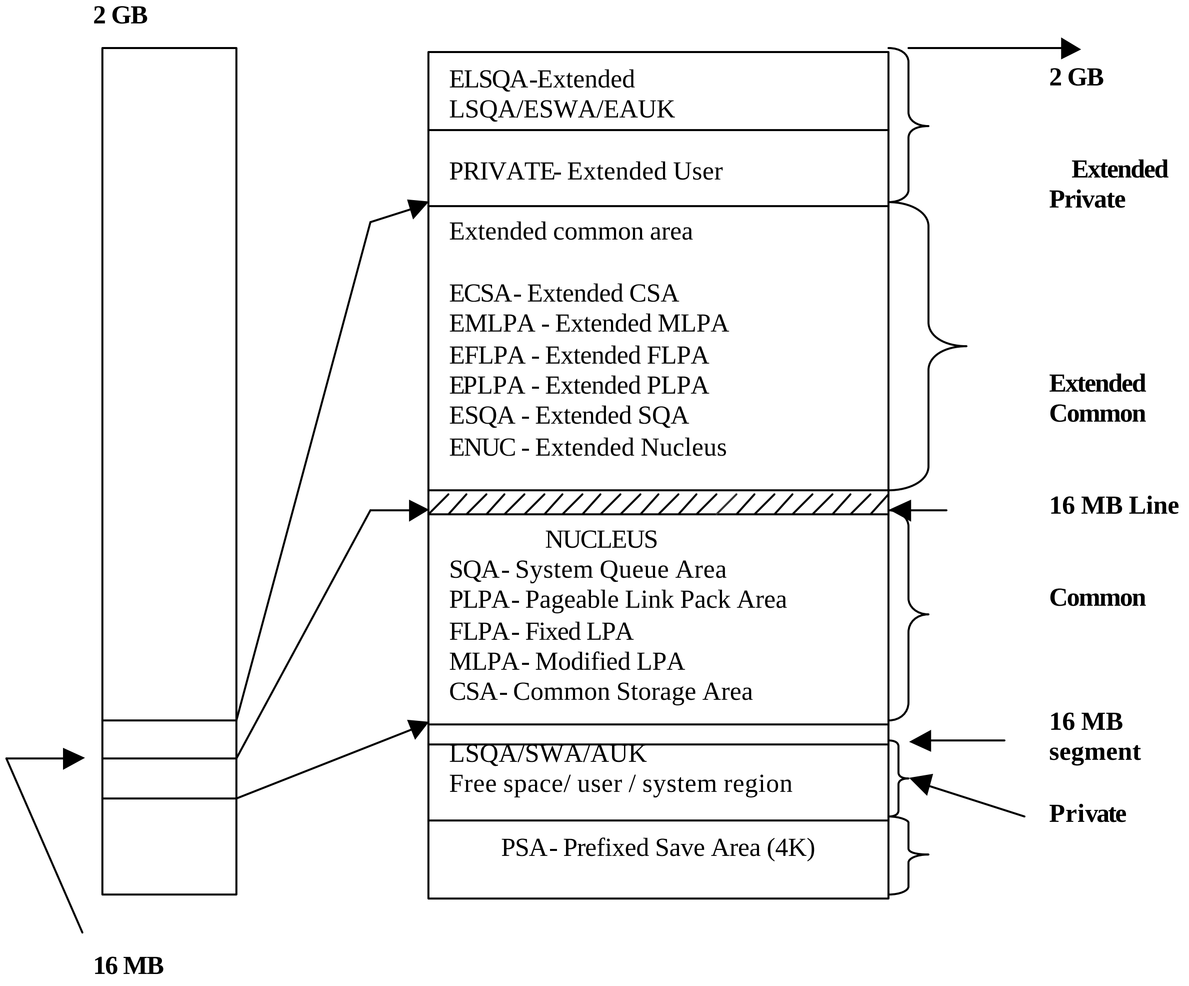


Fig. 2.5 MVS/XA Virtual Storage Management

The first 16MB of an address space can be accessed using 31-bit or 24-bit addresses. So 24-bit programs can run unchanged, still restricted by the 16MB limitation, while 31-bit programs can use the full 2GB-address space.

Fig. 2.5 shows the MVS/XA Virtual Storage map. The Virtual Storage map of MVS/XA is similar to the MVS/370 except that there is a piece of each type of storage above the 16-MB line and a piece of each type of storage below the line.

The main benefit of MVS/XA and MVS/ESA over MVS/370 is Virtual Storage Constraint Relief (VSCR). The first area of VSCR provided by MVS/XA is moving the nucleus to straddle the 16-MB boundary.

The conversion to MVS/XA provides 1.4 to 2.4 MB of VSCR, which comes from splitting the nucleus. MVS/XA does not completely remove the Virtual Storage Constraint. As the MVS/XA Operating System grows, more Virtual Storage will be needed for the Operating System.

Most of the new code and data areas will be "above the line", the real limitation being application programs. As they are converted to fully utilize the 2 GB address space, then VSCR disappears.

2.3.1 MVS/XA above the Line

The areas above the 16-MB line contain the same data/or programs that exist below the line with the only difference that all references to the data and /or programs have been changed to support 31-bit access. The areas are Extended Private area and Extended common area:

Extended Private (EPRIVATE) area comprises of

- Extended User (EUSER)
- Extended Local Systems Queue Area (ELSQA)
- Extended Scheduler Work Area (ESWA)
- Extended Authorized User Key (EAUK)

Extended Common area comprises of

- Extended Common Service Area (ECSA)
- Extended Pageable Link Pack Area (EPLPA)
- Extended Fixed Link Pack Area (EFLPA)
- Extended Modified Link Pack Area (EMLPA)
- Extended System Queue Area (ESQA)
- Extended Nucleus (ENUC)

2.3.2 MVS/XA Below the Line

The areas "below the line" are Common area, Private area and Prefixed save area (Common).

The Common area comprises of

- The NUCLEUS area
- System Queue Area (SQA)
- Pageable LPA (PLPA)
- Fixed LPA (FLPA)
- Modified LPA (MLPA)
- Common Storage Area (CSA)

The Private area comprises of

- Local System Queue Area (LSQA)
- Scheduler Work Area (SWA)
- Authorized User Key Area (AUK)
- User Region
- System Region

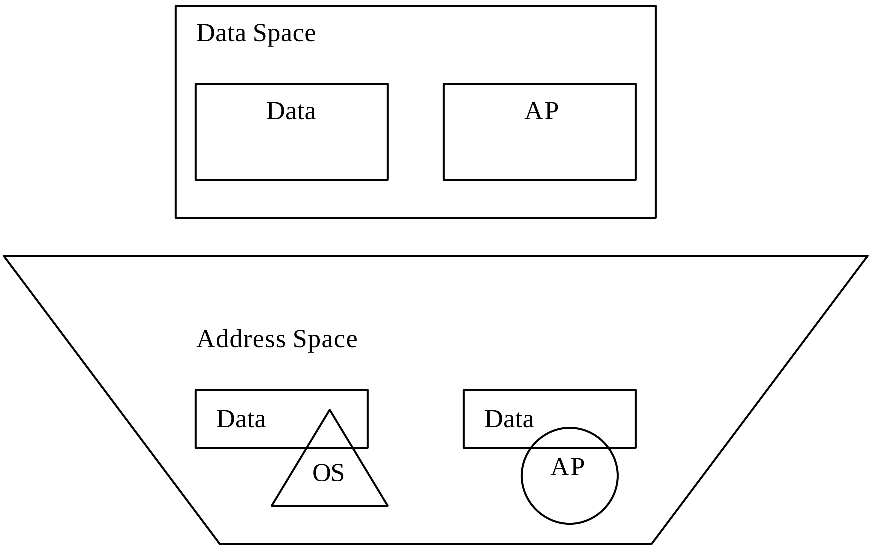
The main difference from MVS/370 is that the private area starts after the Prefixed Save Area (PSA). In MVS/370, the Nucleus is in "low storage".

2.4 MVS/ESA Data spaces and Hiper spaces

MVS/ESA has extended virtual storage by incorporating data spaces into the processing environment. Data spaces have taken some of the burden of keeping data in virtual storage off of address space. Data spaces also allow greater accessing speeds by reducing I/O. This serves to free up large areas of the address space for other uses. Having additional address space for programs allows us to incorporate more functions in them without imparing performance.

There are two types of data spaces. Both may be 2GB in size and this entire amount may be used for data or programs stored as data. The first type is a regular data space. It is backed by main, expanded and auxiliary storage. Data that resides in a regular data space may be address space. The second type of data space is a hiperspace (high performance space). It is backed by expanded and auxiliary storage. Data from a hiperspace must be accessed in 4 K increments and moved to an address space or regular data space for access by program instructions.

Application programs can access data from data space and not from Hiper space.



AP- Application Program

Fig. 2.6 Data Space

Data spaces contain user data and user Application program. Address space contains OS data and AP data. This isolation ensures no chance of data corruption. Programs can be stored as data in Data spaces but cannot be executed there. MVS/ESA supports High Level Languages such as COBOL, C/370, FORTRAN, PASCAL and PL/I and machine language 370 Assembler. Regular data space is maintained by main, expanded and auxiliary storage. Since it is backed up by main storage, it uses more costly resources than Hiperspaces. Hiperspace is backed up by Expanded and auxiliary storage.

Regular data spaces are byte addressable whereas Hiperspace is 4K addressable. Data cannot be accessed from Hiperspace. The data should be moved to Regular data space or address spaces so that Application programs can access them. While working with High level languages such as COBOL, FORTRAN, PASCAL,PL/I , it is possible to access a Hiperspace. With Assembler (370 Assembler) both Data space as well as Hiperspace can be accessed.

There are two types of data spaces viz., private regular data space and shared regular data space. Private regular data space is used by single application. Shared regular data space is used by more than one application program. Regular data space is the best choice for holding data and program when data needs to be manipulated using ESA/370 instructions

Characteristics	**Data Space**	**Hiper Space**
Up to 2GB	Yes	Yes
Hold data	Yes	Yes

Hold Programs and data	Yes	Yes
Execute programs	No	No
Resource space for OS programs and Data	No	No

Hiper spaces could be used in some applications where there are millions of transactions. Data could be moved in blocks with any I/O, here the data access is just a piece of the application. There are two types of shared regular data space., privileged and non-privileged. Data can be moved from address space to private regular data space or shared regular data space.

There are two types of Hiperspaces viz., Standard Hiperspaces and ESO (Expanded Storage Only). Standard Hiperspace is backed up by Expanded and Auxiliary storage. It provides faster data access than DASD. Only Expanded storage backs up the ESO Hiperspace. ESO also offers still faster data access. But these are reserved for privileged users. Data is stored in buffers of the 4K size increments in both Hiperspaces. Just like data spaces, Hiperspaces help in case of virtual storage constraints. If virtual data are crawling programs in address space, the data could be moved into hiperspaces. Using data spaces will be good idea with reference to file used very frequently. Thus the MVS/ESA offers data spaces and hiperspaces which reduces excessive I/O.

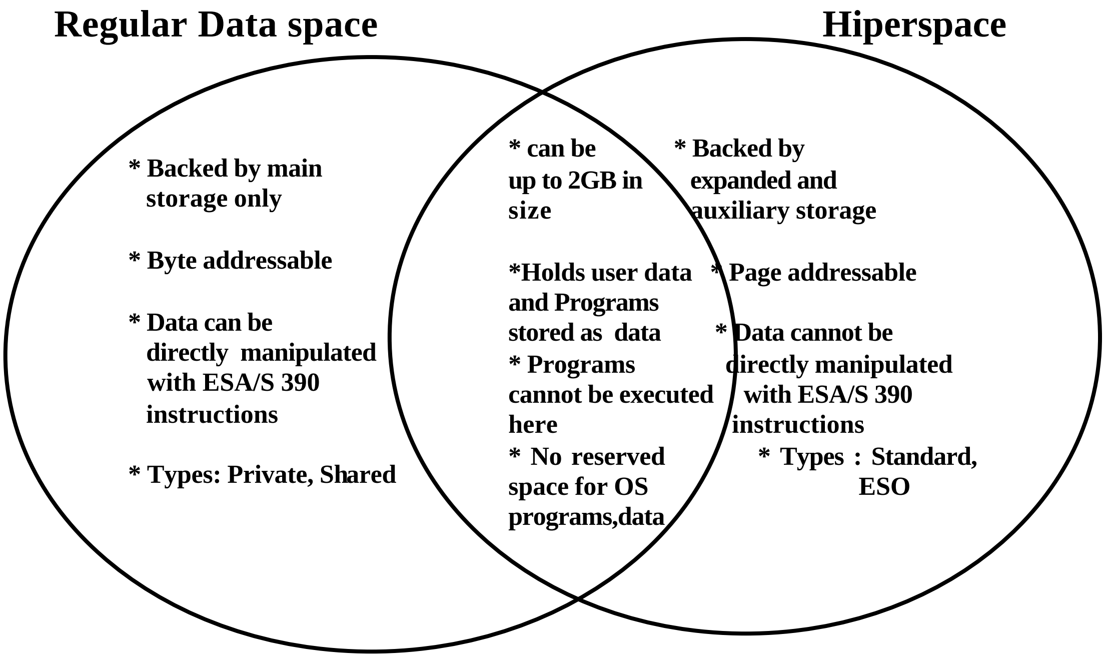

Regular Data space
Hiperspace
* Backed by main storage only
* Byte addressable
* Data can be directly manipulated with ESA/S 390 instructions
* Types: Private, Shared
* can be up to 2GB in size
*Holds user data and Programs stored as data
* Programs cannot be executed here
* No reserved space for OS programs,data
* Backed by expanded and auxiliary storage
* Page addressable
* Data cannot be directly manipulated with ESA/S 390 instructions
* Types : Standard, ESO

Fig. 2.7 Regular Data space and Hiperspace Characteristics

Benefits of using data spaces (regular and Hiperspace):

- I/O avoidance
- Isolation of data from programs
    - For data integrity
    - To allow growth of programs without virtual storage constraint
- Increased virtual storage

2.5 Virtual Storage Services

A virtual address space is created each time a TSO user logs on or a new batch initiator is started. Fig. 2.5 shows the creation of a virtual address space.

2.5.1 Address Space Creation

The address space creation routine checks to ensure a new address space can be created. If too many address spaces are running and there is not room, the request is deferred until the system can make room. In this case, the Virtual Storage Manager (VSM) is invoked to assign the Virtual Storage. This is just the creation of address blocks and adding them to chains that indicate an address space exists. A Local System Queue Area (LSQA) is built in the address space. The Region Control Task (RCT) control blocks, such as page tables are built in the LSQA. The RCT is dispatched and attaches the program or task. An example of a task is LOGON for TSO command.

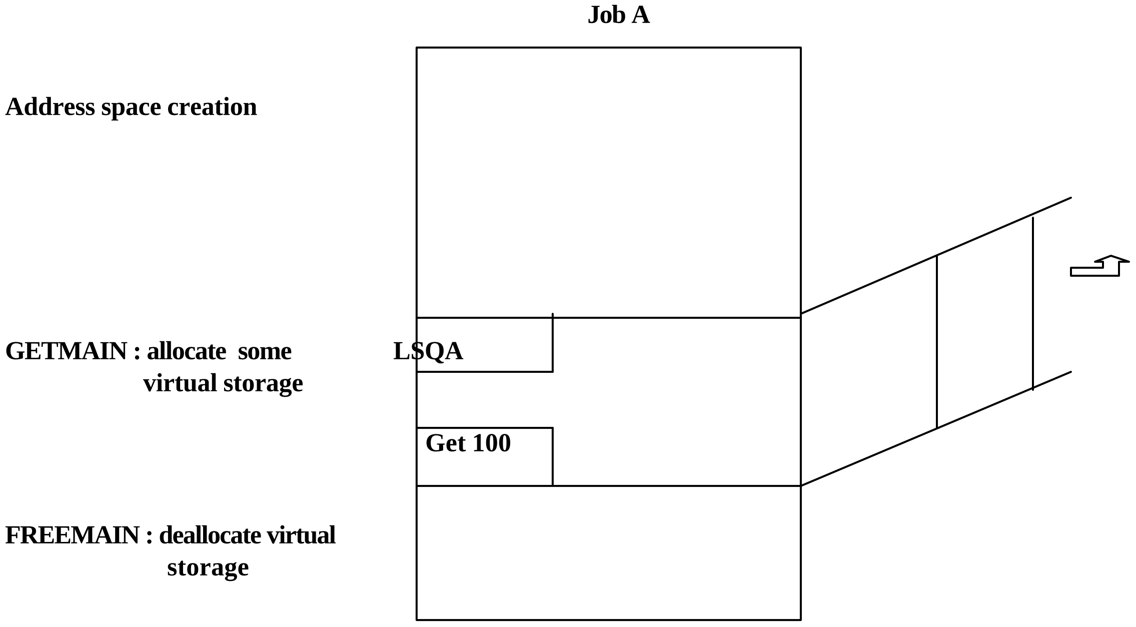

Job A
Address space creation
GETMAIN : allocate some virtual storage
LSQA
Get 100
FREEMAIN : deallocate virtual storage

Fig.2.8 Virtual Storage Processing

GETMAIN routines of VSM service the requests for allocation of Virtual Storage viz., new address space, space within an address space and common storage. The virtual storage is divided into subspools of storage which are numbered from 0 to 255. The subspool number indicates where the storage is located and the attributes of the storage viz., whether they are protected or not, fixed or pageable storage key etc.

The requests to free the Virtual Storage are serviced by the FREEMAIN routines of VSM. These routines update and combine control blocks to make Virtual Storage available to other requestors.

2.6 I/O system, Channels and Control Units

There are three important parts in data processing viz., Input, Process and Output.

Input: The files are read into Central Storage in the Processor Complex so that each logical record is available to be processed.

Process: The data is inspected and changed to the required format, calculations are made, design specifications are applied to follow those specifications and programming is done.

Output: The data is written as defined in the specifications. The output could be only a condition code, which indicates that a file is in balance, or the output could be a whole new master file.

I/O subsystems play a vital role in processing and hence should be given due importance by application programmers. I/O devices operate in the millisecond range and I/O operations require thousands of time units longer than instructions. I/O operations have to be tuned to minimize the elapsed time of a batch job or an online transaction. A Balanced system takes into consideration the CPU, I/O subsystem, Central Storage and the virtual storage. Balanced systems require that each resource in the Processor Complex should be in proportion to and respond accordingly to each of the other resources.

2.6.1 Input / Output System

The I/O system is the most important piece of a Balanced System. The various controlling factors of the I/O system are choice of access method, access pattern and the blocking factors :

- *Access method choice*: The access method is decided at the time of program design.
- *Access pattern*: The way a program calls in logical records affects the way the I/O system reacts. Once coded, it is usually hard to change.
- *Blocking factors* : The number of logical blocks per physical block is the easiest to change, which may be done at any time of the cycle of an application. Blocking factors can be changed to alter the performance of a job.

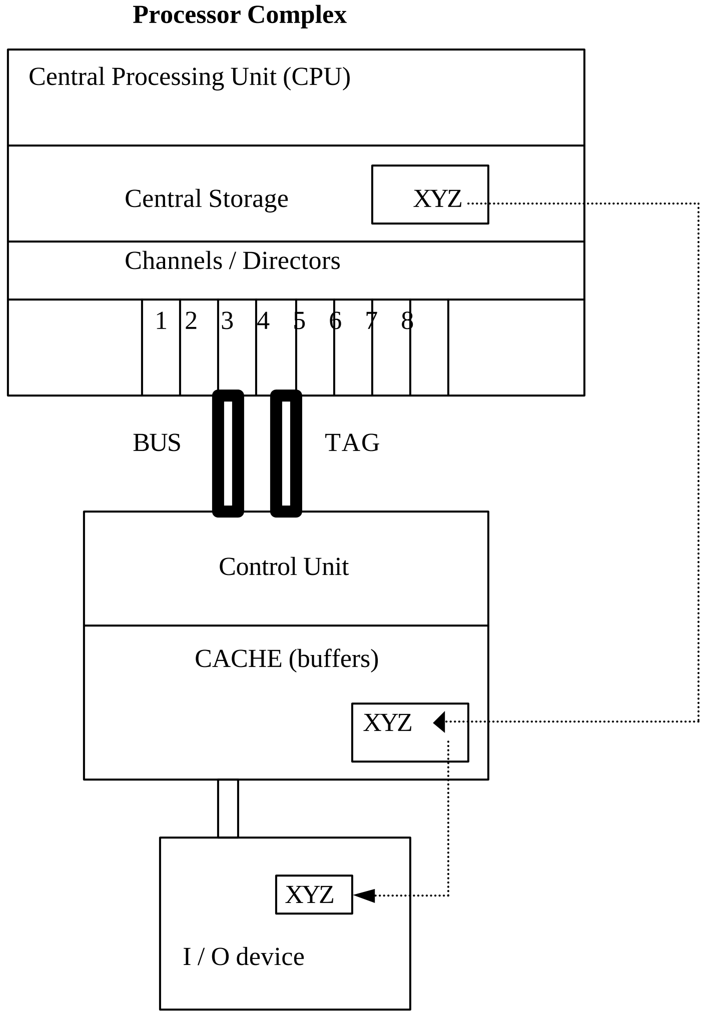


Fig.2.9 I / O System

2.6.2 Device Connection and Addressing

The Processor Complex and the control units are made with BUS and TAG cables or fiber optic cables. The BUS cable carries the data signals (characters of data to and from the device). The TAG cables carry control information about the operation.

The channel connection may be viewed either logically or physically.

- *Logical View* : The logical path is a three-digit hexadecimal device number :2F0 is the channel 2, control unit number F and device 0.

  For MVS/370 systems, the logical device address is the same as the physical device address. MVS/370 has a maximum theoretical limit of 4096 devices, as it uses only the three-digit number. The real MVS/370 limit is much smaller i.e., 1800 devices (depending on how many of each type DASD, tape, teleprocessing devices are defined).

- *Physical View* : The physical path is identified by a six-digit number : cp - cu - dv
  cp : Channel Path or Channel Path Identification (CHIPID). The usual limit is 256 channels, but Processor Complex model dependent.
  cu : Control Unit number. The usual limit for control unit numbers is 256, but can change depending on the configuration.
  dv : Device Address. The actual limit is 256 devices.

MVS/XA and MVS/ESA use the physical path to perform I/O instructions. The theoretical limit of the number of devices in MVS/XA is 65,536 (64K). The actual limit in MVS/XA started at 4096 and is expanded as IBM changes MVS.

2.6.3 Channels

The channel is the part of the Processor Complex that communicates with the control units, which attach I/O devices to the Processor Complex. The channel is a computer by itself. As such "Channel" includes all of the hardware and microcode in the channel subsystem. The channel processes I/O requests from MVS and interrupts the CPU when an I/O operation is completed. The channel directs the flow of information between I/O devices and real storage. This allows some overlap of CPU processing with I/O operations.

Channels are called "Directors" on the 3033, "External Data Controllers "(EDC) on the 308x series and "Channel Control Elements" (CCE) on the 309x series.

There are two modes of I/O operation: *Burst mode* and *Byte mode*. In birst mode, one I/O device monopolizes the channel for the duration of the whole block. In Byte-multiplex mode, any one device only monopolizes for a short birst, or a byte at a time. The byte mode channel has characters for both devices. Byte-multiplex mode is most efficient for very slow devices. It is required for teleprocessing terminals and for slow card readers as the 2501.

2.6.4 Types of Channels

There are three types of channels : byte multiplexor, selector and block multiplexor.

- **Byte multiplexor channels** operate on one byte at a time and handle teleprocessing and other byte-at-a-time devices such as slow impact printers and card readers. A byte multiplexor channel contains 256 subchannels and can operate at any one time in either byte or burst mode (depending on the length of the operation).

- **Selector channels** operate with only one device at a time. After the I/O device is attached to the channel, one byte at a time is transferred to the channel until all the bytes in a block have been transferred. If there are multiple blocks, then the channel stays busy until the device is ready to accept data for the next block. This continues until all the channel commands are completed.

- **Block Multiplexor channels** transfer physical blocks of data to and from devices. These operate in the burst mode. A Block Multiplexor channel is most efficient when running devices designed to operate in burst mode.

2.6.5 Shared and non-shared subchannels

The number of subchannels vary both in the byte multiplexor as well as block multiplexor channels. The subchannels are said to be *shared* when data transfer to and from a set of devices implied use of the same subchannel. A portion of the subchannel that holds the information about an I/O operation is *Unit Control Word (*UCW). Only one of all the tapes attached to a control unit can be reading, writing, or seeking at a time. Subchannels are *nonshared* when they have a single I/O device associated with the subchannel. DASD and the 3480 family of tape drives use nonshared subchannels.

2.6.6 Storage Control Units (SCU)

A storage control unit is a computer, which connects a channel to strings of I/O devices such as DASD. The control program of this computer is called *microcode.* When a control unit is powered up, an Initial Microprogram Load (IML) is performed, which loads the microcode from a floppy disk drive in the control unit. Microcode controls the timing of data transfer to the Processor Complex, provides information concerning the devices attached, and maintains error recording information.

Some control units may be attached to more than one channel. This is called *two-channel switch feature.* Two-channel feature is used for:

- *Redundancy :* If a channel to a control unit fails, the data center can still access data through the control unit.
- *Multiple access by two or more Processor Complexes:* When the data processing needs grow, two or more Processor Complexes will be required to service the need of the data center. In such cases, two Processor Complexes will need to share access to data on a single control unit.

2.6.7 Cache

There are three types of Cache. The first type of cache is the CPU cache in the Processor Complex. The second type is the Direct Access Storage Device (DASD) cache and the third one is the tape cache. Cache is relatively small and expensive but can improve I/O throughput. SCU cache for DASD can be controlled and tuned by the data center.

2.6.8 Central Storage Considerations

The ultimate goal of the MVS system is to transfer data between devices and 4096 byte frames in Central Storage. If too few pages are allocated for I/O buffers, I/O will depend on processing speed. Too many pages could result in Central Storage constraint. Inappropriate choice of data set block size can result in wasted page fragments or wasted I/O balance and moderation. Controlling allocation and use of DASD are the most effective means of improving application program efficiency.

2.7 Direct Access Storage Devices (DASD)

The invention of the transistor, the computer chip, has certainly brought down the size and cost of the computer. The DASD has made computer more popular, since its invention has made implementation of Virtual Storage, Relational database, Interactive processing etc., feasible. There are several terms to refer to direct access devices. The first is Direct Access Storage Devices (DASD) and the other is the disk. Disk really refers to single rotating metal platter. A third name associated with this type of storage media is Winchester Disks.

2.7.1 Storage Hierarchy

A storage hierarchy is used to organize the areas on which data and computer program instructions might be stored. Some areas are used for very short periods of time - "fractions of a second". Some of the areas are used for very long periods of time - "years". The storage hierarchy is shown in the form of a pyramid in fig.2.10. The top section represents data areas that are electronic in nature. This storage is very expensive and so there is usually a relatively small amount. Electronic memory is measured in bytes, kilobytes, and megabytes. The bottom of the hierarchy represents the least expensive media.

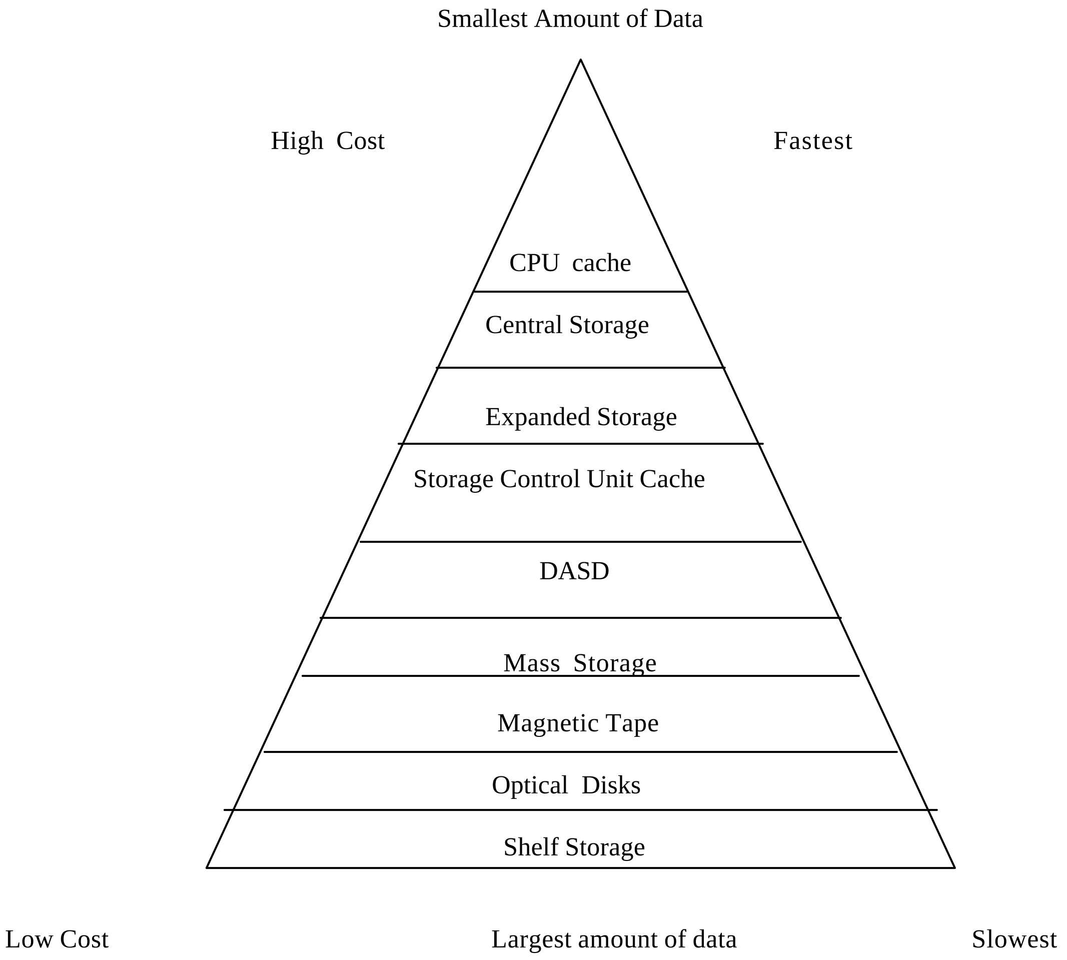


Fig.2.10 Storage Hierarchy

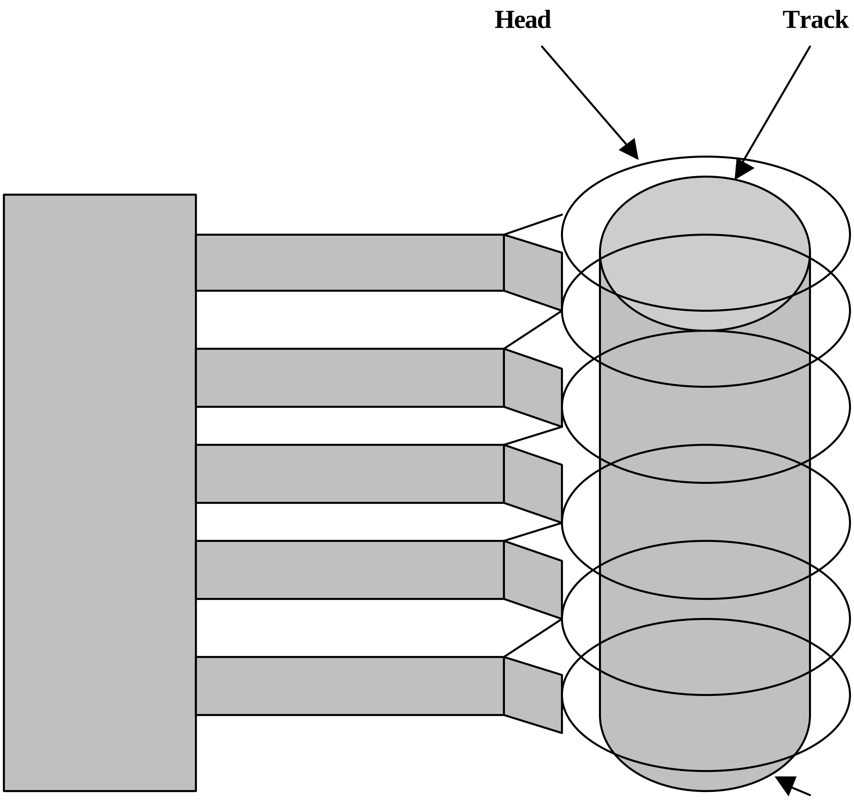
CPU cache is inside the Processor Complex and contains data and instructions that are required for the window of instructions being executed by the program. **Central Storage** is the storage the contains all of the data and instructions the program requires for execution. **Expanded Storage** is an extension on the 309x model series and as MVS/ESA grows, Expanded Storage will become a very valuable area of storage for data. **Storage Control Unit cache** contains copies of data records between the Processor Complex and the I/O device. **DASD** is the spinning medium and is the cornerstone of the storage hierarchy. **MASS Storage** devices such as the IBM 3850 Mass storage system, are a combination of DASD and tape units. **Magnetic Tape Storage** contains largest volume of data, traditionally a place for long-term data. **Optical disks** are not supported by IBM on their mainframes, though they are being used in their laboratories. An advantage of the Optical disks is that it can hold several gigabytes of storage per side, hence called the "jukeboxes". The disadvantage is that the access times can be relatively long since the platters rotate at a slower rate (54 versus 16.5 ms for real DASD). **Shelf storage** is the part of the hierarchy that contains disks and tapes, which can be removed and placed on a shelf.

2.7.2 DASD components

Head Disk Assembly (HDA)

The Head Disk Assembly contain the recording media and the necessary physical read/write mechanisms to record and transfer data to and from the recording media.

- *Disks:* The HDA has several round platters coated with a magnetic surface. Each platter or Disk is attached in the center by a column. Th column and platters rotate in unison, relatively fast- 3600 revolutions per minute. The set of tracks corresponding to one radial head position is called a cylinder.



on **Fig.2.11 "Volume" the MVS**

- ♦ *Performance:* The performance is best if the data is placed on the middle track, since the head movement is the least when taken an average.

- ♦ *Actuator (Volume):* The term Actuator refers to the set of read/write heads, access arms, and a motor to move the head across the surface of the disk. Actuator also refers to the DASD space accessible by one set of read/write heads. The IBM hardware documentation uses the term actuator for the read/write heads. The data center uses the term "Volume", because the MVS JCL uses "VOLUME=" parameter to identify on what volume a data set resides. The size of the area where a single bit is stored, called a bit cell, is determined by

  - ➢ The thickness of the coating of the disk.
  - ➢ The height of the read/write head as it flies above the surface of the disk.
  - ➢ The length of the gap in the read/write head to detect the bit.
  - ➢ Laws of Physics. There are practical limits that cannot be surpassed with current technology.
  - ➢ Profitable manufacturing methods. This is one of the most important factors.

Tracks - Groups of Blocks: On the recording surface of the DASD platter are areas, which are defined as tracks. These are complete circles to record data.

Logical Components of an Actuator - Records:

The application designer and programmer develop logical record descriptions to contain the information needed for their "record". Logical records are written and read by the application. The logical records are grouped into a physical record to be written to and read from the DASDs. Physical records are recorded in small pieces called track cells. Each track of 3380 is divided into 1499 user data segments of 32 bytes each, If physical block were 3200 bytes long, then the block would use a whole number of segments. If the physical block were 3000 bytes long, then the 94 blocks, or 3008 bytes of the track, would be used to store 3000 bytes.

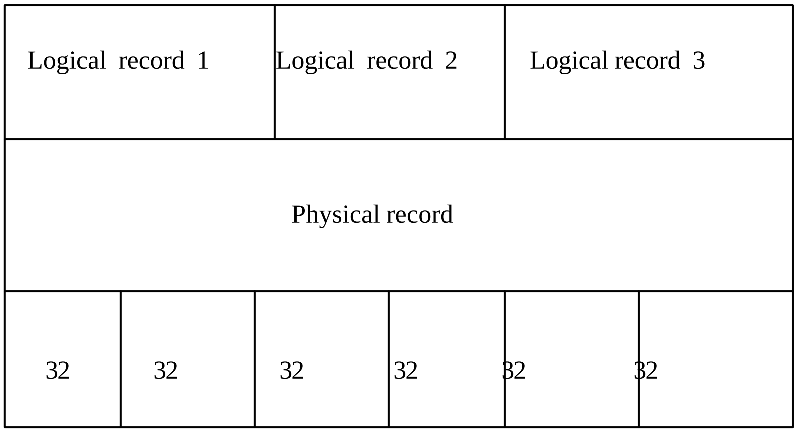


Fig.2.12 Physical record or block

2.7.3 Formatting of DASDs

A track can be formatted in one of the two manners:

- *Fixed Block Architecture (FBA):* Tracks are preformatted with fixed size blocks usually 512 bytes. Every physical record on the track is the same size. If the application needs smaller or larger logical records, they are "fitted" into the fixed size by the operating system.

- *Count Key Data (CKD):* CKD architecture devices have a self-defining record structure which allows records from approximately 18 bytes up to the full track size to be recorded on the surface of the disk. The maximum number of bytes stored on the surface of the DASD is device dependent, it depends upon the surface coating of the disk. Finer material allows bits on the track to be closer. For DASD 3380 this limit is 47476. The maximum supported block is 32760. This reluctance to change is one of the prices we pay for this upward compatibility.

Why 32760?

In the System/360, half word was used to represent the block length, i.e., 16 bits. Out of these 16 bits, one higher bit is used to determine, whether the contents of the register occupying full or half word, remaining 15 bits store maximum of 32767 number. This is further rounded to 32760. Hence is the maximum block size 32760. Non standard access method can provide you to read or write blocks of data that exceeds 32760 bytes.

Inter block Gap

The electronics of the device require a certain amount of time to detect the end of a physical block and get prepared for another block. This gap differs from device type to device type, not because the platter is spinning faster (they all spin at the same rate- 3600 RPM), but because the recording density differs.

Device type	Track capacity	Gap(bytes)	Cylinders/tracks
3370	13030	130	404 / 19
3350	19069	185	555 / 30
3380	47476	500	887

Table shows track capacity and inter block gap for some of the devices.

Count Key Data tracks

Now let us look at the format of each physical block on the track shown in fig.2.13.

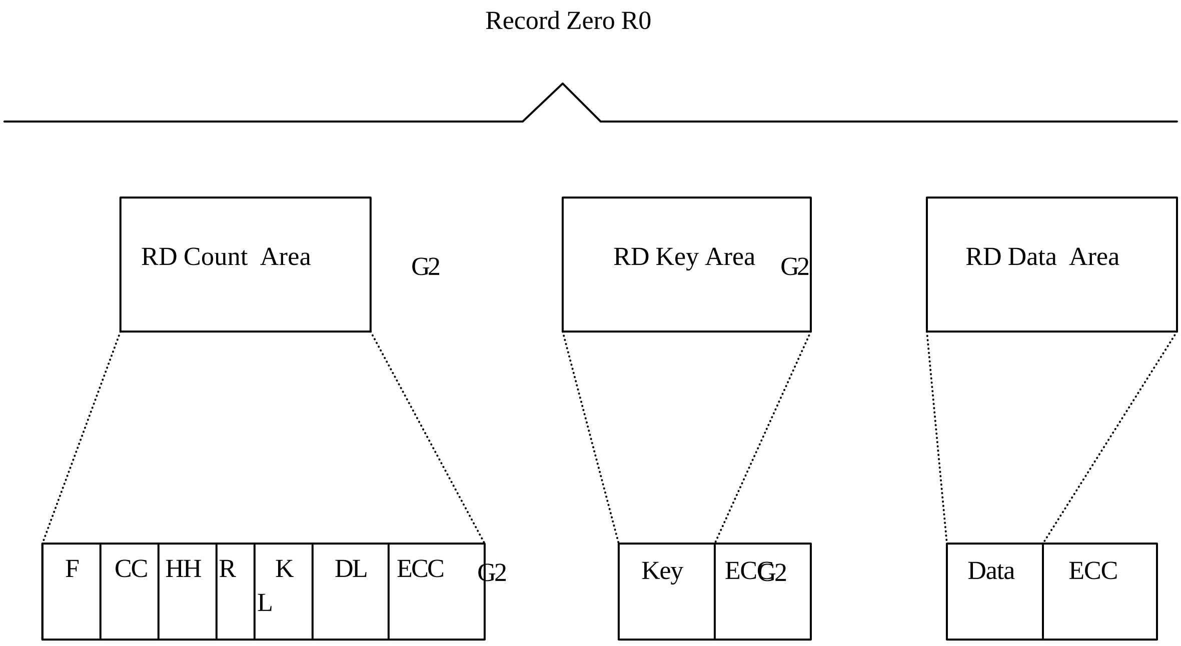


Fig. 2.13 DASD record contents *(Each "record" on DASD is a self-defining record : the record is in three- parts: count, optional key area, and optional data area)*

Count Area

Each block has an eight-byte area which contains flags, the address of the record, the length of the key, and the size of the key area(if any) and the size of the data block.

Key Area

An optimal key area may exist. The key was used by two of the old data set architectures- Basic Direct Access Method (BDAM) and Indexed Sequential Access Method record. Keyed data failed to be of much commercial value, because the key was usually needed inside each logical record, not outside the physical block. There are some specialized data sets that are keyed in MVS. The VTOC(Volume Table of Contents) on a DASD volume is a keyed data set. Partitioned data sets have a directory at the beginning of the data set. The directory is a keyed area. The CVOL catalog structure also had a keyed area.

Data Area

The Data area, which is also optional but usually present, contains the physical block of data. Each physical block can contain multiple logical records.

2.7.4 Track Defects

When a data block is written to the disk surface, the actuator does not have a write head and a read head at the end of each arm, so a write command does not really know if the data was written properly. In the early versions of DASD, write failures had larger probability of failure than today. Job Control Language(JCL) allows the programmer to specify write verify (OPTCD=W). The access method will write the record and then read the record just written to ensure that the record actually made it to the recording surface. This option need not to be used with 3380 or later devices. The reliability of write command is so high that the price is not worth it, the record is written on the track, but you told the access method to try to read the record. A complete revolution is required to read the record.

Each track can actually contain more than 47968 bytes (47766 actually available for 3380). If any one 32-byte segment has a bad spot, then the 3380 can be instructed by MVS to skip the bad area. These skip displacements are used up on a single track, then the track is marked bad.

Delays

Actuator movement - called seeking - is a time-consuming part of data access. The head is moved across the surface of the disk to a new group of tracks. The time it takes for the read/write arm to move from its current position to the desired cylinder is called *seek delay*. Seek delay is usually measured in milliseconds.

Rotational Position Sensing (RPS)

This rotational delay, also called *latency*, is the time for the data on the track to reach a position so that the read/write head can read the data. The time used for most calculations is 8.3 ms.or one half a rotation. RPS improves the overall efficiency of the entire DASD subsystem because it allows multiple actuators to be active at the same time.

2.8 Peripherals

2.8.1 Magnetic Tape

Magnetic tape is the best storage medium for data and contains most of the data that is stored in the data processing environment. Magnetic tape is made by taking a plastic tape and bonding a layer of magnetic material on the tape. A spot on the tape could be magnetized one way and would represent a *one.* Magnetized in the other direction, the spot would represent a *zero.* Groups of these spots can be combined to represent one byte. Groups of bytes are a physical block on the tape. The beginning of the data recorded is indicated by a metallic "reflector" strip (in the case of 3420 tapes) or by the tape drive "counting" to the start of the tape (in the case of 3480).

2.8.2 Performance Characteristics of Tape

The performance of the tape subsystem is measured by the throughput of the subsystem in megabytes per second. The throughput depends on certain architectural factors as well as certain factors decided by the application programmer.

- **Architectural Factors**

The number of active tape drives and channel paths: The number of channel paths limits the number of drives transferring data.

Buffering in the control unit (tape cache): If the control unit does not have to wait for the tape to complete an action, the throughput of the subsystem will be better.

The speed of the tape over the read/write heads: The faster the tape moves, the faster data can be transferred.

Channel speed: The speed of the channel can operate an upper limit on the data transfer rate. If the channel operates at three megabytes per second, then the tape drive can operate at any speed to and including three megabytes per second.

Channel utilization: Selector mode channels can operate up to 100 percent channel busy without degradation of the entire channel. The total transfer of all the blocks from the tape may be longer because the I/O is waiting for the channel to get started.

- **Application Design Factors**

Size of the data set: Very large data sets can monopolize a tape subsystem.

Block Size of the read/written data set: Small blocks waste tape (inter block gaps) and throughput. The channel, control unit, and tape drive perform the same work for a 4096 byte block as they do for a 32760 byte block in the areas of selection, setup, tape-up-to-speed, and stopping.

Selection of write validity checking (tape write immediate on 3480 devices): Caching is disabled when tape write immediate is selected. Tape forward spacing and small block return to the speed of the 3420 tape drives. The only faster item is data transfer rate.

2.8.3 Tape Architecture

There are two types of magnetic tape architecture in use : round tapes and square tapes. The round tape reels are 10.5 inches in diameter and the magnetic tape is 0.5 inch wide and usually 2,400 feet long (for full-sized reels). The coating is generally iron oxide. In the 3420 tape drive family there are three major groups:

- *Tape Density in bits per inch (bpi):* 800 bpi, 1600 bpi and 6250 bpi
- *Transport speed:* The maximum speed of the tape across the heads in inches per second (ips), there are three options : 75 ips, 125 ips and 200 ips.
- *Recording formats:* There are two recording formats. 7 tracks of data across the one-half inch tape and 9 tracks of data across the one-half inch tape.

The 3480 Magnetic Tape Subsystem is a high-performance, more reliable magnetic tape subsystem than 3420 tape family. 3480 data cartridges are four inches square and contain chromium-dioxide-covered tape. It can hold 20 percent more data than the round tape though the cartridge is one-fourth the size of the round tape.

2.9 Storage Management Subsystem (SMS)

The Storage Management Subsystem, commonly referred to as SMS, is an IBM product that became available with MVS/ESA. The purpose of SMS is to provide the user with system-managed data, intended to accomplish the following objectives:

- ♦ Simplify user interface for allocating and maintaining data.
- ♦ Provide the user with a logical view of data, separated from the actual physical device characteristics.

- Provide centralized control of disk storage by putting a data set under SMS control (SMS-managed), the user derives some important benefits.
- Disk allocation for new data sets is directed to the most suitable group of disk volumes under centralized disk-storage administration control. The user does not determine which volumes will be used. SMS does.
- The management of a data set after it is created of archiving, retention, etc., is also under centralized administration control.
- JCL is simplified.

2.9.1 Automating Storage Management

SMS is a group of software products, which automates tasks related to disk storage management that were previously handled manually. SMS is an attempt to separate the logical view from the physical view. SMS is implemented through a group of software products collectively referred to as DFSMS. It enhances the system performance, reduces data center complexity and gives the application programmer more time to develop applications.

2.9.2 Managed Storage (DFSMS)

Components of DFSMS	Function
MVS / DFP: Multiple Virtual Storage / Data Facility Product	Integrates all the Storage Management Subsystem components.
DFHSM: Data Facility Hierarchical Storage Manager	Automates archival procedures for data sets.
DFDSS: Data Facility Data Set Services	Automates backup and space management procedures for data functions.
DFSORT: Data Facility Sort	Provides sorting and copying functions.
RACF: Resource Access Control Facility	Controls access to data sets according to the security requirements specified at the site.
ISMF: Interactive Storage Management Facility	Creates and revises storage management constructs.

Data Facility System Managed Storage (DFSMS) includes tools and procedures used to automate many of the management tasks. DFSMS is a concept that is implemented with the Storage Management Subsystem (SMS) software.

The various components of the Data Facility System Managed Storage (DFSMS) have been tabulated here.

2.9.3 External Storage Manager

The primary way that SMS simplifies the JCL coding requirements is by allowing us to focus on the logical view of the data as opposed to the physical devices it is stored on. This means that automation of the specification of such logical attributes as frequency backups, retention periods, data security, and performance issues when a data set is created could be done. SMS achieves this through the logical data manager and external storage manager.

The logical data is indicated and the logical data manager pages the information to the external storage manager. The external storage manager translates this information into physical attributes that match the various storage devices available.

The system automatically places the data sets for optimum efficiency. The different SMS set attributes together with storage group allows us to specify data set attributes with minimal coding requirements.

The logical data manager manages the following logical data requirements:

- Data set format
- Record management
- Security
- Defines and manages logical views of storage
- Reporting
- Tracking of logical storage
- Access to external storage manager functions
-

The external storage manager manages the following physical data requirements:

- Physical storage space
- Physical storage performance
- Physical storage availability
- Physical storage device installation
- Local device attachment
- Device configuration
- Multisystem storage control
- Common storage format across devices

The application programmer or data administrator specifies logical attributes of data through the Interactive Storage Management Facility (ISMF). The logical manager and the external storage manager work together to translate logical attributes into real storage attributes.

2.9.4 Storage Class, Data Class and Management Class

Let us illustrate the specifics and replaces of the Storage Class, Data Class and Management Class.

SMS Construct	Specifics	Replaces
Storage Class	Physical data set placement according to the desired levels of performances and availability	JCL for unit and volume serial number

Data Class	Physical attributes of data sets	JCL for data set organization, record length, record length, key length and offset, space allocation, and retention period. The need for the IDCAMS utility when allocating VSAM data sets
Management Class	Criteria that will automatically determine the life cycle of a data set	Manual procedures for backup, archiving, and retention of data sets

It is to be noted that a data set associated with a storage class is to be SMS managed.

2.9.5 Storage Group

A storage group is predefined set of DASD volumes which share the same general attributes. Storage group is not specified with JCL. SMS selects storage group based on specified class definition.

The major benefits of Storage Group are as follows:

- Simplifies reconfiguration of DASDs.
- Common devices may be managed in common fashion.
- Different types of data can be isolated from each other e.g. test and production data.
- Ability to isolate devices by their physical characteristics.

2.9.6 Automating Class Selection

Another way SMS provides the automatic assignment of system defaults is through the facility of Automatic Class Selection (ACS). This facility supplies default data definition and life cycle management attributes for the data sets. ACS is implemented as a series of routines developed by data administrator or system programmer. The routines can then supply defaults for data class, storage class, management class, and storage group.

Automatic Class Selection is a set of routines, which assigns defaults for storage class, data class, and management class of a data set if they are not externally specified with JCL. ACS uses class assignments as criteria to assign a storage class.

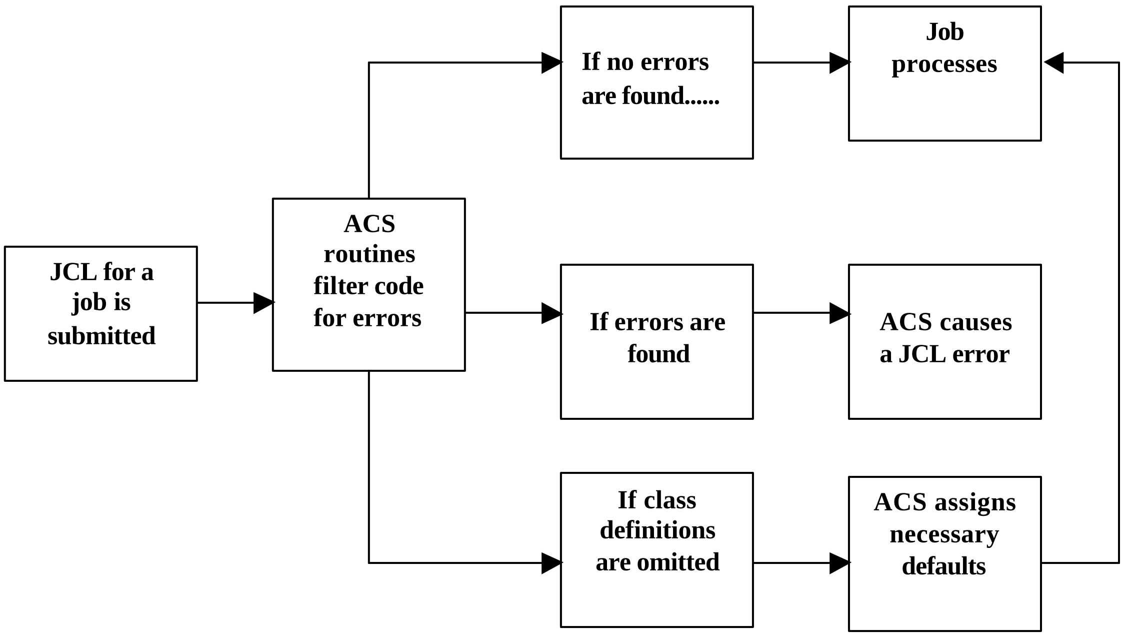

JCL for a job is submitted
ACS routines filter code for errors
If no errors are found......
Job processes
If errors are found
ACS causes a JCL error
If class definitions are omitted
ACS assigns necessary defaults

Fig. 2.14 Automatic Class Selection (ACS)

2.9.7 Partitioned Data Set Extended (PDSE)

All experienced programmers at one time or another have been frustrated with the common drawbacks of using a partitioned data set (PDS). How a PDS allocates and searches its directory and its inability to automatically reuse storage space can create problems for a programmer working in a dynamic environment. The data sets required for most applications are constantly growing and evolving and require more adaptability than a PDS can provide. The implementation of partitioned data set extended (PDSE) overcomes many of the problems encountered with the PDS. PDSE allows a random search of the directory, as well as immediate reuse of storage space occupied by deleted members. A PDSE does not have a fixed PDS. This allows us to dynamically increase the number of members in the PDS as the processing needs demand it.

A Partitioned Data Set Extended is an SMS data set divided into multiple members, each indexed by one or more directory entries.

2.9.8 Advantages of PDSE

- A PDSE directory can be searched randomly providing quicker access to members.
- Free space can be reclaimed automatically when members are deleted from a PDSE.
- A PDSE directory is not fixed in length. It can be enlarge dynamically to fit the number of members stored.
- A PDSE may contain up to 123 extents.

Note that unlike a PDS, a PDSE cannot hold load modules.

2.10 Reliability

Reliability is the probability that a system will not fail during any given period of time. Highly reliable systems rarely fail and if they do fail, the time to recover is relatively short. Reliability is " does the software/hardware break?". A number of measures of reliability exist:

- Mean Time between Failures (MTBF): The average number of hours or days is calculated and tracked.
- Failures Per Megabyte Transferred: This is a popular one to be used for DASD or TAPE.

2.11 Job Entry Subsystems (JES)

The Job Entry Subsystem (JES) is the part of MVS, which manages batch jobs, and SYSOUT under MVS. There are two flavors. The first is JES2. The second is JES3. Similarities between JES2 and JES3 are as follows:

- Both receive batch jobs to be processed by MVS.
- Instream data (SYSIN) is stored until the jobs and OPENs/READs the file.
- Both support writers to print the batch jobs from local users and remote users of the data center.
- They support TSO.
- Both JES subsystem writes accounting information.
- Both support remote computing - Network Job Entry.
- Ability for batch jobs to submit other jobs during execution.

JES2 and JES3 are subsystems to MVS. They are able to handle the above functions because they "register" with MVS method used to communicate between JES and MVS. JES is started after the Master Scheduler is completely initialized. In fact, the only thing that can start after the Master Scheduler is a JES must be running for tasks to start and stop. JES really is in two parts. The first part is a separate address space, which does the work of establishing an environment for other address spaces to use JES services and managing jobs. The second part is a set of service routines, which are available to all address spaces - in the Pageable Link Pack Area.

2.11.1 JES Processing

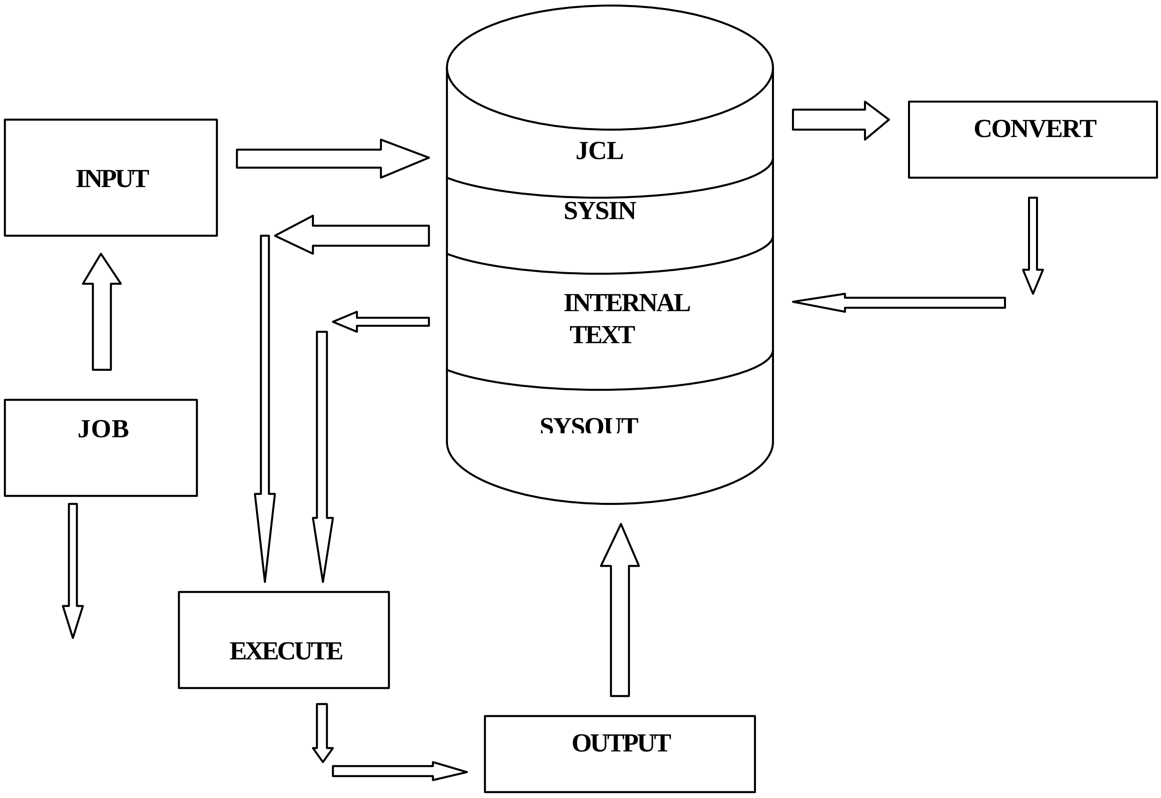


Fig.2.15 JES Processing

Input Processing - Reads a job into the system. An "internal" reader allocated from a batch job, a started system task, or a TSO session may also be used for a software reader interface. Once the whole job has been read, it is placed on a queue for the next phase.

Conversion Processing - The converter scans the Job control language (JCL) for syntax errors and creates an encoded representation of the job called internal text. If all looks well, then the job is placed on the next queue: job transmission or execution. If there is an error cryptic message is included with the JCL and it goes to the Output processor.

Job Class	Characteristics
A	Will execute within 15 mins of submission
B	30 mins
C	1 hour
D	Will execute overnight
H	Will be held until released by operator
L	Within 15 mins of submission; each step limited to 1 min execution time
T	requires tape processing

Execution Processing - JES keeps track of each job by building a Job Queue Element (JQE) to present the job. The JQE has name, the job number, the job class, the Priority, Input, an Output routing, and pointers to other control blocks. A job class may have 1 to 38 possible classes. A-Z are the first 26 possibilities, 0-9 are 10 more, and "Started Task" and "logon" are the 37th and 38th. On the execution queue, a job is selected by these criteria:

- Is there an initiator available with this job class? If an initiator has more than one job class assigned, then jobs are selected in order by job class. For example, if initiator 10 has job class "AB" assigned, class "B" jobs will be selected only if there are no class "A" job.

- Is the priority of the job the highest in the queue for this job class? As the job executes, the Job Entry Subsystem performs simulated Input/Output operations for SYSIN and SYSOUT data sets. At the completion of the job, JES places the job on the next queue- output processing.

Out put Processing - As a job produces output, Job Output Elements (JOEs) are built. Each unique output data set with different characteristics is represented by JOEs are kept in a table called the Job Output Table (JOT). Output data sets are written to print and punch files in the JES SPOOL. They can be routed to local and/or remote locations. Only after all spooled data sets are processed can the job proceed to the purge queue.

2.11.2 JES2 AND JES3 Comparison

JES2	JES3
Non-centralized processor complex management	Centralized management techniques where one processor complex is the King and other JES3 are slaves
Device allocation is controlled by MVS routines	The Global JES3 Processor Complex controls all allocation
Device allocation is done at step level	Device allocation is done at JOB level
If one JES2 fails, the work continues in other Processor Complexes	If the JES3 Global Processor Complex fails, the GLOBAL must be switched to a Local Processor Complex.
In JES2 controlled environment, all consoles are MVS consoles. JES2 uses MVS console services to communicate with operator	In a JES3 environment, most consoles are controlled from the GLOBAL Processor Complex

2.11.3 Initiator

Initiator is an address space that runs in the system region of an address space that is eligible for batch job processing. Each initiator can handle one job at a time. The number of active initiators on a system determines the number of jobs that can be multiprogrammed at once. Each initiator has one or more job classes associated with it. Within a job class, a job for execution is assigned depending upon the priority. JES assigns the jobs to the initiator.

Initiator	Eligible job classes
1	A
2	B,C,D,H,L,T
3	B,C,D,H,L,T
4	B,C
5	B,C
6	C

2.12 System Generation and Initialization

System generation and initialization are activities that are required to establish a working MVS system.

2.12.1 System Generation

System generation is the process of creating an MVS system, and System initialization is the process of starting a previously generated MVS system.

The basic components that make up MVS are on a series of tapes, called distribution libraries. System generation selects and assembles the various distribution libraries. To control system generation, normally known as sysgen, a system's programmer codes special macroinstructions that specify how the MVS components from the distribution libraries have to be put together.

An installation must have a working MVS before it can create a new one. Since a working MVS is required to execute the macroinstructions. Sysgen is usually used to upgrade to a newer version or to make changes to the current version. For installations which does not have a MVS already. A small, limited function MVS system is setup that can execute the Sysgen for the complete full function MVS system.

The macroinstructions that Sysgen uses fall under two categories. The first category of macros defines the system hardware configuration. They are needed because MVS must know about every I/O device that is attached to the system. As a result for every I/O added, the system must be generated again. (Actually smaller, less time-consuming type of Sysgen called an iogen can be used to change the device configuration).

The second category of macroinstructions in a sysgen indicates which options of the operating system should be included. They indicate whether JES1 or JES2 is used, what optional access methods are installed, and so on. The output from a sysgen is a set of system libraries that among other things contains the executable code that makes up the operating system.

2.12.2 System Initialization

System initialization is a process by which MVS system code is loaded from selected system libraries into central storage. This process defines the MVS system.

The system requires initializing:

- after generating a new system.
- after changes have been made to the system.
- after a system failure.

To begin the system initialization, the system operator uses the system console to start an initial program load that causes the computer system to clear its real storage and begin the process of loading MVS into storage from the system libraries.

Additional System Libraries

SYS1.LPALIB is the link pack area library. It contains re-entrant code such as access methods (other than VTAM or TCAM), supervisor routines, JES routines, and user written routines.

SYS1.LINKLIB contains programs and routines (such as utility programs, assemblers, the linkers, editors and service aids).

SYS1.LPARMLIB is a parameter library, which contains IBM-supplied and installation created system parameters.

Hardware Requirements

- A minimum of 8M bytes of central is required to initialize either an MVS/ESA system. Processor storage includes both central storage and expanded storage.

- The system console communicates with the hardware while the MVS console communications with MVS.

- The system console and the MVS console are separate devices on a 3090, but are the same device on a 4381.

Steps required in the initialization process are:

- Load the nucleus.
- Locate the master catalog and other system data sets.
- Initialize MVS using SYS1. PARMILIB.
- Build first address space.
- Create other system address spaces.

Review Questions

1. Distinguish between real storage and virtual storage.
2. What do you mean by an address space?
3. What are page data sets?
4. What are the sizes of the virtual storage in MVS, MVS/XA and MVS/ESA?
5. What is paging?
6. Define the terms: page and page frame.
7. What do you mean by expanded storage?
8. What is swapping?
9. What is a swap data set?
10. What do you understand by the terms 'MVS/XA above the line' and 'MVS/XA below the line'?
11. Compare the characteristics of Data Space and Hiper Space.
12. Compare and contrast Regular Data Space and Hiper Space.
13. What are the advantages of using Data spaces?
14. State the use of GETMAIN routines.
15. What are the types of modes in which the channels are operated?
16. What are the different types of channels?
17. Name the DASD components.
18.What is an Actuator in a DASD?
19.What are the two types of formatting the DASD?
20. What do you mean by the CKD format of DASDs?
21. What are the track defects that occur on DASDs?
22.What do you mean by seek delay?
23. What does Rotational Position Sensing mean?
24. Name some of the peripheral devices.
25. What are the different architectural factors deciding the performance of the tape devices?
26. What are the different application design factors that decide the performance of the tape devices?
27.What are the different recording formats of the tapes?
28.What are the advantages of the Storage Management Subsystem?
29. What do you mean by DFSMS?
30. What is the function of the DFHSM?
31. What is the use of DFDSS?
32. What is the function of DFSORT?
33. What are the various components of the DFSMS?
34. What is the function of ISMF?
35. What is the advantage of having an external storage manager?
36. What are the different types of classes that an SMS can have?
37. What are the benefits of Storage Group?
38. What is the advantage of Automating Class Selection provided by SMS?
39. How does a PDSE differ from that of a PDS?
40.What are the advantages of a PDSE?
41.How do you describe the term "Reliability" in MVS?
42. What are the measures of reliability of the MVS Operating System?
43. What are the functions of the Job Entry Subsystem?
44. What are the different stages in JES Processing?
45. Compare and contrast JES2 and JES3.
46. What is the function of an Initiator?
47. What do you mean by the terms "System Generation" and "System Initialization"?
48. What are the steps required in the initialization process?
49. What are the additional system libraries that are used by MVS during System Initialization?
50. What are the hardware requirements of MVS for System Initialization?

3. Data Set Management, Performance, Recovery and Security

3.1 Access methods and Data set management

Access methods are the MVS programs of routines that get control when a program issues on OPEN, CLOSE, GET, PUT, READ, or WRITE macro or high level statement. Access methods give MVS programmers and analysts a very flexible environment to develop systems.

3.1.1 Sequential Access Method (SAM)

SAM files have records placed in a physical rather than logical order. Sequential files are created one record after the other. Magnetic tape, cards, and printed output are sequential files. Direct-access devices may contain sequential files. Sequential files are usually written and read one after the other.

MVS defines five types of record formats.

Fixed Unblocked

In a fixed unblocked file, all records are of the same length (hence "fixed"). Each physical block contains one logical record. The physical and logical sizes are same.

Fixed Blocked

In a fixed blocked file, each physical block contains an integral number of logical records. All logical records must be the same size. Two reasons exist for blocking tape or DASD files: a) maximize the efficiency of the storage media. Efficiently blocked files have fewer gaps. Fewer gaps mean more of the magnetic medium is used for data. b) Speed up data transfer.

Variable Unblocked

In a variable, unblocked file, every logical record in the file can be of different length. Because of this, there is a four-byte record length (RL in front of each logical record. Each physical block also has a four byte block length (BL) as its first four bytes. Because it requires eight bytes of "lengths" for each variable record, this format is not recommended.

Variable Blocked

In a variable blocked file, each physical record contains more than one logical variable record.

Undefined

Undefined records are of any length. The length is determined by the programs that read and write the records. With advanced assembler language using execute channel program, the program can read a block and determine how many bytes were read after the block is read into storage.

Fixed Unblocked

1 Logical 1 Physical

Fixed Blocked

Physical

Logical	Logical	Logical	Logical

Variable Unblocked

68 BL	64 BL	Record
4	4	100

Variable Blocked

BL	RL1	Rec1	RL2	Rec2

Undefined

Fig.3.1 Sequential files. *Sequential files are written block by block from the start of the file to the end of the file. The logical records may be blocked (more than one logical record per physical record) or unblocked (one logical record to one physical record)*

3.1.2 Partitioned Access Method

It is also known as Partitioned Data Set (PDS). PAM files have the records grouped into independent segments called members. Each member has a name, and the name is up to eight characters, the first must be alphabetic. It is basically a data set, which contains suballocated sequential data sets, except that the sub allocated data sets are called members. A partitioned data set is actually two data sets in one. The first directory - consists of 256 byte fixed length records, which contains information about each member. At the beginning of a PDS, directory entry provides information about a member, including its location.

A PDS, which is also called a library or directory, is subject to certain restrictions and peculiarities:

- The members of a PDS are always alphabetically organized.

- The size of directory is fixed at the time of allocation and cannot be changed unless the file is copied to another file with a different sized directory.
- When a member of PDS is deleted, its entry is logically removed from the directory but its space becomes unusable until the PDS is compressed.

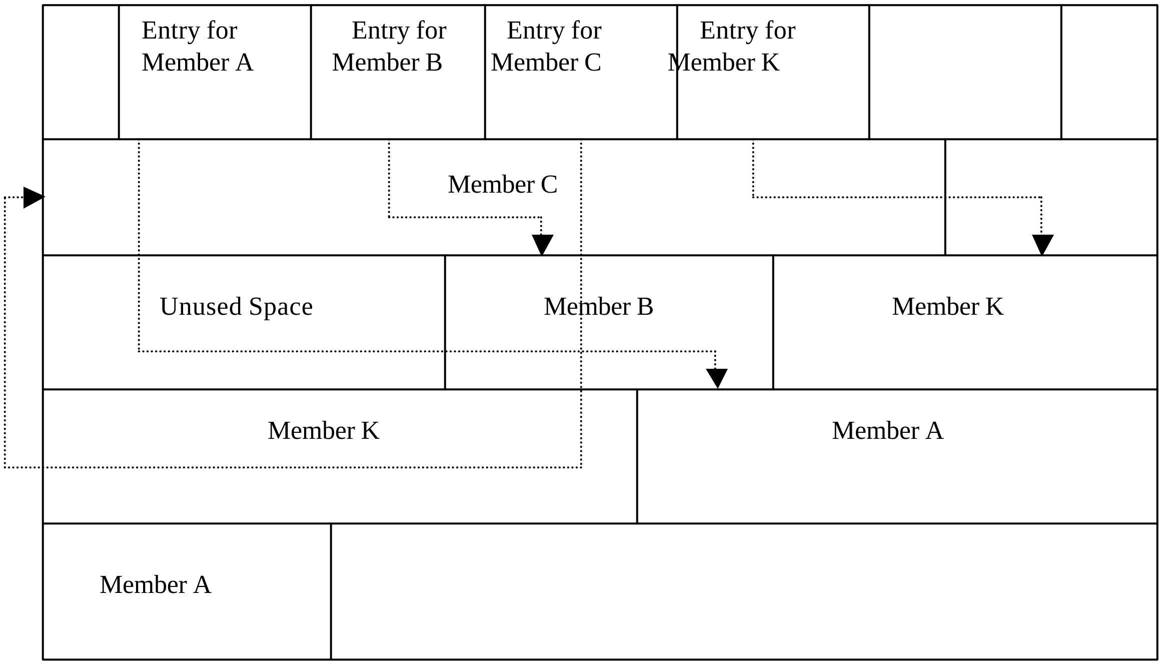


Fig. 3.2 Partitioned Access Method

- A PDS cannot be multivolume. It is always confined to a single volume.

When a member of a PDS is replaced, the replacing member will be placed at the first available position at the end of the PDS and the space of the replaced member is also unusable until the PDS is compressed. (PDS compression means to copy the members into another PDS, so that the holes created in between the members can be removed. The original PDS is deleted).

Partitioned data sets are very common for several reasons:

- ❖ A cataloged procedure is always a member of a PDS, commonly referred to as a procedure library.
- ❖ An executable program, also called a load module, is always a member of a PDS called a load library.
- ❖ Each TSO/ISPF user is the owner of an ISPF library, as mentioned before, is a synonym for a PDS.
- ❖ JCL submitted for execution via TSO/ISPF is invariably a member of a PDS. This is not obligatory, but it is convenient, and as a result, a widespread practice.

Data Set Definition and Access

MVS provides services to create and access "Data Sets" which are areas on DASD or tape that are files of information kept by the data center and its users.

Space Allocation on DASDs

- *Contiguous*

Refers to a set group of consecutive items. If we needed 10 tracks, the tracks, the tracks could be supplied by providing tracks 3 through 12. That allocation contains contiguous tracks, but they are not contiguous.

- *Extent*

Refers to contiguous data space on DASD. In the above example, one extent would be the tracks 3 through 12, but when MVS supplies tracks 1 - 5 and 8 - 12, that is represented by two 'extents'. The VTOC is always a contiguously allocated single data set.

- *Cylinder Allocation*

Refers to allocating data sets in whole cylinder units.

DASD Volume Table of Contents (VTOC)

The Volume Table of Contents is the 'data set' on each MVS DASD volume that contains the name of every data set that is on the volume. The VTOC can be anywhere on the volume, but some usual locations are:

- *At front of the volume*

Many volumes have the VTOC starting at cylinder 0, head 1. The reason for this is that the VTOC is inspected each time OPEN is issued for a data set on the volume. The VTOC is read to determine the location of the data set. The front is recommended for volumes with a few large data sets and for which total number of data sets is less than 40.

- *At one-third of the volume*

One third seems to work well for relatively busy volumes.

- *At the one-half point*

Some very busy volume due to allocation gets better performance when the VTOC is at the halfway point. On the average, the VTOC will have to be read and then a data set located. If the VTOC were at the midpoint, then an average of one-fourth of a volume would have to be spanned by head movement.

VTOC contents

The VTOC contain several types of records. All records in the VTOC are 140 character long, and are in two parts. The first part of a VTOC record is 44-byte physical key (separate from the data portion) which contains the record type, data, or a data set name. The second part is a 96-byte data component. The records are called Data Set Control Block (DSCB). Observing the value stored in the 44th byte (i.e., in fact 45 byte since counting starts from 0) in hex will determine the type of the record. For example x 'F4' implies it is a format 4-type record.

Different types of format

Format 0

This is an available DSCB in the volume i.e., an available VTOC record that could be used to store the information of the data set. If no record of this type is available, then the creation of the data set on this volume is not possible, because VTOC may not be extended as other data sets.

Format 1

This format holds the information of the data set, i.e., its name in key part, and in the data part it can hold the information of 3 extents. If data set has more space then it may contain the pointer to the Format 3 record, which would contain the information of the additional extents.

Format 2

Used only with ISAM.

Format 3

This format contains the information of additional space allocated to the data set. This can hold the information up to 13 extents only. This means that including format 1, format 3 can support only 16 extents for any data set. This limit is for a volume. Hence if the data set is spanned across over more than volume, it can have extents per volume.

Format 4

This DSCB describes the VTOC itself. Format 4 is always the first record in the VTOC, and there is only one Format 4. The record describes the VTOC, the location on the volume and absolute address of the VTOC.

Format 5

Describes the free space available on the volume. This format can hold information of 8 extents in key part and 18 extents in the data part. This format also contains the pointer to the next Format 5 record.This is the way a linked list is maintained.

Format 6

In previous IBM operating system, cylinder could be shared by data sets. This was too complex to implement, and support is dropped for split cylinder by MVS.

3.1.3 Allocation

A large number of MVS modules are the software support for data set creation and deletion. The term allocation is used for this function. This part of MVS that is responsible for allocation of data sets on DASDs is called Direct-Access-Device Management (DASDM). DASDM routines are primarily concerned with

- ♦ Allocation of primary space for new data sets on DASDs.

- ♦ Allocation of secondary space (extending the data set) for program that have written to the end of the current allocation and would like to write more.

- ♦ Releasing DASD space

- ➢ Freeing all the space associated with a DASD data set if the data set is scratched.

- ➢ Freeing space at the end of the data written if the user has specified RLSE for the data set in the JCL.

Types of Allocation

Cylinder means that space should be in units of cylinders. It also states that the data set must begin and end on the cylinder boundaries.

Track means that the number of tracks specified are to be provided but may start at some track that is not at the beginning of a cylinder.

Block length means that the allocation is in number of block times this blocks length. For example, if a 3380 track can hold 10, 4096-byte blocks and the number of blocks wanted are 150, and then 5 tracks should be allocated. If the user also wants to have the data set begin and end on cylinder boundaries, the ROUND parameter will tell DADSM to allocate one cylinder. (ROUND is used to get cylinder allocation for performance while specifying space in terms of block length and number of blocks).

Primary and secondary allocation specification

- ♦ Primary allocation amounts are the space that DADSM is to find before the task can start. The primary allocation amount is also required for secondary volumes. DASDM will be successful if the amount given can be obtained within the five extents. When a data set is first allocated, it would be nice to get the entire primary allocation as the first extent. In the real world, volumes become fragmented. DASDM tries to allocate the space in the following way:

- ➢ Find the smallest contiguous area that satisfies the request.

- ➢ 'Builds" noncontiguous area from no more than five available extents.
- ➢ Stops and forces an ABEND if the primary space cannot be found.

- ♦ Secondary allocation amounts are what is specified by the requester for DASDM to use or get space if the requester is writing to the data set and there is no more room in the current file allocation space. The program can ABEND in several situations:

- ➢ No secondary space was specified.
- ➢ The volume that the data set is on does not have any free space.

The data set has reached the maximum number of extents (16 in the case of non-VASM data sets).

3.1.4 ISAM and VSAM

ISAM records can be writtn in sequence according to a key that is part of record. ISAM files must be on direct access devices, although copies of the files may be unloaded (sequentially) onto magnetic tape. An index or set of indices maintained by MVS gives the location of each record. The user can go directly to a record or sequentially process the record.

There are three types of areas in an ISAM file:

1. Indices. The cylinder index is the highest level of index and is always present. Records in the cylinder index point to track indexes. There is only one cylinder index for file.Opionally, one to three master indexes can be created if the cylinder index is too largee (over four tracks)

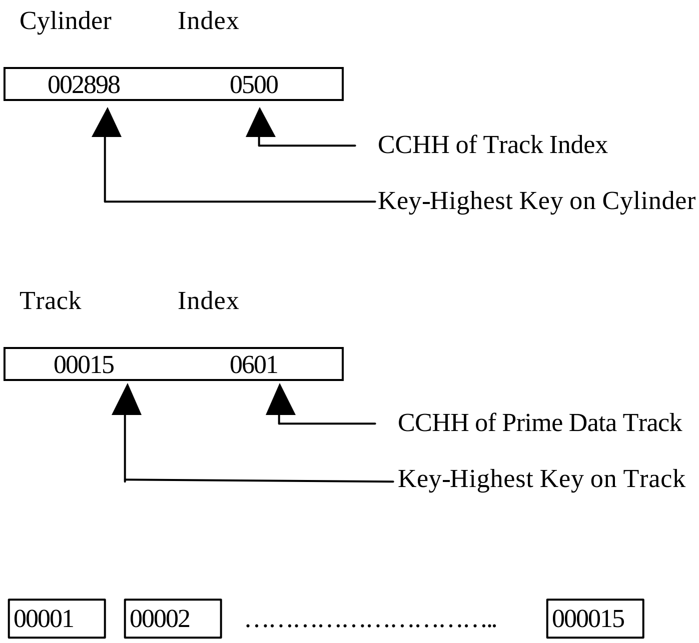


Fig 3.3 Indexed Sequential Organization. An IASM file consists of a cylinder index, a track index. Prime Data Area, and an Optimal Over Flow Areas

Figure 3.3 shows the contents of a cylinder index and the reason why ISAM is a such a problem to a data center. The index has a pointer to the highest key and the absolute track address of the track on which the key exists. If the data center or user wishes to move this data set, it must use ISAM access methods, not dump/restore techniques, because the contents of the record describe the physical address of the data. In order for a dump/restore or archive program to back up and restore this data set, the data set must be restored to exactly where it was dumped from. These restrictions are almost impossible to meet in an active data center.

2. Prime data area. The prime area holds records. The prime area must be formatted with hardware keys.
3. Overflow area. The two type of overflow areas are cylinder and independent overflow areas. Records are written to these overflow areas as additions are made to the file. Cylinder overflow areas are tracks within cylinder. It is best for performance if the extra added records can be written to the cylinder overflow, because the access method does not have to move the DASD read/write head to get tothe independent overflow areas.

The independent overflow area is at the end of the prime area. There is no index into the overflow area. there records are read sequentially until the proper record is found. As records are added to the independent area, the read/write heads move over ever-increasing distances, and the channel path gets more congested.

Virtual Storage Access Method(VSAM)

VSAM supports both sequential and direct processing. It was designed to replace ISAM. VSAM is component of MVS (and DOS/VSE). In Virtual Machine (VM), the CMS/VSAM support is based on DOS/VSE.

VSAM Design Objectives

The VSAM organization was originally presented to replace all data set organizations. This did not occur. VSAM accounts for a large part of he data sets created, but sequentiallly and partitioned data set remain as popular as ever.

1. Operating system independence. The data should be portable from DOS/VSE to MVS/XA to MVS/ESA.
2. Device independence. The access method should allow the data sets to moved from device, yet exploit the random nature of DASD.
3. Data Integrity. The access method should be insensitive to unschedule operating system outages.
4. Data security. Changes to user files and catalogs should only be performed by “authorized” users and data should not be “lost”
5. File performance. Insertion activity should not slow down as more inserts are done (ISAM problem)

VSAM Features. How well did VSAM live up to its objectives?

Data Portability

VSAM can be moved from one operating system to another, but in practice few data centers move data among DOS/VSE, MVS/370, MVS/XA and MVS/ESA Processor Complexes.

Data Independence

VSAM is independent of track sizes. VSAM stores records of all types of data in a fixed length area of DASD called control interval (CI). By having fixed physical blocks, the units of transfer are quickly copied to almost any DASD volume.

Data Integrity

How well does VSAM protect data integrity? VSAM has a low vulnerability during record insertion. The potential remains to lose records, but the portability is lower than ISAM. Data sharing is limited. The user understand VSAM data sharing is limited. The user must understand VSAM data sharing in order to share a file among many users. VSAM implements **Software End-of-File**, which defines where the "add" process was and makes it possible to recover the file if the system or program "crashes" or abends.

Security/change control: How well does VSAM rate in security and change control? VSAM provides centralized control over:

1. File Control
2. File Access
3. File Deletion
4. Space occupied by files

VSAM provides a multilevel password facility, but it is not recommended that you must use VSAM passwords. Use a comprehensive security product such as RACF.

VSAM provides an excellent audit trail of actions taken on data sets. VSAM writes records to the System Management Facility(SMF) each time an adderess space creates, deletes, or changes a VSAM data set. SMF type 60-63 records are created by VSAM. Type 60-63 are for audit purposes. Type 64 is for each open.

Consistent Performance

How well does VSAM give consistent performance to overcome the problems with ISAM insertions? VSAM has several free space options which can reduce record insertion delays.

IDCAMS - VSAM *utility program:* IDC Access Method Services (IDCAMS) is the IBM utility manage VSAM data sets. It is also called "Access Method Services". The major functions of IDCAMS are:

1. Create "objects" which are:
    - a) System catalogs to keep track of:
        - i) User files
        - ii) MVS page spaces and swap data sets
    - b) User files
2. Move data from one device to another:
    - a) For file load and unload
    - b) For file backup and recovery
    - c) For catalog backup and recovery
    - d) For file copy
3. Miscellaeous functions
    - a) List file characteristics
    - b) List file contents
    - c) Delete file/catalog
    - d) Alter file characteristics
4. Assign and remove securities for:
    - a) Catalogs
    - b) User files

Uses for VSAM: VSAM is the choice for most of data centers for high performance, random access applications. VSAM is ideal for CICS, IMS, and batch databases, but VSAM is also used by IBM for a large part of the data that MVS creates and/or uses.

Critical MVS functions: VSAM is used for many MVS functions, and each release of MVS finds new uses for VSAM data sets. The critical MVS uses of VSAM are:

1. The Master catalog. MVS cannot operate without the master catalog. It contains the pointers to the systems data sets and pointer to other catalogs which contain pointers to all the data center data sets.

2. Page and swap data sets. These are VSAM data sets but are "special" type of VSAM. Special parameters tell IDCAMS to format the page and swap data sets so that the Auxiliary Storage Manager can use them for 4,096-byte page slots.

3. The SMF data set is a VSAM data set. SMF record sizes vary widely, so VSAM is an ideal data set method. Some SMF records are several hundred bytes. Some SMF records are thousands of bytes long.

Space Allocation

Control Interval (CI) is a basic unit of VSAM data set. CI contain one or more than one Logical Records(LR). A group of CI called Control Areas. VSAM data set is one or more Control Areas.

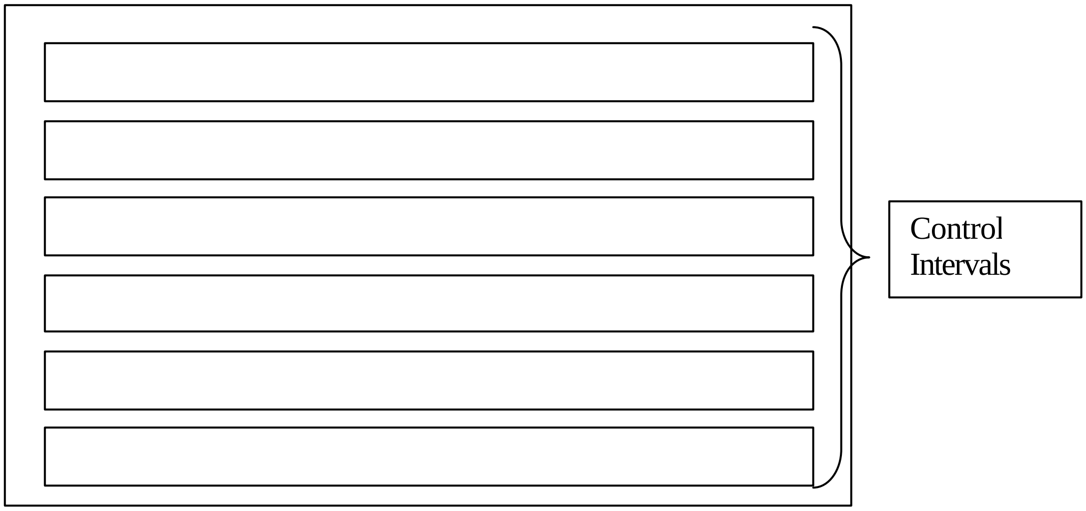


Fig 3.4 : Control Area and Control Intervals

File Support for High Performance Applications:
VSAM supports following data sets:

- ESDS: Entry- Sequence Data Set
- KSDS: Key- Sequence Data Set
- RRDS: Relative Record Data Set
- LDS: Linear Data Set

ESDS vs Sequential:

ESDS files do not have prime index associated. When a record is added VSAM returns "Relative Byte Address" (RBA). The programmer analyst must keep track of the RBA if direct address is needed to a ESDS. ESDS should not be used to replace sequential data sets. Traditional SAM is generally faster, has possibility of loading more data on a track of DASD, and is easier to use. One reason to use ESDS is it provides Secondary or Alternative index into the data.

L R 1	L R 2	Unused Space	R D F 2	R D F 1	C I D F

Fig 3.5 : Structure of Control Interval for ESDS

Control Interval Definition Field(CIDF): 4 bytes long. It contains information about the CI, including the amount and location of unused space.

Record Definition Field(RDF): 3 bytes long. It describes the length of records and how many adjacent records are of the same length.

A CI is a unit of information that VSAM transfer between virtual storage and disk storage. The size of CI's can vary from one data set to another, but all CI's within the data portion of a particular data set must be of same length.

RRDS
A Relative Record Data set has no index. The file contains a string of fixed length slots. Each record written and read based on the record number.

RR1	RR2									RR11	RR12	RR13

Fig 3.6 : Structure of Control Interval for RRDS

First record will be placed in first slot, second in second slot and so on. If a particular record pertaining to any record no is not there than that slot will be left empty.

KSDS:

A Keyed Sequential Data Set (KSDS) is always defined with Index. The index consists of set, which may contain more than level. Index records contain Information about the next lower level. The lowest level is called sequence set. The sequence set contains pointers to several control intervals. The sum of Control intervals pointed to by a single sequence set is called a Control Area.

VSAM uses vertical pointer from the highest level to the Sequence Set to get the address of Control Interval for VSAM records. VSAM uses horizontal pointer in a sequence set when it is reading sequentially and needs the next key in a collating sequence.

VSAM performance compared to ISAM are:

1. Better by about 4:1 for record inserts
2. Better by about 2:1 for random direct-inserts
3. About the same for sequential reads
4. About the same, may be a little slower than ISAM, for file loading.

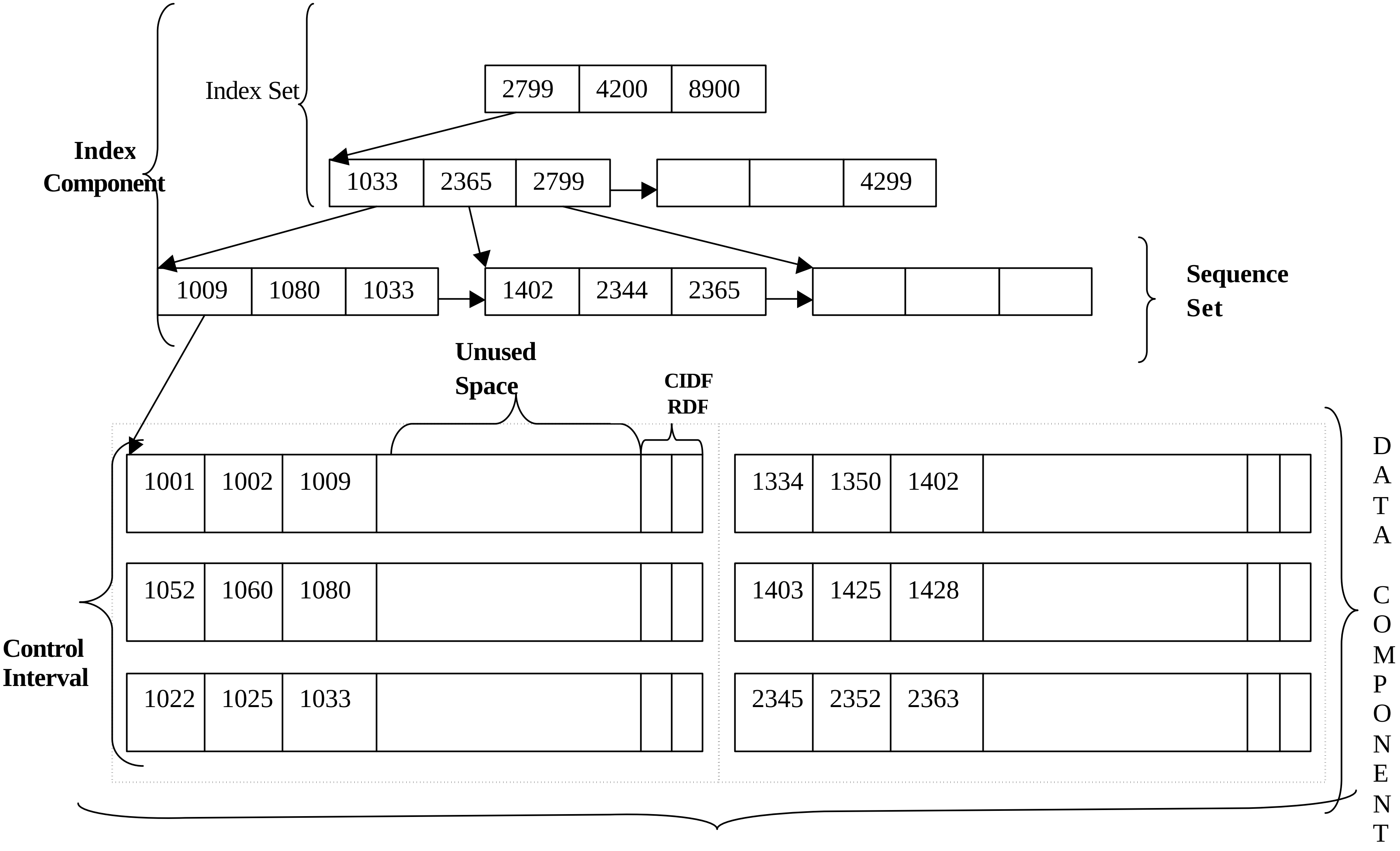


Fig 3.7 Structure of Control Interval for KSDS

Alternate Index (AIX): An Alternate Index performs the same functions as a prime index of a data set. The data set itself is referred to a base cluster. The base can be a KSDS or an ESDS. The AIX is similar to a cluster. It contains an index component and a data component. The index is identical in format to any other KSDS. The data component contains pointers to the base cluster.

3.2 Performance Tuning

VSAM performance evaluation is appropriate whenever there is a change in system hardware or software. The different parameters that can be tuned are Buffer space, Control interval size and Free space (CI- percent & CA - percent)

3.2.1 Control Interval Size

Choice of control interval size depends on the kind of processing performed and device characteristics like Direct processing and sequential processing.

- For sequential processing, smaller data CI sizes are desirable. A 2K data CI, for example, requires twice as many I/Os as a 4K data CI.
- For random or direct processing, smaller data CIs are desirable. Larger CIs require longer data transfer time without benefit, since a new CI is usually required for each record read.
- For data sets accessed both randomly and sequentially, a data CI size between 2K and 8K is desirable.
- Tune to maximize direct processing performance.

3.2.2 Free Space

Free space should be specified whenever there is an insert activity within a data set or modification to the length and variable length and variable length records are expected. Free space is used to reduce the number of CI and CA splits.

Large Free Space

- More DASD space.
- More I/O for sequential processing for same number of records.
- More levels of index, so possible increase in run time for direct processing
- No free space or too little free spaceNO FREE SPACE OR TOO LITTLE FREE SPACE
- More CI/CA splits
- After splits, possibly more time for sequential processing when file not in physical sequence.

3.2.3 Buffer Space

Too many buffers

- Additional virtual storage.
- Additional system paging.

Too few buffers

- Poor performance.
- Multiple DASD accesses.

3.3 Security

IBM released MVS in 1974 and was the largest and most complex operating system ever produced. Since then, it has grown several times larger. The predecessor of MVS - MVT - relied on the PASSWORD facility to provide security. Passwords could be applied to non-VSAM data sets, VSAM data sets, and TSO Logon user-ids, but these were too cumbersome and not really secure.

IBM developed the Resource Access Control Facility (RACF) as an add-on to MVS. OEM developers created ACF2 and Top Secret (both now owned by Computer Associates International). These security systems provide the security tools, which the data center needs to implement a security system.

IBM is not the only one to open the data center to violations. OEM software manufacturers usually provide an SVC as part of their database or other subsystem. In at least one instance, an OEM vendor left the interface of the SVC such that anyone could get into Supervisor State and Protection Key zero with only minimal knowledge. *If a program achieves Supervisor State, even RACF-like protection is worthless.*

A method of extending the security offered by MVS is to encrypt the data. IBM provides cryptography hardware and software to change the EBCDIC representation of the data to a non-readable form. The user must provide a 'key' to decode the data. Even so, that key must be in storage sometime, however brief, to decrypt the data and a program that can achieve protection key zero could possibly steal the key.

Security is never absolute: The probability of a compromise may approach, but is never, zero. Securing a system is the act of moving that probability closer to zero than it was before.

3.3.1 Preparing for Security

The data center and its users should decide on the security required for the data and facilities provided by the data center. Almost always, a security system such as RACF should be in place, if for no other reason than to protect the investment the data œnter has in the MVS operating system. The following are a few of the considerations, which should be covered in preparing for security in a data center:

1. Obtain an agreement on the goals of a Security system

➢ *With management:* The management of the data center and/or the company as a whole should decide broad security goals. All security costs something. Near-total security will cost the company a huge amount of money. No security will open the company up to potentially large expenses and possibly legal violations and law suits. Somewhere in between is a security system which management is willing to pay for and live with.

➢ *With data processing:* The data processing department should require security to prevent unauthorized access to programs and files.

➢ *With the user department:*. If the user department understands it is making decisions and investing time and money in the information coming from the programs and files at the data center, it should be willing to pay some of the 'costs' of a serrate system.

2. Identify and train the key members of the security team.

3. Develop and publish a security policy, including procedures for all to use. Technology such as RACF is only part of the answer. The primary security resources are the people the technology exists not only to elicit the machinery's assistance but also as a means of galvanizing support from the people entrusted with the machinery.

4. Develop and publish an implementation plan.

5. Implement.

6. Monitor. This is usually forgotten in most data centers. Violations occur in any security environment and many fall into the "ok if it happened before" category. Examples are application programmers who forget to change production names to test names and are ABENDed because they cannot update production files. In some cases, people and departments are attempting - or succeeding - to access data they should not. If the data center does not have a security administrator and allocate time to monitor security, then the security is not complete.

Classification of Job Functions: The job classifications of the company are a good place to start defining security job classifications. Systems Programmers, Operators, Production Control, Application Programmers, User departments, and hardware and software vendors have different needs and responsibilities. These area definitions are a good place to start defining job functions.

The personnel department may want to mention security in job description, even for those positions not specifically charged with maintaining it, so newcomers are automatically sensitized to the desire for a secure environment after it has been established.

Classification of data: Each type of data in the corporation should be grouped into classifications to provide a policy for the data. Each organization is different, but the following given guidelines for areas that will be used for the data.

System libraries: The MVS and data center libraries must be protected to ensure the operating system and data center data sets are not modified without authorization.

User department data: Each department should define the data and requirements. The payroll data should probably not be accessed by anyone except the department and selected others. Accounts payable data may have a wider audience.

Program libraries: These libraries contain the software investment of the corporation. They should probably be protected from read or write access, not only to protect the integrity of the programs themselves but also to ensure that they are not used as 'Trojan horses' to threaten the integrity of the rest of the system.

3.3.2 Obtaining agreement

Obtaining agreement is a very simple process. The following are a list of the key items needed:

- The security policy must be a top priority of upper management. Top priority means a verbal, written, and funded commitment.

- All members of the security committee must understand the policy and understand the need for the policy.

- Resources (time and money) must be available to implement the policy.

3.3.3 Data Security Policy

The policy should begin with a brief statement of the corporate intent. It should clearly state the policy is part of the contract with the user department. Security is a two-edged sword. The policy should state what the corporation is trying to accomplish with the security system.

The data security policy should come from the highest levels of the corporation. Security is a bother. Security is expensive in both system and people resources. Security prevents some people from doing things they might want to do, but do not have the corporate direction to do.

Security should be delegated to the responsible areas. The systems programmers should protect MVS and the associated operating system functions. The accounts payable department should protect the accounts payable files. Audit trails should be in place to ensure multiple administrative safeguards are in place.

3.3.4 Implementation

Implementation Plan

A team of representatives from all departments should accomplish implementation of a security system. It will be difficult to have a team accomplish this complex task, but if it is not a team effort, then problems will certainly occur. The following is a sample scenario for a phased implementation of security:

1.Decide on and install the security system. The IBM MVS system is the Resource Access Control Facility (RACF). OEM software vendors have security system, which will work in place of RACF.

2. Define groups and/or departments - wide areas of responsibility.

3. Define security administrators who will accept responsibility for their department's security implementation.

4. Set up a security subsystem, which includes the above organization. Create names for each group.

5. Define system protection and implement. This is the easiest to check and a very important first step to security.

6. Define applications to start protecting. If the system has a "warning" feature - where the task is warned with a message and not actually cancelled - this is a good feature to run in production to see the effect of the security before it is fully implemented.

7. Implement protection for all departments.

8. Monitor security violation

Resource Access Control Facility (RACF) Flow

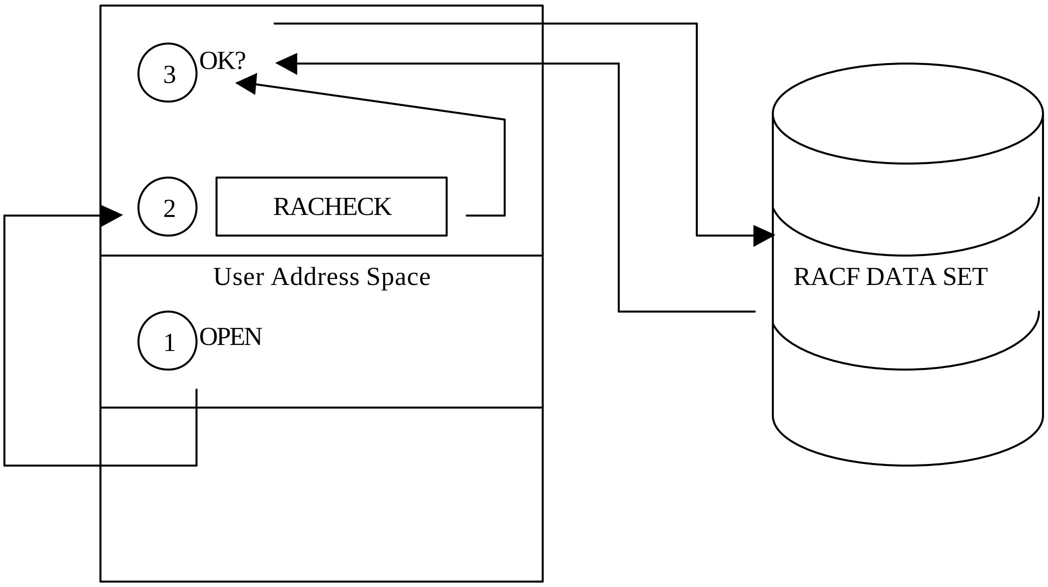


Fig. 3.8 RACF

(*The users address space issues an OPEN request. DADSM issues a RACHECK macro to see if the user can access the data set. If the user can access the data set as outlined in the RACF profiles in the RACF data set, the access is slowed. If not, DADSM is instructed to Abend the task with a 913 Abend.).*

RACF is a combination of modules, which interface with MVS subsystems (allocation, JES etc.) to ensure TSO LOGINs and batch jobs are authorized to operate at the data center. The key interface is the RACHECK macro. In Fig.3.9, the user's address space asks for a resource such as data set allocation. The "application", in this case, is Direct Access Storage Management (DADSM), DADSM issues a RACHECK macro to have RACF (or the equivalent OEM program product) look up the request and allow the access or disallow the access.

Fig.3.8 shows the logical view of RACF processing.

The user address space asks for a service - for example, opening a data set. The OPEN routines issue a RACHECK macro, which calls MVS modules to check the RACF ID against the profile of the resource.

The profile is usually loaded into virtual storage by MVS routines and identifies the classification of the resource and specifies the access limitations to impose on it. Whenever the profile is changed, it must be "refreshed" for the changes to take effect.

The RACHECK MVS modules look in virtual storage or read parameters from the RACF data sets. The user is either allowed to access the resource or denied access.

The resource does not have to be something real like a data set. It can be a made-up name to control access to a system. For example, the resource "payroll" could be created to allow or disallow access to the payroll system in CICS. The security routine could issue a RACHECK macro to see if particular user could access the payroll system.

3.4 MVS Recovery

3.4.1 Goals of MVS Recovery

- Keep MVS functioning
- Isolate Errors
- Free Resources
- Provide Documentation for problem determination

MVS contains many recovery routines that keep the operational. These routines attempt to isolate errors to a particular program or system routine so that the falling program can be removed from the system and its resources (storage, data sets, etc.,) can be freed. Documentation for later problem determination is also provided for the failed component.

3.4.2 Recovery / Termination Perspective

The supervisor initiates the recovery termination process for my program either when I request if (for example, at the end of a job step or when I LOGOFF from TSO) or when the system detects an error condition. The Recovery/Termination manager (RTM) component of MVS cleans up system resources when a task or address space terminates and provides the ability for a program to recover from unexpected errors.

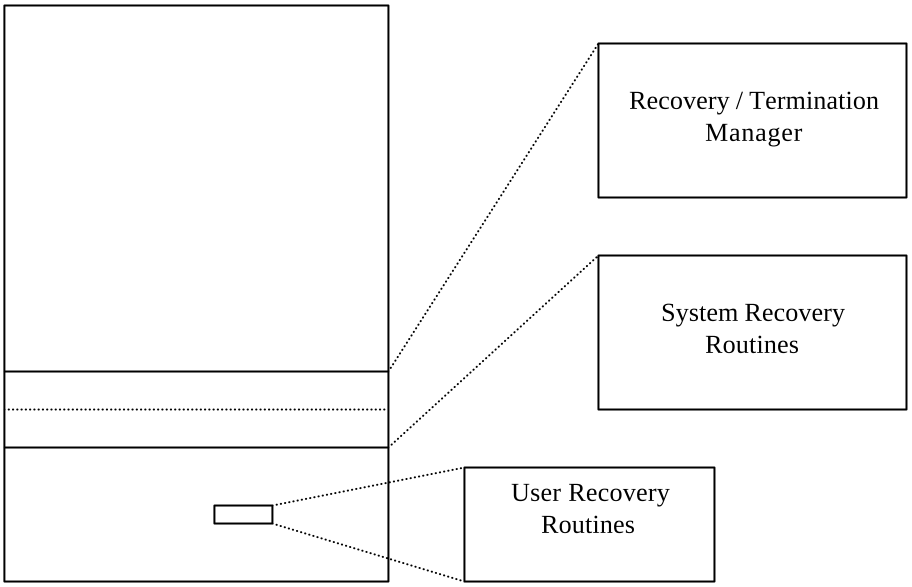


Fig. 3.9 Recovery /Termination Perspective

The RTM component and the MVS system recovery routines are mapped in the common area of the address so that they are available to all users. Subsystems such as IMS and CICS provide additional recovery facilities and application programs can furnish their own recovery routines.

3.4.3 Recovery /Termination Manager

The Recovery/Termination Manager (RTM) controls the flow of software recovery processing by handling all normal and abnormal terminations of tasks and address spaces. The hardware detects certain types of abnormal conditions such as processor failure and causes a machine check interrupt to occur. The machine-check interrupt handler interfaces to RTM to log the error and attempt recovery from it.

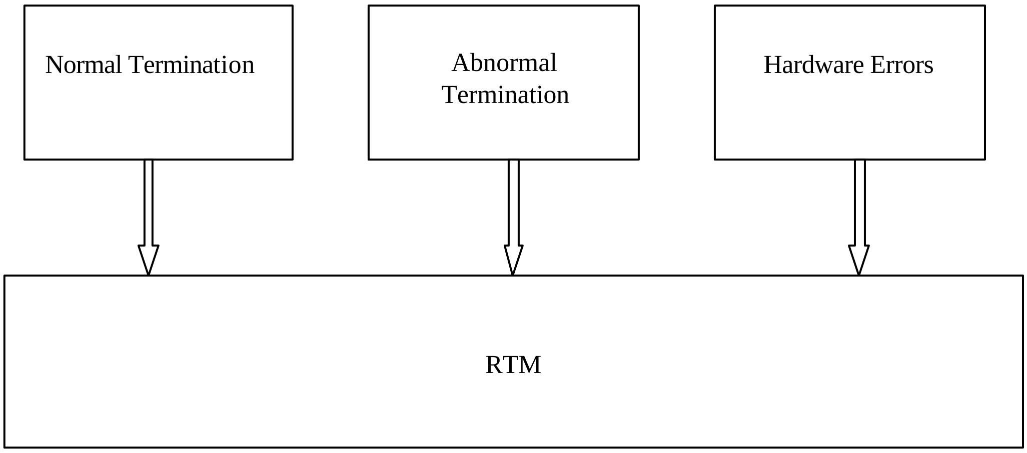


Fig. 3.10 Recovery/Termination Manager

The hardware also detects an abnormal condition such as an attempt to execute an instruction with an invalid address and causes a program check interrupt handler interfaces to RTM to abnormally terminate the program. The software detects other abnormal conditions such as an attempt to open a data set that is not defined to the system and causes abnormal termination. RTM selects the appropriate termination or recovery process according to the status of the system.

3.4.4 Hardware errors

Hardware errors in one of the central processors or in central storage or in the channel system will cause a machine check interrupt and the Machine Check Handler (MCH) will initiate recovery.

Some machine checks notify software that the processor detected and corrected a hardware problem that required no software recovery action. This is known as software recovery action. These are known as soft errors and RTM needs only to log them. The hardware or RTM recovers from these errors without effecting the program in execution.

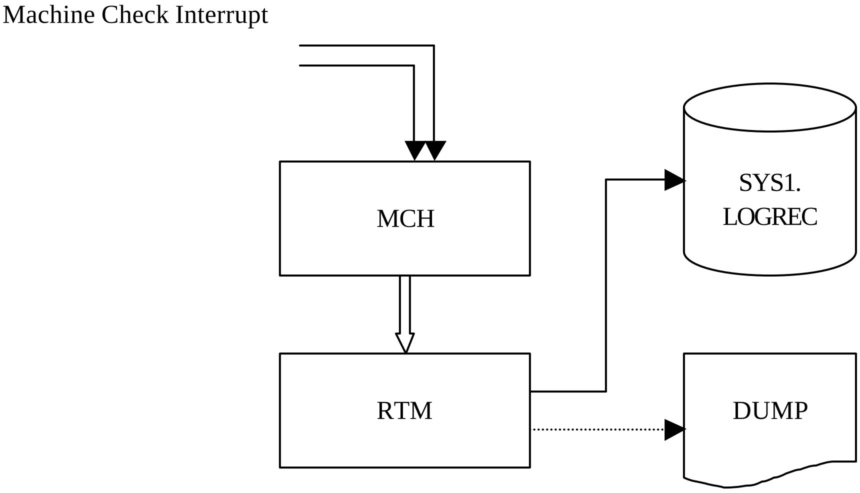


Fig. 3.11 Hardware Errors

Hard errors are hardware problems detected by the processor that require software-initiated action for damage repair. Such errors are probably confined to the program in execution and that program will be abnormally terminated.

When MVS cannot recover from a hardware error, the Machine Check Handler (MCH) assumes the processor must be terminated. In a multiprocessing environment, MCH invokes alternate CPU recovery (ACR) on another processor and takes the failing processor offline.

RTM records errors in the SYS1.LOGREC data set to provide a record of all hardware failures as well as selected software errors and selected system conditions.

3.4.5 I/O Recovery Actions

At predefined intervals, the operating system checks devices of a specific type (for example, at 15 second intervals for DASD devices) to determine if expected I/O interrupts have occurred. If an expected interrupt has not occurred across two of these checks, that interrupt is considered missing. An informational message is issued to the operator, and the operating system tries to correct the problem.

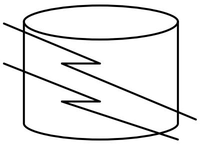
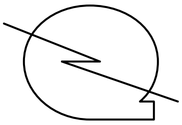
	Missing Interrupt Handler At predefined intervals scan for missing Interrupts
	*Hot I/O* Malfunctioning Device causes repeated Interrupts
	*Dynamic Device Reconfiguration* Operator can invoke DDR to move a demountable volume

Fig. 3.12 I/O Recovery Actions

"Hot I/O" refers to a hardware malfunction that causes repeated, unsolicited I/O interrupts. If such I/O interrupts are not detected, the system can loop or the system queue area (SQA) can be depleted. The I/O supervisor (IOS) tries to detect the condition and perform recovery before the system requires a reIPL. When a device fails, operators can enter the SWAP command to perform dynamic device reconfiguration (DDR). DDR allows the operator to move or swap a demountable volume from a device. When DDR is active, the system dynamically requests the swapping whenever device encounters device errors.

3.4.6 Abnormal termination of a program

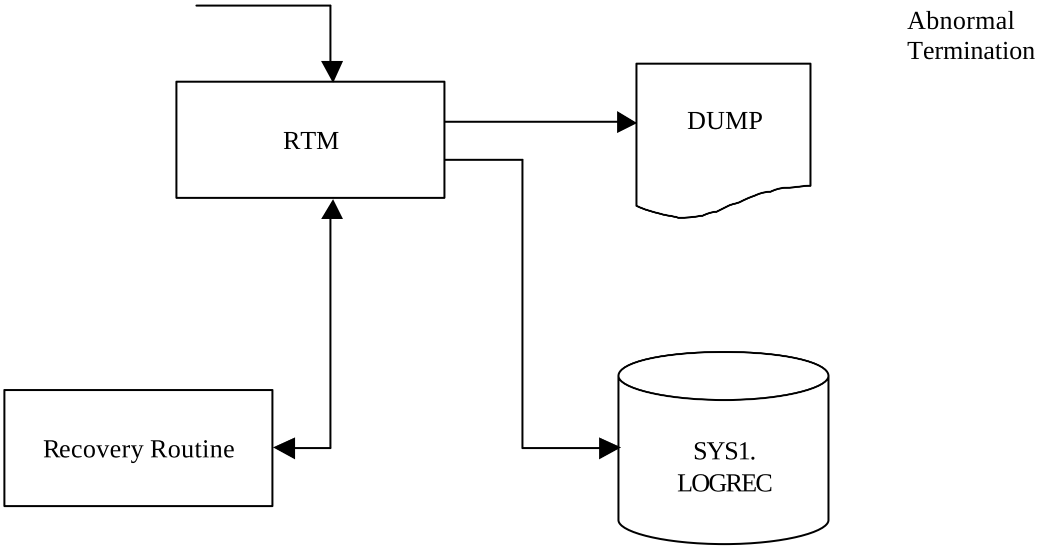


Fig.3.13 Abnormal Termination of a Program

MVS does a great deal of checking for abnormal conditions during the execution of my program. It uses hardware to detect errors such as protection violations or addressing errors. Such error causes a program-check interrupt.

The data management and supervisor routines provide some error checking facilities to ensure that, based on the information provided in the program, only valid data is being processed and that no conflicting requests have been made.

For abnormal conditions that can possibly be corrected, MVS may return to my program with a return code indicating the probable source of the error. MVS contains many recovery routines that keep the system operational.

These routines attempt to isolate errors to a particular program or system routine so that the failing program can be removed from the system and its resources (storage, data sets, etc.,) can be freed. Documentation for later problem determination is also provided for the failed component.

Review Questions

1. What are the different access methods of data sets?
2. What are the different types of record formats in Sequential Access Method (SAM) of MVS?
3. What do you mean by a sequential file?
4. What is the significance of the Partitioned Access Method?
5. What are the restrictions and peculiarities of the PDS created?
6. What do you mean by a load module?
7. How do you allocate the space on the DASDs?
8. What do you mean by an extent on a DASD?
9. What is the significance of the VTOC of DASD?
10. What are the usual locations of the VTOC on the volume of the DASD?
11. Describe the contents of the VTOC.
12. What are the different formats of the VTOC?
13. Which type of format of the VTOC is used with ISAM?
14. What are the types of allocation in DASD?
15. What are the types of VSAM Data Set Organization?
16. What do you mean by Control Interval and Control Area?
17. What are the differences between RRDS, KSDS and ESDS types of VSAM data set organization?
18. What do you mean by a linear data set?
19. What is the advantage of having an Alternate Index in VSAM?
20. What do you understand by the term "ISAM"?
21. What is the importance of Performance Tuning?
22. What are the different parameters that can be tuned to enhance the VSAM performance?
23. What is the effect of buffer space on performance tuning in VSAM?
24. How does having too few buffers degrade the VSAM performance?
25. Substantiate the statement: If a program achieves Supervisor state, even RACF- like protection is worthless.
26. What are the different security features available in MVS?
27. What are the different steps involved in obtaining security to the MVS Operating System?
28. What are the goals of MVS recovery?
29. What is the function of an MVS Recovery / Termination Manager?
30.What is the purpose of a Machine Interrupt Handler in MVS?
31. What are the different possibilities of getting hardware errors?
32. What are the different I/O recovery routines available in MVS?
33. How does MVS handle abnormal termination of a program?
34. State the function of a DDR in MVS recovery.
35. What are the advantages of I/O recovery routines in MVS?

APPENDIX A

GLOSSARY

Above the line: An address greater than 16 Mb. Normally refers to virtual storage but also applies to central storage.

Actuator: The portion of direct access storage devices (DASD) that contains the read/write heads and the disk platters that contain the data. The actuator is a portion of the DASD that is addressable to the data center by volume serial number.

APG: Automatic priority group. The early MVS methods of controlling dispatching priorities of address spaces.

ASCB: Address space control block. The control block which represents an address space to MVS.

ASM: Auxiliary storage manager. The MVS modules that manage the page and swap data sets. ASM builds channel program to transfer pages in and out of central storage.

Assembler: A program that transfers source statements into object code. The IBM assemblers allow “MACROS” which are single assembler statements that result in multiple assembler statements.

AUXILLARY STORAGE: The term used for MVS paging devices, which contain central storage page frames that are not in use.

BCS: Basic direct access method.

BLOCK: A physical look of logical records. Blocks are separated by gaps on recording medium. Blocks usually contain a group of logical records. Hardware is more concerned with blocks, whereas software is more concerned with records.

BLOCK MULTIPLEXER CHANNEL: A part of a processor complex channel subsystem that can keep track of more than one device transferring blocks and interleave blocks on the channel to different devices.

BPAM: Basic partitioned access method. The MVS access methods, which is used to update partitioned data sets.

BSAM: Basic sequential access method. An MVS access method, which is used to read from or write sequential files. BSAM does not block or deblock records.

CACHE: A random access memory used to buffer data from larger storage media. Whenever data is requested, the microcode first checks the cache.

Central storage: A term for the random-access, read/write memory inside a processor complex.

CICS: Customer information control system. CICS accepts transaction codes from teleprocessing terminals and calls programs to process the transactions.

CLPA: Create link-pace area. An IPL parameter used to copy SYS1. LPALIB to virtual storage.

DASD: Direct access storage device. An I/O device, which allows modules to read or write specific, blocks anywhere in a fife.

DAT: Dynamic address translation. The term used for the combination of MVS software and control blocks and the system/370 hardware to use a multi-level table lookup to convert virtual address to addresses in central storage as instructions are executing.

DYADIC: A term for “multiprocessor” which the IBM uses for processor complexes with multiple CPUs. A dyadic has two CPU’s and both have I/O capabilities.

ECB: Event control block. The MVS control block used to communicate between MVS and application or system modules for timer interventions, i/o requests, or other “events” at some future time.

EFLPA: Extended fixed link pack area. The MVS/XA and MVS/ESA area in virtual storage for the `above the line’ extension of the fixed pack area.

GDG: Generation Data Group. A special data set naming convention that has MVS append sequence numbers to the name and keep track of generations.

HDA: Head Disk Assembly. The portion of Direct Access Storage Devices (DASD) that contain the physical media that has data recorded on surfaces of a spinning disk.

HIPER Space: An address space that can exist under MVS/ESA that contains only data.

Hit Ratio: The result of dividing the number of Input/Output accesses that are “found” in the cache storage of a control unit by those, which are not.

ICE: Integrated Catalog Facility. The term used by IBM for the third generation of VSAM catalogs.

IDCAMS: The IBM utility which reads control statements and performs data set functions such as creation, deletion, cataloging and uncataloging.

Initiator: A set of MVS modules to which JES assigns jobs. Each initiator operates in its own address space and takes control of the address space before and after each job step executes.

IML: Initial Microprogram Load. IML is loading the microcode into the Processor Complex to enable the Processor Complex to process instructions.

IMS: Information Management System. One of IBM's large DB/DC application systems.

IOGEN: The term used to describe the I/O configuration to the MVS Input/Output Supervisor.

ISAM: Indexed Sequential Access Method. A very old access method watch supports both sequential and direct access by key to records in its files.

ISPF/PDF: Interactive System Productivity Facility/Program Development Facility

JCL: Job Control Language. The term used for a structured language, used by MVS to execute a unit of work.

JES: Job Entry Subsystem. The name used for the MVS modules which read jobs, interprets their JCL, Schedule execution, spool the print and punch output they generate, and finally produces that output at the appropriate destination.

JES2: Job Entry Subsystem-2 IBM rewrote portions of HASP for MVS and called the new subsystem JES2.

JES3: Job Entry Subsystem-3 IBM rewrote portions of HASP for MVS and called the new subsystem JES3.
KSDS: Key Sequenced VSAM Data Set.

LPAR: Logical Partition. The VM/XA feature that allows a 309x Processor Complex to be logically partitioned to divide up the processor complex for multiple System Control Programs.

MFT: Multiprocessing with a Fixed number of tasks. One of the first System /360 operating system.

MIPS: Millions of Instructions Per Second. A commonly used measure of CPU power.

Multiprogramming: Refers to a technique or computer system capable of sharing a single CPU's instruction stream with more than one thread of execution by periodically interrupting an executing program and allowing another to continue.

MVS: Multiple Virtual Storage.

MVT: Multiprocessing with a variable number of Tasks. One of the early Operating systems for the System/360 Processor complex.

Nucleus: That part of a system control programs which is always resident in Central Storage.

PLPA: Pageable Link Pack Area. An area of Virtual Storage, which holds modules, used by more than one address space at a time.

PSW: Program Status Word. The PSW is an 8-byte control register, which contains the address of the next instruction to be executed and information about the task.

RACF: Resource Access Control Facility. An IBM program product, which controls user access to resources such as data sets, and TSO Logon Ids.

SAM: Sequential Access Method. The name of the collection of access method modules which process sequential data sets.

SIGP: Signal processor. A MVS/370 instruction which allows a CPU to force another to perform one of a number of operations.

SMS: System Managed Storage, the MVS/ESA implementation of storage management that implements allocation by data center specifications, not JCL.

SQA: System Queue Area. The MVS virtual storage used to create control blocks, which must be both fixed and globally addressable.

Subsystem: A defined group of MVS modules which control access to their address space through the subsystem, communications, vector table and processes request from the IEFSSREQ macro.

SYSGEN: System Generation. The process used to customize MVS to the data center's requirements.

SYSRES: System Residence. The DASD volume that contains all information about a single task.

TSO: Time Sharing Option. IBM subsystem for interactive access to MVS services

UCB: Unit Control Block. The MVS control block that controls each control unit.

UCW: Unit Control Word. The special register in the channel subsystem that stores the status of each I/O request as it progresses from start to completion.

VSAM: Virtual Storage Access Method. An MVS access method for DASD data sets which supports entry sequenced, relative record and key sequenced files.

VTAM: Virtual Telecommunication Access Method. The IBM subsystem which controls local and remote terminals for application programs or allows communication. VTAM supports systems network architecture for MVS.

VTOC: Volume Table of Contents. An area on each DASD volume, which contains data set names and pointers to all extents of all data sets in the volume.

VVDS: VSAM Volume Data Set.

XA: Extended Architecture. MVS/XA introduced the ability to address two gigabytes of virtual and central storage, improved I/O processing, and improved reliability serviceability and availability of the operating system.

APPENDIX - B

Working with ISPF

How to access ISPF

Although the exact procedures used to access ISPF vary from one installation to the next, three things should always happen. First, you need to establish a session between your terminal and TSO. Second, you must identify yourself to TSO. And third, you must start ISPF. Typically, the first two steps are combined by entering a single LOGON command that both connects and identifies you to TSO. Once you have gained access to TSO, you usually start by entering the userid and then the password.

- If you are a valid user you will be interact with a screen as given below which is the ISPF Master Application Menu.

The format of ISPF panels

```
                        ISPF Master Application Menu

 P    PDF               PDF Applications                       Userid . : INF6244
 SD SDSF                System Display and Search Facility   Time . . : 07:55
 S    SORT              DF/SORT Dialogs                        Terminal : 3278
 FC FilAID & CA         File AID Tools / IDMS                  Pf keys  : 12
 N   CA-Panvalet        Browse, Edit, and Utilities            Screen . : 1
 6    TEST              FOR COBOL CICS DB2                     Language : ENGLISH
 X   Exit               Terminate ISPF using list/log defaults Appl ID  : ISP
                                                               Release  : ISPF 4.5


Enter END command to terminate application

 5655-042 (C) COPYRIGHT IBM CORP 1982, 1996
 Option ===> P
  F1=Help    F3=End     F4=Return   F5=Rfind    F6=Rchange  F10=Left
 F11=Right   F12=Cretriev
```

- You will find some choices and an option field to enter the choices. There are some special keys known as Attention Identifier Keys ex. F1, F3 etc at the bottom of the screen which perform some special task that can be identified by a small mesg. On the right hand side of each Keys ex. F3=End. On pressing F3 you would end the ISPF session.
  If you give the option **P** you will interact with the ISPF Primary Option Menu

```
Menu  Utilities  Compilers  Options  Status  Help
------------------------------------------------------------------------

ISPF Primary Option Menu

0  Settings        Terminal and user parameters          User ID . : INF6244
1  View            Display source data or listings       Time. . . : 08:17
2  Edit            Create or change source data          Terminal. : 3278
3  Utilities       Perform utility functions             Screen. . : 1
4  Foreground      Interactive language processing       Language. : ENGLISH
5  Batch           Submit job for language processing   Appl ID . : PDF
6  Command         Enter TSO or Workstation commands     TSO logon: IKJINTST
7  Dialog Test     Perform dialog testing                TSO prefix: INF6244
8  LM Facility     Library administrator functions      System ID : INFOR5
9  IBM Products    IBM program development products      MVS acct. : #ACCT
10 SCLM            SW Configuration Library Manager      Release . : ISPF 4.5
11 Workplace       ISPF Object/Action Workplace
12 DB2        Perform DB2 Interactive functions

Enter X to Terminate using log/list defaults



Option ===>
F1=Help    F3=Exit    F10=Actions  F12=Cancel
------------------------------------------------------------------------
```

Most ISPF panels have action bars at the top. The choices appear on the screen. And you have the Attention Identifier or the PF keys at the bottom as discussed in the ISPF Master Application Menu.

For Editing a dataset we have to first have a dataset.

To create a PDS i.e.,a directory you will give the option **3** of the ISPF Primary Option Menu to interact with the Utility Selection Panel.

```
Menu  Help
--------------------------------------------------------------------------------
                 Utility Selection Panel

 1  Library          Compress or print data set.  Print index listing.  Print,
                     rename, delete, browse, edit or view members
 2  Data Set         Allocate, rename, delete, catalog, uncatalog, or display
                     information of an entire data set
 3  Move/Copy        Move, copy, or promote members or data sets
 4  Dslist           Print or display (to process) list of data set names.
                     Print or display VTOC information
 5  Reset            Reset statistics for members of ISPF library
 6  Hardcopy         Initiate hardcopy output
 7  Download         Download ISPF C/S, VA for ISPF, transfer map, or data set.
 8  Outlist          Display, delete, or print held job output
 9  Commands         Create/change an application command table
 *  Reserved         This option reserved for future expansion.
 11 Format           Format definition for formatted data Edit/Browse
 12 SuperC           Compare data sets                       (Standard Dialog)
 13 SuperCE          Compare data sets Extended              (Extended Dialog)
 14 Search-For       Search data sets for strings of data    (Standard Dialog)
 15 Search-ForE      Search data sets for strings of data Extended (Extended Dialog)
 Option ===> 2
  F1=Help     F3=Exit     F10=Actions  F12=Cancel
--------------------------------------------------------------------------------
```

Choose the option 2 to allocate a data set.