GC33-5370-2
File No. S370-20

Systems

Introduction to DOS/VS

Release 29

IBM

GC33-5370-2
File No. S370-20

Systems

Introduction to DOS/VS

Release 29

IBM

Third Edition (November, 1973)

This is a major revision of, and obsoletes, GC33-5370-1. It includes changes reflecting support for the System/370 Model 115, new devices (3203, 5203, 3340, 3540, 3780, and 3420 Models 4, 6, and 8), and other DOS/VS system enhancements. Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to Version 5, Release 29, of the IBM Disk Operating System/Virtual Storage, DOS/VS, and to all subsequent versions and releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the *IBM System/360 and System/370 Bibliography,* GA22-6822, for the editions that are applicable and current.

Note: *For the availability dates of features and programming support described in this manual, please contact your IBM representative or the IBM branch office serving your locality.*

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Laboratory, Publications Department, P.O.Box 24, Uithoorn, The Netherlands. Comments become the property of IBM.

This Manual.....

...is a general summary of the IBM Disk Operating System/Virtual Storage (DOS/VS). Its purpose is to provide new users of DOS with a basic introduction to the system. For users familiar with DOS, it also gives a summary of the features and functions new in DOS/VS.

There are six major parts:

Part 1: **What is the Disk Operating System?** briefly describes the major characteristics of DOS/VS.

Part 2: **The Functions and Facilities of DOS/VS** develops a description of the system's functions and facilities, highlighting their use in an actual DOS/VS implementation wherever appropriate.

Part 3: **What's New in DOS/VS?** summarizes the new features of DOS/VS. It is intended primarily for users already familiar with the system, who may wish to turn directly to this section.

Part 4: **DOS/VS Component Programs** presents an overview of the DOS/VS system control programs (SCPs) and brief descriptions of some of the program products (PPs) that can be used with DOS/VS.

Part 5: **Configurations** is a list of System/370 CPU models and input/output devices that DOS/VS supports, as well as a chart showing the minimum system configuration.

Part 6: **DOS/VS Documentation** is a brief survey of DOS/VS manuals.

The reader is expected to have a basic knowledge of data processing. Supplementary information about System/370 functions and instructions may be found in *IBM System/370 Principles of Operation,* GA22-7000, together with *IBM System/360 Principles of Operation,* GA22-6821.

Table of Contents

Part 1. What is the Disk Operating System? 7
The Component Programs of the System 9

Part 2. The Functions and Facilities of DOS/VS 11
System Control 13
Controlling Jobs 13
Automatic Job-to-Job Transition 14
Assigning Actual I/O Devices to Symbolic Names 14
Loading Programs for Execution 18
Handling Program Termination 22
Resource Utilization 23
Storage Organization 23
Single-Partition System 23
Multiprogramming System 24
Multitasking 28
Virtual Storage Support 29
Virtual Storage and Address Areas 29
Allocating Storage to Partitions 30
The Concept of Paging 34
Performance Considerations 39
An Example of Partition Allocation 41
POWER 43
Remote Job Entry 44
Performance with POWER 45
Practical Considerations for Using POWER 45
Job Accounting 46
Libraries 49
Using the Libraries 49
Linkage Editor and Relocating Loader 50
Librarian Program 51
Data Management 55
Data Organization and Access Methods 55
Sequential Access Method (SAM) and Organization 56
Indexed Sequential Access Method (ISAM) and Organization 58
Virtual Storage Access Method (VSAM) and Organization 60
Direct Access Method (DAM) and Organization 62
Summary of Retrieval Methods for Disk 64
Teleprocessing Access Methods 64
Logical and Physical IOCS 66
High-Level Language Support for Data Management Functions 67
Data Security and Data Integrity 67
File Labeling 68
Protection Against Duplicate Assignments 69
DASD File Protection 69
TrackHold Function 70
Data File Security 70
Resource Protection Macros 70
I/O Devices Supported 70
System-Operator Interaction 71
Reliability, Availability, and Serviceability 73
Integrated Emulators 75
System Generation 77
Planning System Generation 77
Shipment of DOS/VS 78

Part 3. What's New in DOS/VS? 81
Processing Flexibility 83
Virtual Storage Support 83
Implementation 85
Relocating Loader 86
Implementation 86

Additional Foreground Partitions 86
Implementation 87
Variable Partition Priority 87
Implementation 87
The Shared Virtual Area (SVA) 87
Implementation 87
Extended I/O Device Assignment 88
Implementation 88
Supervisor Selection 88
Implementation 89
The Procedure Library 89
Implementation 89
Interval Timer Support for Multiple Tasks 90
Implementation 90
The DOS/VS Assembler 90
Implementation 91
Emulation on Models 115 and 125 91
Implementation 91
I/O Flexibility 93
VSAM - A New Access Method 93
Implementation 94
The POWER Program 95
Remote Job Entry 95
Implementation 95
New Devices Supported 96
Compatibility 97

Part 4. DOS/VS Component Programs 99
System Control Programming 101
Program Products 103
RPG II Compiler 103
FORTRAN Compiler 103
COBOL Compilers 104
DOS/VS COBOL 104
Full ANS COBOL Compiler, Version 3 104
Subset ANS COBOL Compiler, Version 1 104
PL/I Compiler 105
PL/I Optimizing Compiler 105
PL/I Libraries 106
Interactive Terminal Facility 106
DOS/VS Sort/Merge 106

Part 5. Configurations 107

Part 6. DOS/VS Documentation 117
Education 117
The DOS/VS SCP Library 117
Topical Groups in the SCP Library 117
Types of Manuals in the SCP Library 118
Manuals in the Program Product Library 118
Program Logic Information 118

Glossary 123

Index 133

Part 1. What is the Disk Operating System?

DOS/VS (Disk Operating System/Virtual Storage) is a comprehensive collection of programs designed to make full use of the resources of a data processing system. DOS/VS and the hardware system it controls combine to form a complete, effective computing facility.

DOS/VS operates on any size central processing unit (CPU) of System/370 Models 115, 125, 135, 145, 155-II, and 158.

Through the system generation procedure, DOS/VS can be tailored to complement the hardware system and meet the specific needs of the individual installation.

The Component Programs of the System

The component programs that make up DOS/VS may be divided into:

- Control programs
- Processing programs
- Data management routines

These programs and routines combine many data processing functions into a programming package that is designed to make maximum use of a hardware system and to relieve programmers and operators of a great deal of manual work.

For execution, the components of DOS/VS are stored online (that is, immediately and directly accessible whenever required) in areas on magnetic disk, called libraries. This allows fast loading of any program or routine into storage whenever its function is needed.

control programs

As their name implies, the control programs control the execution of all processing programs, IBM-supplied as well as user-written. DOS/VS control programs comprise the initial program loader, the supervisor, and the job control program.

The **initial program loader** is used to start operation with the system. It loads the supervisor into storage.

The **supervisor** controls overall system operation and provides general functions required by the job control program and all processing programs. It resides in the lowest area of storage, called the supervisor area, throughout system operation.

The **job control program** is loaded by the supervisor to initiate the execution of each new program and to establish which system facilities are to be invoked while that program is running.

processing programs

Processing programs are classified as all programs whose execution is initiated by the job control program and controlled by the supervisor. Processing programs can be divided into the three categories: language translators, service programs, and application programs.

Language translators translate source programs written in the various programming languages supported by DOS/VS into machine (or object) language.

Service programs assist with the use of the computing system and in the successful execution of problem programs, without contributing directly to the control of the system or the production of results. Among the most important service programs are the linkage editor, which converts the output of language translators into executable object programs; the librarian, which performs service and maintenance functions for the libraries on disk; POWER, which provides spooling support for unit record input/output; and emulators, which allow the execution on System/370 of programs written for certain other computing systems.

Application programs include user-written and, in some cases, IBM-supplied commercial and scientific programs.

data management

A third important class of components of DOS/VS are its data management routines. These are available for inclusion in problem programs to relieve the programmer of the detailed programming associated with the transfer of data between auxiliary storage and programs.

Figure 1.1 summarizes the concepts of DOS/VS described in this chapter. Part 2 contains a more comprehensive survey of the functions and facilities of the system.

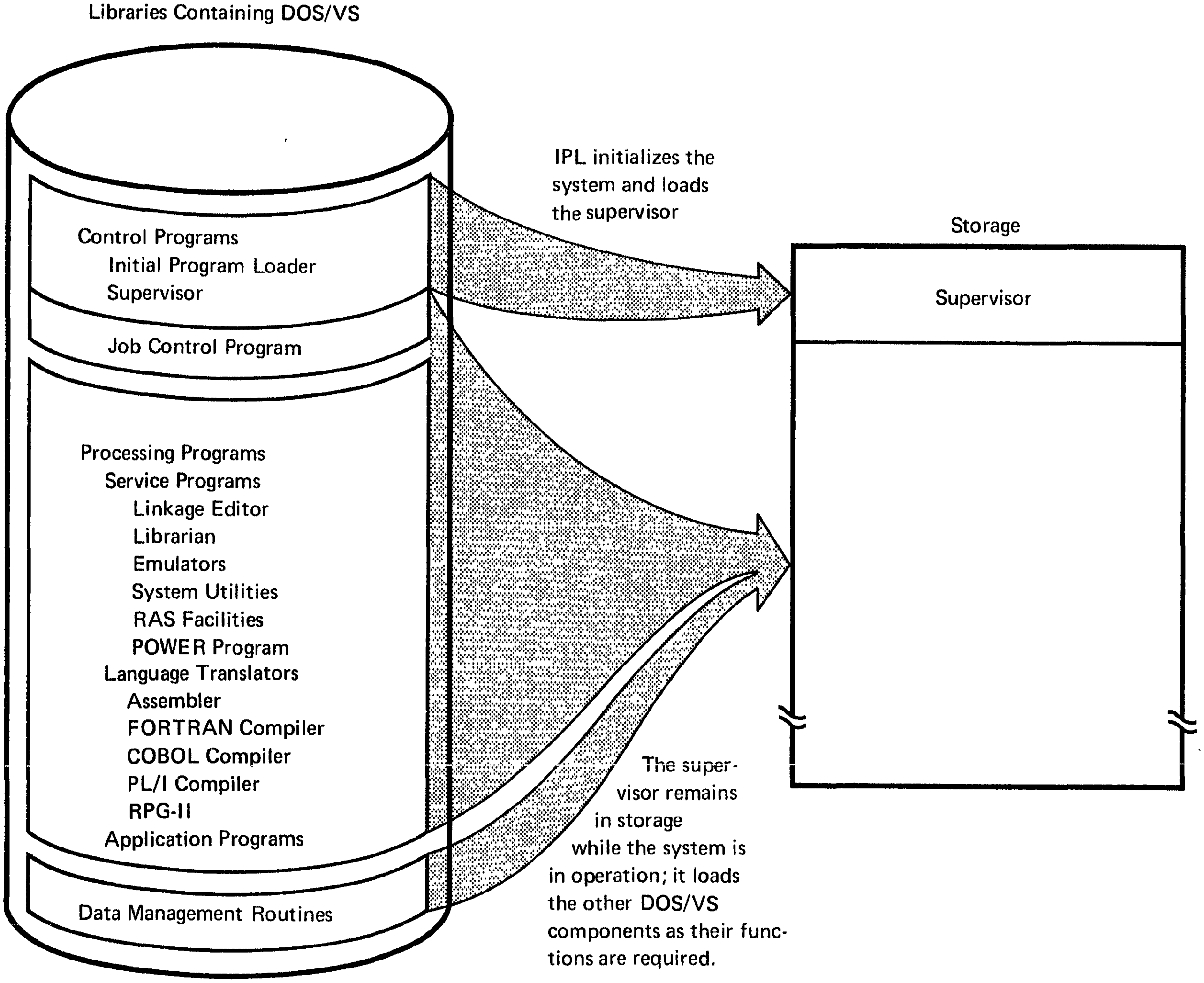


Figure 1.1. DOS/VS Component Programs

The components of DOS/VS are stored online in libraries on magnetic disk to permit fast loading into storage as needed.

Part 2. The Functions And Facilities of DOS/VS

Part 2 is a general survey of the system's most significant functions and facilities. These can be described as follows:

system control

DOS/VS controls the work (input, processing, output) to be performed by the computing system. It supervises the use of system resources and, based on control information from the user, their allocation to the jobs run on the computer system.

libraries

The programs and routines that make up DOS/VS are stored in libraries on disk storage. These libraries may also contain user-written programs and control information. There are four types of libraries - source statement, relocatable, core image, and procedure - which correspond to the basic formats in which program modules and control information may be maintained.

data management

The transfer of data between auxiliary storage devices and programs, as well as the organization of such data, is usually controlled by the DOS/VS data management routines. The services of data management are invoked by all system and user-written programs whenever they require the execution of input or output operations.

system-operator communication

Devices, alone, do not perform any data processing job. To get jobs done, the operator must initiate and monitor system execution and interpret and respond to messages issued by the system or the program.

reliability
availability
serviceability

To ensure a high degree of trouble-free operation of the complicated computing system, DOS/VS provides a number of routines for the detection, analysis, and recovery of machine and system malfunctions.

integrated emulators

For programs that were written to run on 1401/1440/1460, 1410/7010, or System/360 Model 20 computers, combinations of machine features and system programming are provided to allow these programs to run under DOS/VS.

system generation

The general version of DOS/VS distributed by IBM must usually be tailored to the needs of the user's specific machine configuration and operating requirements.

System Control

The functions of system control fall into two main categories:

- Functions that control the processing of jobs
- Functions that control the allocation and the use of system resources.

Functions that control the processing of jobs include automatic job-to-job transition, symbolic device addressing, loading of programs from the libraries for processing, and the handling of program termination.

Functions that control the allocation and the use of system resources include multiprogramming with concurrent execution of a number of programs, virtual storage support, input/output queueing to achieve efficient use of local and remote I/O devices and the CPU, and job accounting.

Controlling Jobs

After the system has been successfully started up by the initial program loader (IPL), it is ready to accept input for processing.

job and job stream

The unit of work the user submits to the system for processing is called a job. A succession of jobs presented to a computer is a job stream. Within each job, one or more programs may be executed. These programs may be IBM programs, such as a compiler that translates a user's source program into object code, or a user program that is already in executable format and processes data files.

job control statements

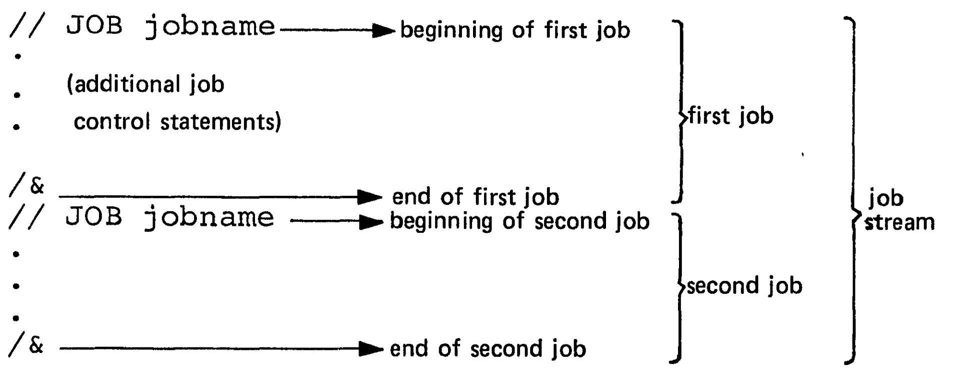
A job and the environment in which it is to run must be defined to the system by means of job control statements. Job control statements specify, for example, whether user programs are to be compiled, link-edited, and/or executed, from which library or device the system or user programs are to be loaded, what files they process and where these files reside.

When handling jobs, DOS/VS performs the following four basic functions:

- It provides for automatic transition from job to job, with a minimum of operator intervention.
- On the basis of job control statements supplied by the user, it assigns actual I/O devices to the symbolic device names specified in programs.
- It loads executable programs from online disk libraries into storage for processing.
- It handles program termination.

Automatic Job-to-Job Transition

Jobs are defined to the system by means of job control statements. The beginning of the job is always indicated by a JOB statement. Job control statements are, in most cases, identified by two slashes (//) as the first two characters. The exceptions are / & (indicating end of job), /* (indicating end of data), and * (indicating comments).



job step

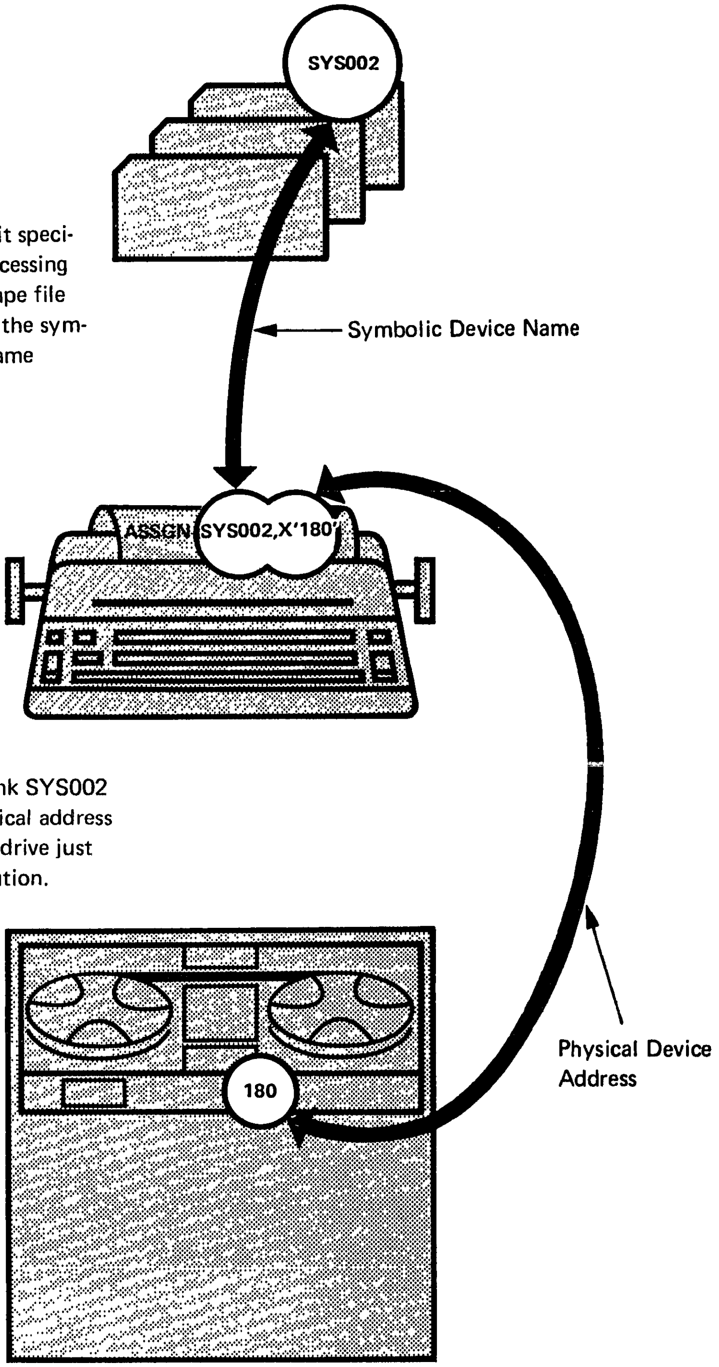
The user may define jobs as multiple or single job steps. Each job step consists of one program, which executes after the preceding job step is completed. Defining a series of related programs as a single job may sometimes be required or it may provide specific advantages. For instance, multiple job steps within a single job would be preferred if execution of a later job step were directly dependent on successful completion of an earlier one. If a job step terminates abnormally, the remaining steps are bypassed, and the next job is initiated.

Whenever a job or job step has been completed, the job control program is automatically reloaded by the supervisor. Job control reads and processes job control statements from the device assigned to the system input stream until it finds an EXEC statement, which requests execution of a program. It then passes control to the supervisor, which locates the requested program in the core image library. The core image library contains all executable programs, both IBM-supplied and user-written. Finally, the requested program is loaded for execution and gains control to start processing. In this way, the transition from one job to the next one in the job stream is handled by the system without intervention by the operator.

Assigning Actual I/O Devices to Symbolic Names

Processing programs that need access to a data file must usually inform the system of the type of device involved. The actual device need not be specified in the program, only a symbolic name referring to a logical, rather than physical, unit must be specified.

The assignment of a logical device name to a physical device (channel and unit number) is made by the user either during system generation or during system operation.



Processing Program

The logical unit specified in the processing program is a tape file referred to by the symbolic device name SYS002

Job Control

The ASSGN command may be used by the operator to link SYS002 with the physical address 180 of a tape drive just prior to execution.

I/O Device

Figure 2.1. Assigning an Actual I/O Device to a Symbolic Device Name by the Operator

standard assignment

Standard assignments, called default values, are made during system generation and are effective unless overridden by either permanent or temporary assignments.

permanent and temporary assignment

At any time during system operation, the user can specify a permanent or a temporary assignment to the system. A permanent device assignment overrides the default value for the duration of system operation and remains in effect for any jobs unless overridden by a temporary assignment. A temporary device assignment holds for one job or until it is overridden by another assignment.

The user can specify permanent or temporary assignments in two ways. The programmer can include them in job control cards in the job stream or the operator can enter them directly from the console keyboard that he uses to communicate with the system.

Permanent and temporary device assignments offer the user the following advantages:

- If the configuration of the computer is changed by the addition or removal of a particular device, the programs need not be altered as long as another unit of the same device type is available.

- If a device-type, device-class, or address list is used in an assignment, the program is to a large extent independent of the configuration of the system on which it is run. Defective, inoperative, or assigned devices are of no consequence in that case.

- If, before the execution of a program, a device appears to be defective, inoperative, or in use by another program, the operator can select another unit of the same device type, enter a new assignment to reflect the new physical address, and run the job or job step. (See Figure 2.1.)

A full list of the symbolic device names (logical units) recognized by DOS/VS is given in Figure 2.2. The use of the logical units will become clear later in the book.

Logical Unit	Device Type	Used for
System Logical Units:		
SYSRDR	card reader, magnetic tape unit, disk extent, or diskette extent	reading job control statements
SYSIPT	card reader, magnetic tape unit, disk extent, or diskette extent	input of system data, such as source statements for language translators, or control information for service programs
SYSIN	card reader, magnetic tape unit, disk extent, or diskette extent	**Can** be used when SYSRDR and SYSIPT are assigned to the same card reader or magnetic tape unit; **must** be used when SYSRDR and SYSIPT are assigned to the same disk extent or diskette extent.
SYSPCH	card punch, magnetic tape unit, disk extent, or diskette extent	punched output of the system
SYSLST	printer, magnetic tape unit, disk extent, or diskette extent	printed output of the system
SYSOUT	magnetic tape unit	**Must** be used when SYSPCH and SYSLST are assigned to the same magnetic tape unit. It **cannot** be used to assign SYSPCH and SYSLST to disk since these two units must refer to separate disk extents.
SYSLOG	console printer keyboard, display operator console, or printer	operator messages and for logging job control statements. It can also be assigned to a printer if used in a single-partition environment.
SYSLNK	disk extent	input to the linkage editor
SYSRES	disk extent	system residence device
SYSVIS	disk extent	for virtual storage support
SYSCAT	disk extent	VSAM catalog
SYSCLB	disk extent	private core image library
SYSRLB	disk extent	private relocatable library
SYSSLB	disk extent	private source statement library
SYSREC	disk extent	logging error records
Programmer Logical Units:		
SYS000 SYS001	any device in the system	user program input/output
SYSnnn		

Figure 2.2. Symbolic Device Names Recognized by DOS/VS

The system logical units are primarily used by the system. Units SYS000 through SYSnnn are primarily used by problem programs and are called programmer logical units. The maximum value of SYSnnn varies with the number of partitions in a system. In addition to the programmer logical units, problem programs may also use the system logical units SYSRDR, SYSIPT, SYSPCH, SYSLST, and SYSLOG.

Loading Programs for Execution

All executable object programs, both IBM-supplied (such as supervisor, linkage editor, language translators, utilities) and user programs, must be stored in a core image library. Programs that are used frequently are stored permanently; those not often used may be stored only temporarily, just prior to loading for execution. Whether a program is to be stored permanently or temporarily can be specified by a parameter in a job control statement.

Resident programs that can be shared between partitions and that are used frequently may be stored in the shared virtual area (SVA). (These programs must be relocatable and reenterable.) Whether a program can be stored in the SVA is specified by the SVA parameter of the PHASE statement. The control cards following the SET SDL=CREATE statement are used to specify if a program is to be stored in the SVA; this is done when the system directory list (SDL) is built immediately after IPL.

The storing (or cataloging) is done by the linkage editor, which processes the output from language translators and places its output, one or more executable program phases, into a core image library.

A program is loaded from the core image library for execution by the supervisor. The supervisor itself may make the request, for example, in the case of error recovery procedures; the request may come from the job control program after it has read the control statement identifying the next core image library phase to be loaded; or the request may come from any other program.

The following sections illustrate two specific sequences of job control statements. The numbers on the left in Figures 2.3 and 2.4 indicate the sequence of events and correspond to the same numbers in the illustrations, which give a clearer picture of the flow.

The first example (see Figure 2.3) is the execution of a program that has already been cataloged in the core image library.

The second example (see Figure 2.4) is more complex. It illustrates how a source program would be compiled and executed in a single job. The job consists of three job steps: one for the assembly, the second for the link-editing of the object module, and the third for the execution of the core image phase. This sort of job control sequence is typical of the testing phase of program development. Test data is submitted during execution, but the program is only placed in the core image library temporarily. After all debugging has been completed and after successful runs with test data, the program would probably be cataloged permanently into the core image library.

Sequence Indicator	Job Control Statement	Explanation
1		The job control program is loaded at the completion of the previous job or at IPL. Job control then reads statements from the device assigned to SYSRDR.
2	// JOB jobname	The // JOB card is the first of a set of control cards for a job. Its function is to indicate the start of a job, to provide job accounting information, and to give the job a name. The programmer or operator may choose any name he wishes, within the rules imposed by DOS/VS.
3	// EXEC PROG1	When job control reads this card, it causes the supervisor ④ to locate the program with the specified name,PROG1, in the core image library, load this program into storage ⑤, and execute it.
		Additional job control statements may be read and processed between the // JOB and // EXEC statements. If not, the system assumes 'default' values for the possible variables, according to specifications made during system generation or IPL.
6		PROG1, during its execution, reads data on cards from the programmer logical unit SYS004. This may be the same physical device as SYSRDR.
7	/*	This card indicates the end of data. When PROG1 terminates its execution, it returns control to the supervisor ⑧ which again loads job control from SYSRES, ⑨
10	/ &	Job control reads the next statement from SYSRDR. The characters / & in the first two positions indicate the end of the job. Job control then terminates the job and proceeds to the next job in the job stream.

Figure 2.3. Loading and Executing from the Core Image Library (Part 1)

The job control statements cause programs permanently cataloged in the core image library to be loaded into storage and executed.

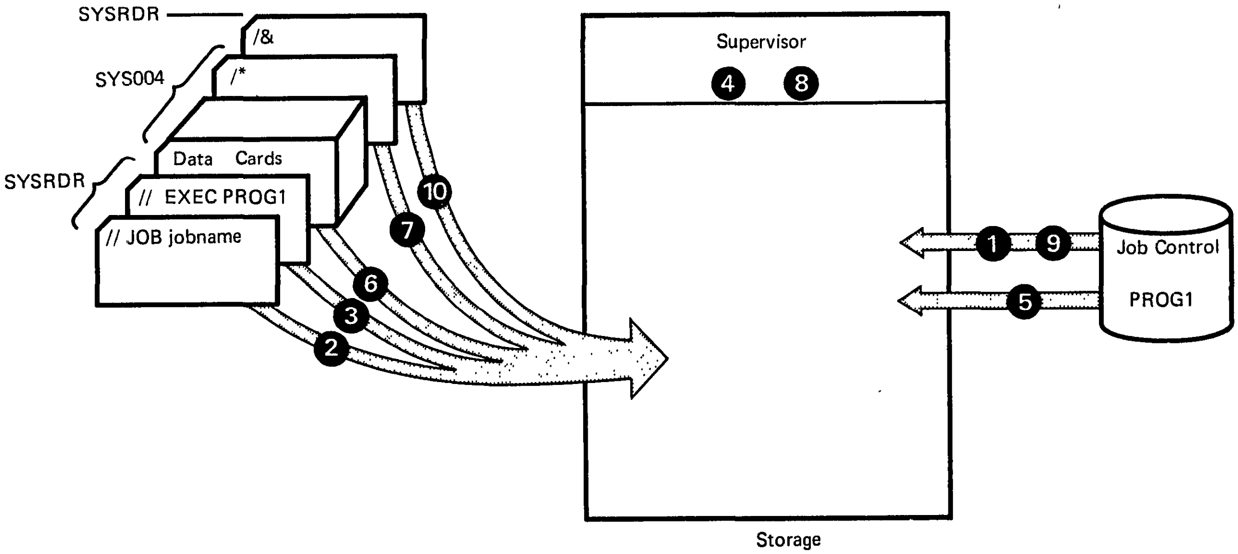


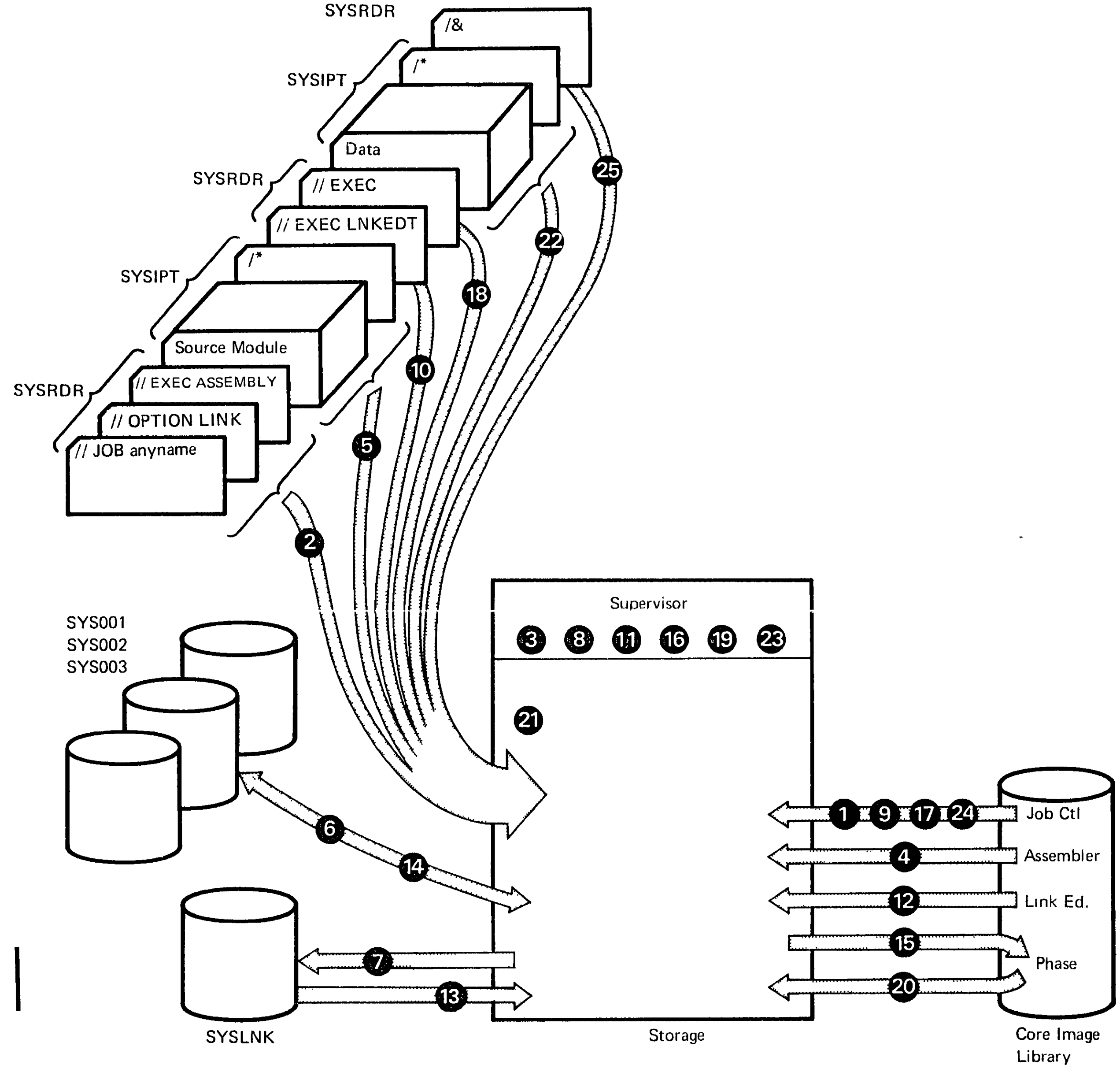
Figure 2.3. Loading and Executing from the Core Image Library (Part 2)

The job control statements cause programs permanently cataloged in the core image library to be loaded into storage and executed.

Sequence Indicator	Job Control Statement	Explanation
1		After the IPL procedure, or on completion of the previous job, the job control program is loaded from the core image library on the SYSRES device.
2	// JOB anyname	Job control reads and processes the control cards, starting with the JOB card, from SYSRDR until an EXEC card is encountered.
	// OPTION LINK	OPTION LINK signals that the assembler output is to be link-edited and cataloged temporarily (**not** permanently) into the core image library.
	// EXEC ASSEMBLY	The program name on the EXEC card is then passed to the supervisor ③ which receives control and ④ loads the desired program (in this case, the assembler).
5		The assembler reads and processes the source program input cards from SYSIPT, up to the /* card, which indicates the end of the source statements.
6		When processing the input cards, the assembler uses three work files for storing intermediate results. These are SYS001, SYS002 and SYS003.
7		The object module produced by the assembler is written on SYSLNK for subsequent processing by the linkage editor. This action is triggered by the OPTION LINK card.
8		When the assembly is complete, the supervisor receives control and loads the job control program again.
10	// EXEC LNKEDT	Job control reads the next card from SYSRDR.
11		It passes the name of the desired program (LNKEDT) to the supervisor, which loads the linkage editor into storage ⑫.
13		The linkage editor retrieves the object module from SYSLNK, uses SYS001 as a work file ⑭, and places its output, an executable program phase, temporarily into the core image library ⑮.
16		The supervisor loads the job control program again ⑰.
18	// EXEC	Job control reads the next card from SYSRDR. The EXEC card with a blank operand indicates that the program phase just link-edited should be executed.
19		The supervisor receives control and ⑳ loads the program phase from the core image library.
21		The assembled program begins processing and, in this example, reads data from SYSIPT ㉒ up to the /* card.
23		At program termination, the supervisor loads job control ㉔
25		Job control reads the /& statement, indicating end-of-job.

Figure 2.4. Assembling, Link-Editing, and Executing a Source Program (Part 1)

The program is temporarily cataloged into the core image library and executed immediately.



Note: Disk files shown in this example may be on one or more physical units.

Figure 2.4. Assembling, Link-Editing, and Executing a Source Program (Part 2)

The assembler and the linkage editor are processing programs. They operate just as other problem programs using the supervisor and standard data management routines.

Handling Program Termination

The job control program handles normal program termination to ensure job-to-job transition. It also handles any unusual situations that would require abnormal program termination. There are three reasons for abnormal program termination:

abnormal program termination

- Operator request for program termination from the console keyboard. He would request cancellation, for example, if he suspected that the program had entered an unending program loop.
- Program failure or illegal program action. An example might be a program's attempt to address a non-existent or non-assigned I/O device.
- Unrecoverable hardware failure. If an exceptional hardware condition occurs, the system tries to recover. Frequently, recovery is successful, but when it is not, an abnormal program termination occurs. An example would be a persistent read or write error that could not be resolved by the system's error recovery procedures.

In case of abnormal program termination, the following series of events occurs:

- If the system's error recovery procedures were involved, hardware and environmental data is recorded on a system recorder file. (See the section on Reliability, Availability, and Serviceability (RAS) Aids.)
- The system issues a message to the operator, indicating the cause of the failure. (A malfunction of the console keyboard or display could prevent this.)
- The system may produce a storage dump (a hexadecimal printout). A dump can be specified or suppressed by the programmer at any point in the job stream. The operator can request a dump or a trace of a particular set of instructions. Collectively, these are referred to as problem determination aids. (See the section on RAS.)
- The remaining data and control cards for the job are bypassed in the input stream.
- The next job in the input stream is started. (It is possible that system operation is so extensively impaired that the IPL procedure has to be repeated.)

user exit routines

If the user is programming in the assembler language, he has also the option of including program check handling and user exit routines in his program. Then, if an abnormal program termination occurs, the supervisor passes control to the appropriate user routine, giving information on the cause of the abnormal termination. The user routine can examine this data and take appropriate action.

Resource Utilization

For the user, the main measure of a system's efficiency is the throughput, that is, the amount of work handled in a certain period of time. The major resources to be examined when discussing the system's throughput are (1) CPU processing time and its use, (2) virtual storage space and its exploitation by problem programs, (3) disk library space and its employment, and (4) I/O devices and their efficiency.

DOS/VS provides features that improve system throughput by allowing efficient utilization of system resources:

- Efficient use of CPU time through multiprogramming which allows concurrent execution of more than one program
- Efficient use of the problem-program area through multiprogramming and virtual storage support
- Efficient use of disk library space through DOS/VS system enhancements and the relocating loader feature
- Efficient use of CPU time and unit-record I/O devices through the POWER program.

Facilities also exist in DOS/VS for the user to monitor CPU usage and I/O activity through the job accounting facility. This may allow him to balance his mix of concurrently running jobs to achieve better total system utilization.

In order to examine these features in more detail, it is first necessary to look briefly at the storage organization under DOS/VS.

Storage Organization

DOS/VS allows the user much greater flexibility than previous versions of the system in organizing and utilizing the storage in which to process his programs. In order to design a storage organization that best fits the needs of a particular installation, he must choose among a number of basic alternatives when generating his system.

Single-Partition System

The single-partition system presents the simplest type of storage organization. The lower storage area contains the supervisor, which remains resident throughout system operation. The remaining area, or background partition, is where all other programs, both IBM-supplied and user-written, execute. (See Figure 2.5.)

A standard DOS/VS storage protection feature isolates the supervisor and all processing programs during system operation, so that one program cannot cause damage to another.

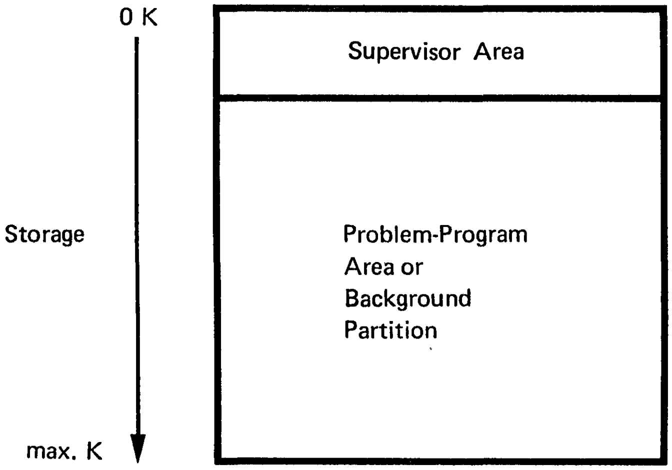


Figure 2.5. Single-Partition Storage Organization

In a single-partition environment, storage is divided into the supervisor area and the background where all problem programs execute. Only one program can occupy the background at a time.

CPU usage

In a single-partition system, only one problem program can be in storage at a time. If it needs input or output, it issues an I/O request to the supervisor. The supervisor passes this request to a channel program, which then executes the I/O operation. During most of this time interval, the CPU itself remains idle, or in the *wait state.* (See Figure 2.6.)

Multiprogramming System

DOS/VS allows the user to divide the problem-program area into as many as five areas, called partitions (see Figure 2.7).

Each partition can contain a separate program, which allows concurrent execution of multiple programs. This is called multiprogramming. Each program is logically independent, but it takes turns with the other programs in using the CPU facilities, thus reducing the time that the multiprogramming system is in the unproductive *wait state.*

The user specifies the number and size of partitions at system generation time. The number of partitions cannot be changed during system operation, but the partition sizes can be modified between jobs or job steps by means of an operator command. The amount of storage allocated to a partition can even be set to zero, which, in effect, reduces the number of partitions.

The number and size of partitions, which best meet the needs of an installation, depend upon such factors as the total amount of storage available, the size and structural characteristics of the processing programs, their balance among job streams, and the operating environment. The user may choose to vary the multiprogramming capability of his system at different intervals during the day or shift operation. He can even allow his multiprogramming system to function in single-partition mode by changing the size of all but one partition to zero, leaving all of the storage available to that one partition.

partition priorities

With programs taking turns executing in a multiprogramming environment, processing must obviously proceed according to a set of priorities. The priority of a program for receiving CPU resources is dependent upon the priority of the partition in which it resides. The supervisor, of course, always has the highest priority.

The default values for the partitions are, from low to high, in the order of their distance from the supervisor: background, foreground-4, foreground-3, foreground-2, and foreground-1. In DOS/VS, the user can change these default values during system generation or by means of operator commands. By assigning a job to a certain partition, a programmer or an operator therefore assigns the priority of that partition to the job as well.

CPU usage

Figure 2.8 illustrates the passing of control among programs in a multiprogramming environment. Note how much less CPU time is left unused or idle when compared to CPU usage in a single-partition system (Figure 2.6).

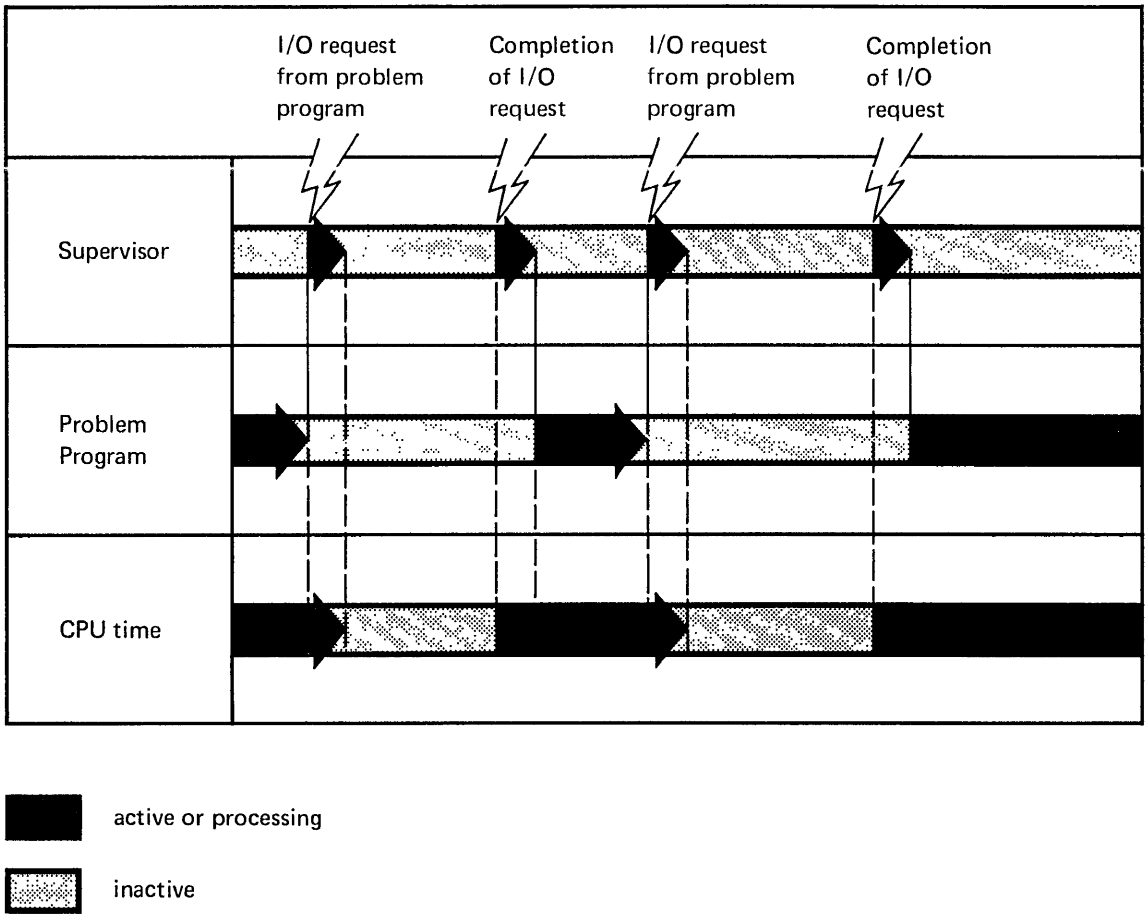

I/O request from problem program
Completion of I/O request
I/O request from problem program
Completion of I/O request
Supervisor
Problem Program
CPU time
active or processing
inactive

Figure 2.6. CPU Usage in a Single-Partition System

The CPU is in the wait state between the time an I/O request is issued and the operation is completed. A significant amount of CPU time is spent waiting in this way (gray areas in the diagram).

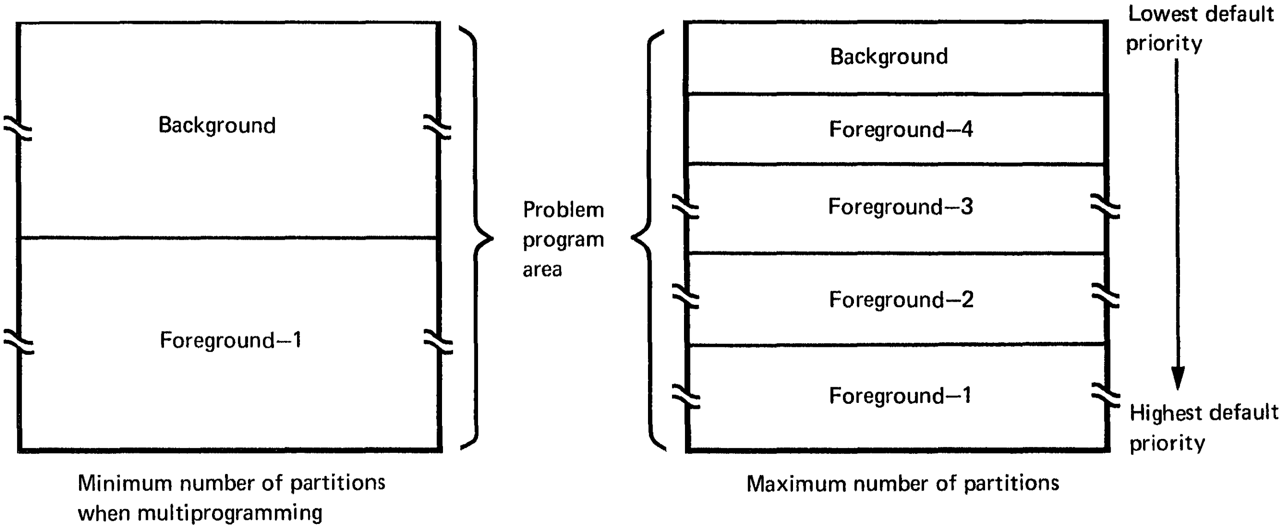
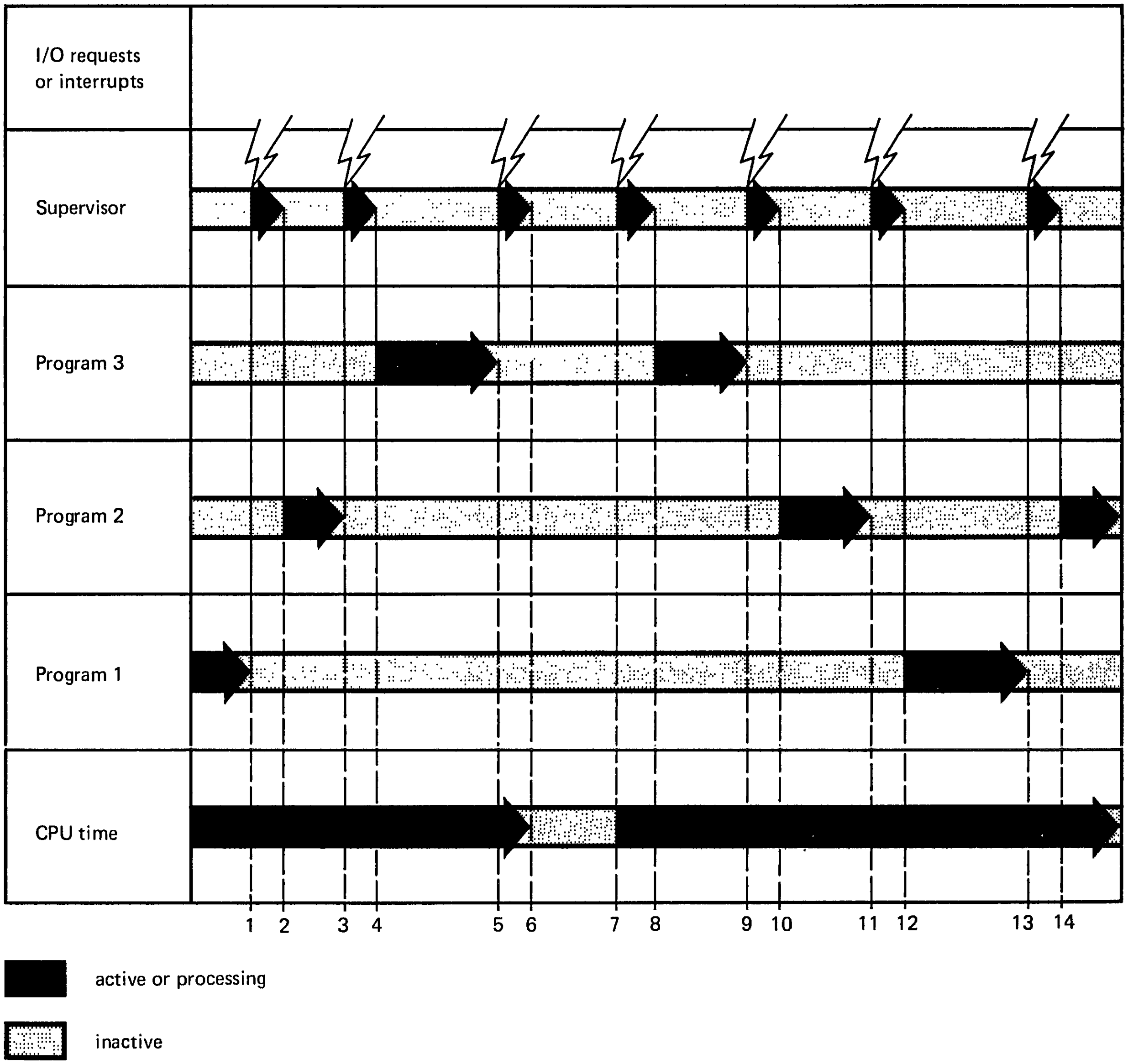


Figure 2.7. Default Priorities in a Multiprogramming Environment

The default priorities assigned by the system may be changed through an operator command.



Note: This example is based on a multiprogramming system with three active partitions. Program 1 has the highest priority, program 3 the lowest.

Figure 2.8. CPU Usage in a Multiprogramming System.

CPU time is more usefully employed when three programs call upon the CPU resources. The bottom line shows a noticeable reduction in the amount of time the CPU spends in an inactive state (gray areas).

Points 1, 3, 5, and 13. A program (numbers 1, 2, 3, and 1 respectively) issues an I/O request and enters the wait state pending its completion. The supervisor takes over to start the I/O operation, and to determine which other program can start processing.

Points 2, 4, and 14. Programs 2, 3, and 2, in that order, receive control, because they are waiting with no I/O requests pending.

Point 6. The supervisor is unable to find a program waiting with no I/O requests pending. The system enters the wait state, waiting for an I/O interrupt.

Points 7, 9, and 11. An I/O interrupt occurs, signaling the completion, in turn, of the I/O operation for programs, 3, 2, and 1. The supervisor determines which is the highest priority program ready to resume processing.

Point 8. Program 3 regains control, because programs 1 and 2 are still waiting.

Points 10 and 12. Programs 2 and 1 resume processing, because they are the highest priority programs waiting with no pending I/O requests.

In the example, note that the switch from one partition to another is always made upon an I/O request, or I/O interrupt.

Multitasking

Multitasking is a special form of multiprogramming which allows concurrent execution of two or more sections of a program, called "tasks", within a single partition. The purpose of this facility is again to make more efficient use of CPU time by providing a means of allowing various parts of one program to execute concurrently.

main and subtasks

With multitasking, one main program, the main task, "attaches" one or more subprograms, or subtasks. The main task gets control from the supervisor following an EXEC statement and then initiates, or attaches, the subtasks. The main task and its attached subtasks always reside in the same partition.

Usually, the main tasks and its subtask(s) are parts of one program. It is, however, also possible to attach completely different programs to a common main task for execution in the same partition.

The total number of tasks in a system depends on the number of partitions. The sum of all subtasks and all partitions must not exceed 15:

2-partition system:	13 subtasks maximum
3-partition system:	12 subtasks maximum
4-partition system:	11 subtasks maximum
5-partition system:	10 subtasks maximum

All of these subtasks can be attached to one main task, or they can be divided among any number of main tasks.

Subtasks have higher priority than the main tasks for processing within the partition. The pnonty of a subtask in the partition is determined by the sequence in which it is attached by the main task. The subtask attached first has the highest priority, and so on.

That part of the main task that attaches or detaches subtasks must be written in DOS/VS assembler language. The rest of the main task and the subtasks may be written in any high-level language, provided certain restrictions are observed.

Virtual Storage Support

Virtual storage support is the most significant new function in DOS/VS. DOS users in the past have been restricted to an address space defined by the storage physically contained in their computer system. There are a number of terms used to describe this storage. The most common are processor storage, main storage, CPU storage and internal storage. We will refer to the computer's internal, physical storage as **real storage.** Prior to the availability of DOS/VS, the total of supervisor and partition sizes could not exceed this constraint and programs were confined to real storage partition limits. The size of a partition was usually determined by the size of the largest program that was to run in it. This meant that whenever programs smaller than this maximum were executed, real storage was **fragmented,** frequently leaving some storage in each partition unused and unproductive. DOS/VS changes these concepts.

Virtual Storage and Address Areas

Through a combination of System/370 hardware design and programming system support, DOS/VS has an address space, called **virtual storage,** that starts at zero and can extend to the maximum allowed by the system's addressing scheme. A System/370 address consists of 24 bits, providing for up to 16,777,216 bytes of address space. How much of this address space will be used in a particular system depends upon a number of factors: the size of the computer's real storage, the number of partitions, their size, and the characteristics of the installation's programs and operating environment.

Based on these factors, trade-offs are made to arrive at an optimal virtual storage size for the requirements of the particular installation. A tailored system is then generated to this size, which will contain a virtual storage smaller than the maximum limit of 16,777,216 bytes, and normally larger than the real storage installed in the system.

real address area

Assume a virtual storage system of 544K with 160K bytes of real storage. (Later in this section, figures 2.15 and 2.16 will illustrate the example that produced this result.) That part of the installation's virtual storage which can be directly equated to real storage is called the real address area.

virtual address area

That part beyond the end of the real address area, up to the limit of the system's generated virtual storage, is the installation's virtual address area. The relationship can be represented as shown in Figure 2.9.

Through the availability of the virtual address area, the constraint imposed by real storage on program size is no longer absolute. The virtual address area, as well as the real address area, is available for DOS/VS partitions. Since the maximum size of virtual storage is very large, such partitions and the programs in them can also, theoretically, be of similar magnitude. In practice there are limitations on these sizes. The user must consider such factors as the amount of real storage available, the size and structural characteristics of the programs in the virtual address area, and make trade-offs between program size limitations and the efficiency of program execution. The impact of these factors will be evident as the description of virtual storage proceeds.

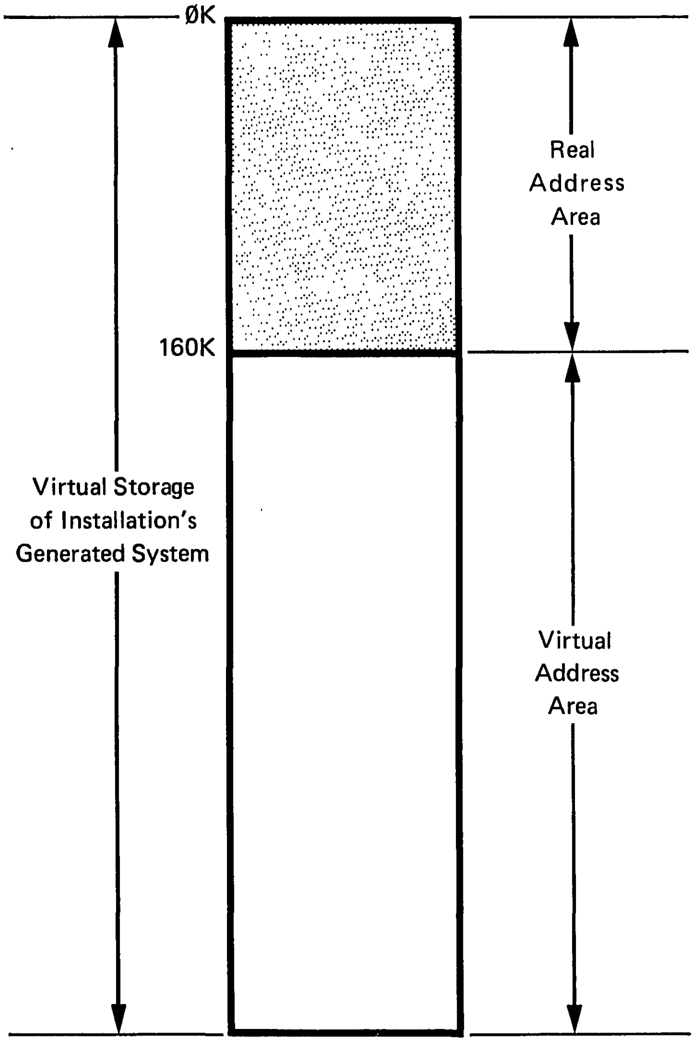


Figure 2.9. Virtual Storage

Virtual Storage in a DOS/VS System is made up of the real address area and the virtual address area.

Allocating Storage to Partitions

During system generation the user specifies the number of partitions his system will support. In DOS/VS this may be from one to five, as discussed previously. Next, storage must be allocated to partitions. As in earlier releases of DOS, storage is allocated to partitions at system generation time and the amount of storage can be subsequently modified by the operator through a job control command. The difference in DOS/VS is that storage in both the real and the virtual address areas is allocated to the partition.

real address area allocation

The computer installation's real storage is used for **four** functions:

1. **Supervisor Residence.** The supervisor resides in the real address area.

2. **Partition Allocation.** Portions of the real address area may be allocated to any or none of the partitions defined during system generation. If this allocation is made, programs may be loaded from a core image library into the real address area allocated to the partition and be executed there. This allocated portion of the real address area is called a **real partition** and the program loaded there runs in **real mode.** Programs running in real mode cannot exceed the limit of their real partition. They are characterized by a high level of performance efficiency.

3. **Execution of Programs from the Virtual Address Area.** As an alternative to running a program in real mode, a programmer may think of his program as executing in a partition in the virtual address area. However, instructions must be physically resident in real storage to be executed, and DOS/VS assumes the responsibility for placing the code from the virtual address area into real storage for execution. The mechanism that does this is explained in more detail later; it is important at this point to understand that an area of real storage must be available to receive the code from the virtual address area.

4. **Execution of Programs Resident in the Shared Virtual Area (SVA).** The SVA contains reenterable, relocatable user phases that can be shared between partitions. The code of these phases is in executable format. Because the SVA is in the high portion of virtual storage, a phase that is to be executed need not be loaded again. If the user wants to execute a program, the system directory list (SDL), which contains pointers to all phases in the SVA and pointers to frequently-used phases in the CIL, may be searched to see if the required phase is in the SVA. If so, the program is executed immediately.

Some programs must execute in real mode. The supervisor always runs in real mode because the control of the entire virtual storage mechanism is contained within the supervisor. For any of these functions to occur, then, the routines must be present in real storage. A category of user programs which should execute in real mode are those which have a high level of time dependence. The QTAM message control program, for example, must occupy a real partition. The POWER program, too, has to run in real mode.

virtual address area allocation

Storage in the virtual address area must be allocated for all active partitions whether the user programs that run in them will execute in real mode or in **virtual mode.** Virtual mode means, conceptually, that when the program is loaded from a core image library for execution, it is loaded into the virtual address area allocated to the partition. Virtual mode also means that programs or phases that are in the SVA are executed immediately from that area. The virtual address area allocated to a partition is called the **virtual partition.** For actual program execution, DOS/VS then places the program, or sections of it as required, into real storage.

There are several factors that would cause a user to plan to run certain programs in virtual mode. One is program size. Virtual partitions can be allocated to contain programs that are too large to reside in the real partitions. Frequently it is more economical to have a program execute in virtual mode than to require the programmer to develop an overlay structure to force the program to fit into a real partition. In cases where a program contains significant amounts of code that are only infrequently referenced, execution efficiency need not be substantially impacted. However, the user must be aware that execution efficiency can decrease if the sum of the sizes of programs running in virtual mode is considerably greater than the size of real storage available for those programs. Some of the pertinent factors are discussed under the heading "Performance Considerations" in this section.

Here is another factor favoring virtual mode. DOS/VS assumes the responsibility for managing that portion of real storage where programs from virtual partitions are placed for execution. DOS/VS storage management involves the dynamic assignment of real storage among all the programs running concurrently in virtual mode. This can substantially reduce or eliminate the storage fragmentation which occurred with earlier versions of DOS when programs were smaller than the partition in which they ran. Assignment of real storage is done according to partition priority and storage requirements of each program at different stages of its execution. The system can also temporarily suppress one or more programs when there is not enough real storage to support all programs running in virtual mode at a reasonable level of performance. The more real storage available for this storage management activity and the higher the percentage of the installation's programs running in virtual mode, the greater the utilization of DOS/VS storage management to exploit the valuable system resource of real storage. This benefit to an installation may prove to be the greatest advantage of DOS/VS over prior DOS releases.

Job streams are usually built for execution in a specific partition. Each program of the job stream executes either in real mode or in virtual mode; each partition may have parts of both the real and virtual address areas allocated to it. Each partition **must** have part of the virtual address area allocated to it in order to contain certain of the IBM system programs, such as job control, which run in virtual mode. The minimum virtual partition allocation allowed by the system is 64K. Therefore each partition must have at least 64K of virtual address area. It may of course have more.

Consider the example of a DOS/VS system, as shown in Figure 2.10, to which the following considerations apply:

- Number of partitions: Four.
- Allocate real storage to F3 and F2.
- Remember that portions of the virtual address area must be allocated to each partition.

(The terms F3-R and F3-V, meaning the *Foreground-3 Real partition* and the *Foreground-3 Virtual partition,* describe the *real address area allocated to the foreground-3 partition* and the *virtual address area allocated to the foreground-3 partition.* Comparable designations and descriptions apply to each of the other partitions.)

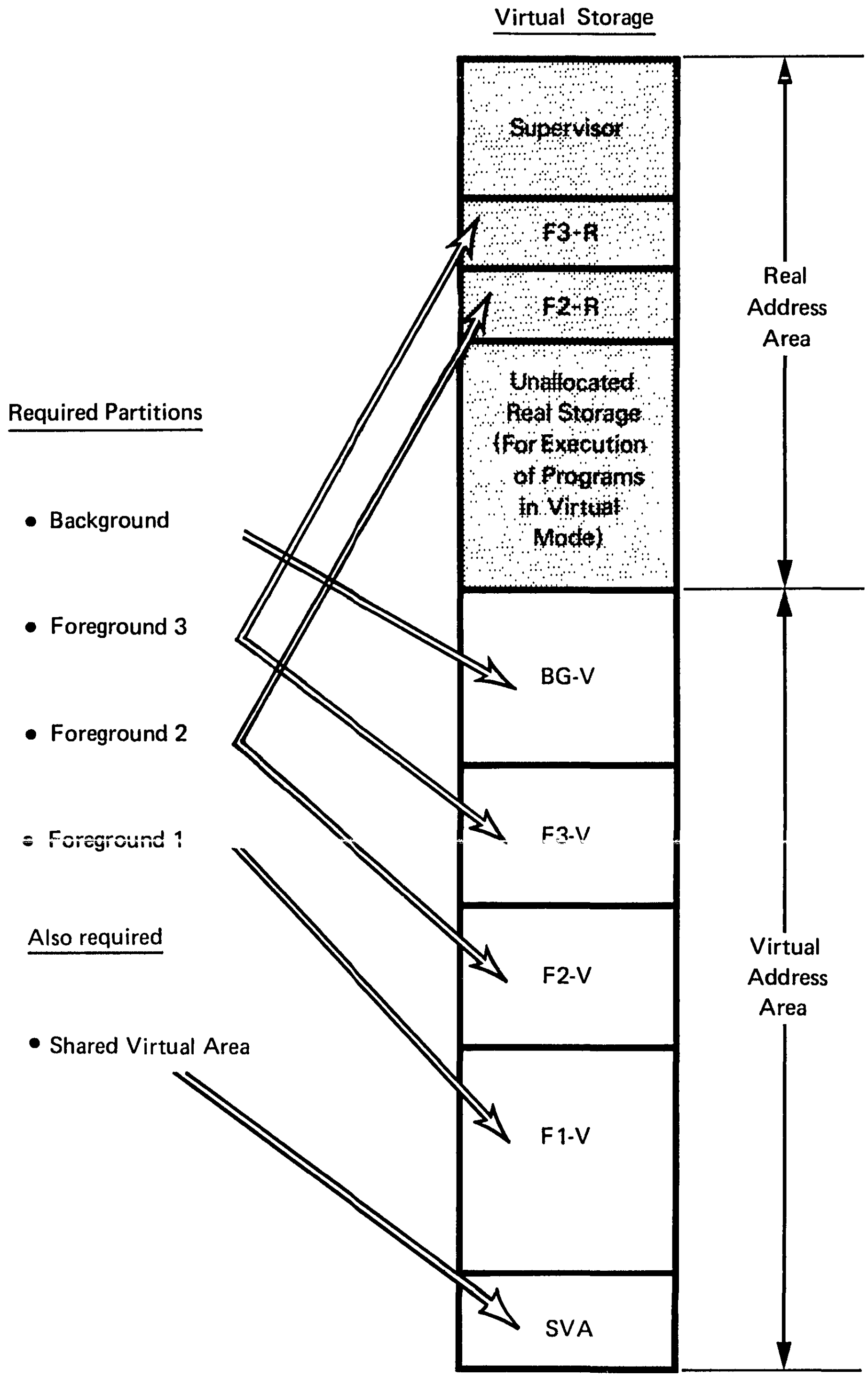


Figure 2.10. An Example of Storage Allocation

Each partition must have an associated virtual address area and may have an associated real address area.

page data set

DOS/VS must have a means of physically representing and containing the programs which at any instant are running in the virtual partitions. For this purpose, the user establishes an area of disk storage that is equivalent in capacity to the virtual address area allocated to the system. The disk area is called the page data set and it is used by DOS/VS to contain programs or parts of programs currently running in virtual mode for which there is no real storage available.

pages, page frames, and page pool

As already discussed, a part of real storage has to be kept available to contain programs running in virtual mode. When the limitations of this real storage prevent all programs running in virtual mode from being simultaneously present in real storage, DOS/VS exchanges sections of programs between the page data set and real storage, as they are required for execution. The program sections are called pages, each 2K bytes in length. The area of real storage into which the system loads a page is called a page frame. All the real storage page frames into which pages from any program running in virtual mode may be brought for execution make up the page pool. (Refer to Figure 2.11.) The page pool can dynamically change in size as the system runs. Real storage contributing to the size of the page pool at any moment is made up of:

- Real storage not occupied by the supervisor and not allocated to partitions. (This must be at least 16K bytes unless all programs are running in real mode. In this case it may not be required at all.)
- Real storage allocated to a partition when the program in that partition is running in virtual mode. In this case, the program may be thought of as occupying the **virtual** address area of the partition, leaving available to the page pool any **real** storage allocated to that partition.)
- Any real storage allocated to a partition in excess of that actually required by the program currently running in that partition in real mode. (For example, if a 40K byte program is running in real mode in the 54K real address area of a partition, the surplus 14K bytes can be made available to the page pool.)

Contributions to the page pool from these last two sources can obviously change as each subsequent program is initiated.

page fault

When a program is running in virtual mode, all code required for execution may not be in real storage. When a program tries to refer to a storage address within a page which is not in real storage, a page fault occurs. The DOS/VS supervisor then performs a **page in** operation, locating the page containing the required code in the page data set, and bringing it into a page frame in the page pool. The interrupted program can then continue its execution. If all page frames are occupied, the system tries to locate a page frame not recently referenced and makes it available for the incoming page, after first **paging out** the contents of that page frame, if necessary.

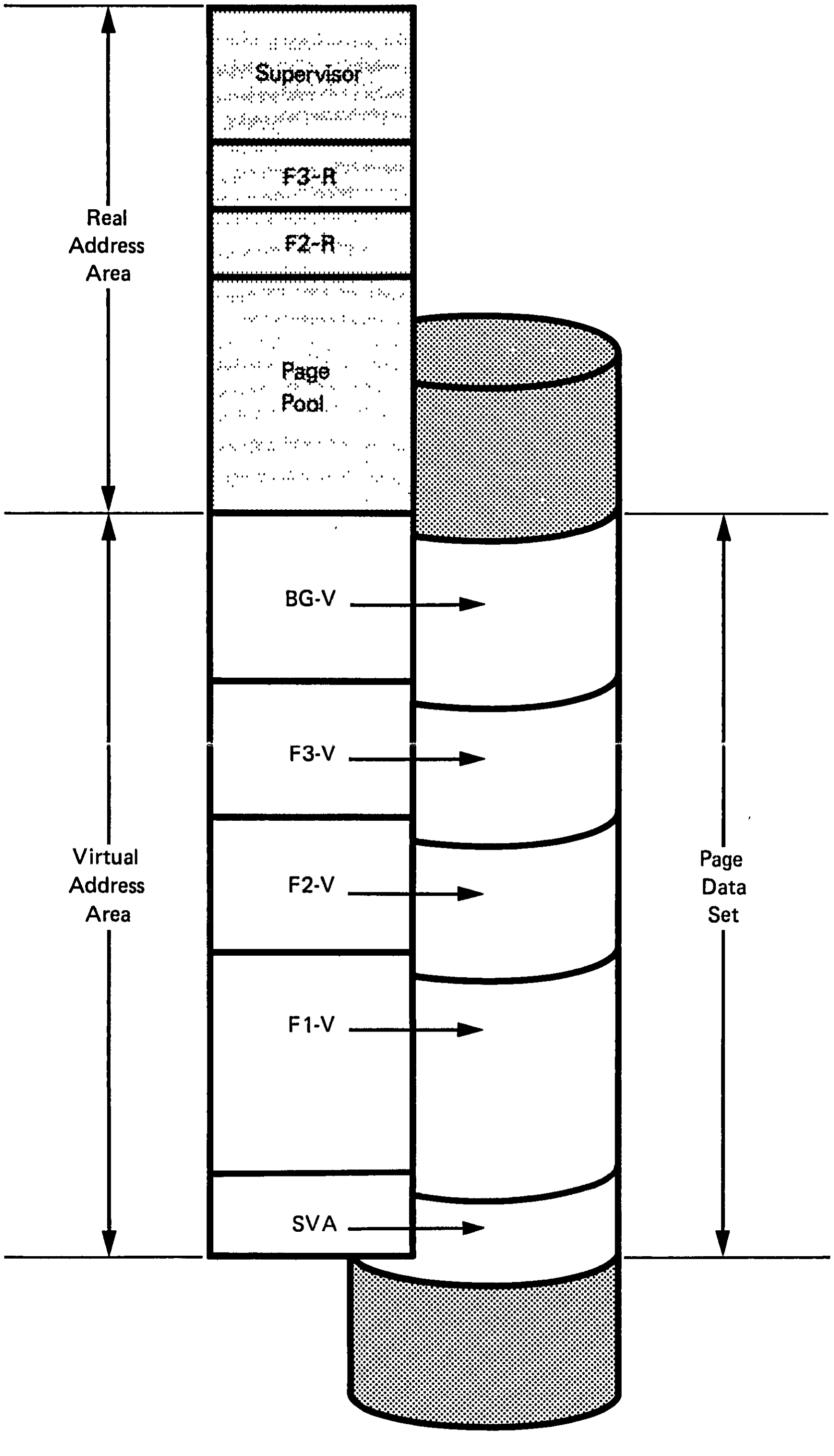


Figure 2.11. Page data set

The capacity of the page data set is equivalent to the size of the system's virtual address area.

fixing pages in real storage

Some programs that run in virtual mode contain code that must be in real storage at a certain time and therefore cannot tolerate paging. In such cases the page or pages must be fixed in real storage and not written out onto the page data set.

The supervisor always fixes an I/O area until successful completion of the operation. There are other parts of some programs, however, that also cannot tolerate paging, and these parts are not necessarily kept in real storage by the system. For instance, I/O appendages and programs that control time-dependent I/O operations cannot tolerate paging. The user can avoid page faults in these programs by fixing the affected pages in real storage.

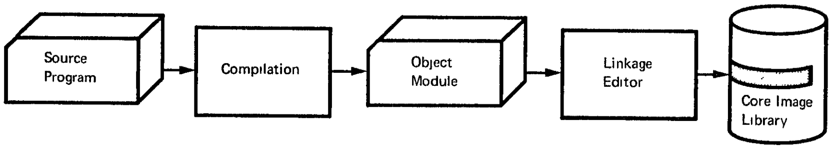
Fixing pages in real storage means that in a multiprogramming environment fewer pages are available to other programs running in virtual mode, potentially degrading system performance. That is why the system frees the pages it fixed as soon as possible, to make the page frames available to all programs running in virtual mode. the user should do the same.

dynamic address translation

The system cannot anticipate where in real storage a page from a program running in virtual mode will execute, until the page is actually placed in a page frame by the DOS/VS supervisor. Therefore, the determination of absolute addresses is delayed and, in fact, takes place **dynamically** during program **execution**. This is accomplished by dynamic address translation which is a basic part of System/370 design.

The following points summarize how a program is prepared for execution in **virtual mode,** and the activity that takes place while-it executes.

- After a program is written and assembled or compiled, it is link-edited for a virtual partition or the user may specify link-editing for relocatable loading into any real or virtual partition. The linkage editor places all phases in a core image library. If the user requests it and if the phase is reenterable and relocatable, it is also declared SVA-eligible and placed in the SVA (provided it is indicated as such in the SDL).



- As a program is loaded for execution in virtual mode, pages are transferred from a core image library to page frames in real storage. If sufficient page frames are not available, pages are paged out to the page data set until the entire program has been loaded. Figure 2.12 illustrates this concept. If the program is in the SVA, it need not be loaded and can be executed immediately from the SVA.
- As program execution proceeds, pages not in real storage are retrieved by the system from the page data set as they are needed and placed in page frames in real storage for execution. If the contents of a page frame have been altered during execution (or if a duplicate copy of the replaced page does not exist on the page data set), the page is paged out onto the page data set before the new page is brought into that page frame. (See Figure 2.13.)

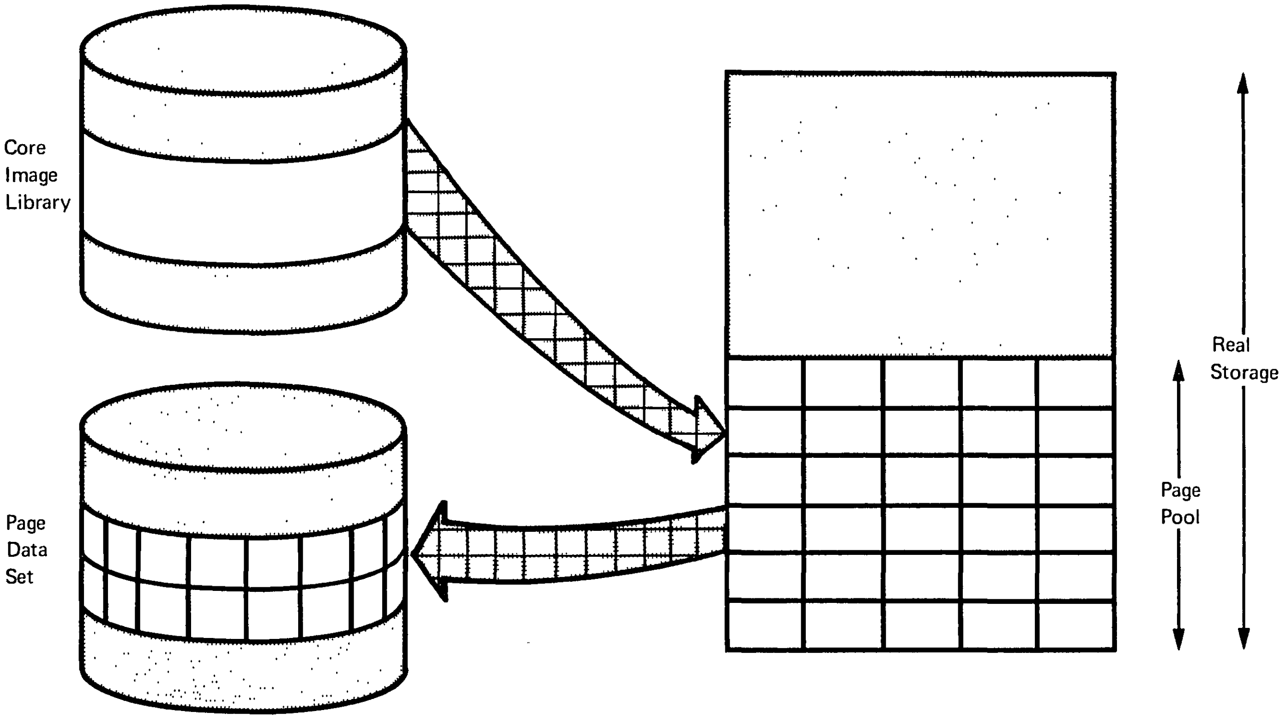


Figure 2.12. Loading a Program for Execution in Virtual Mode

During loading, pages spill over to the page data set if sufficient page frames are not available in the page pool.

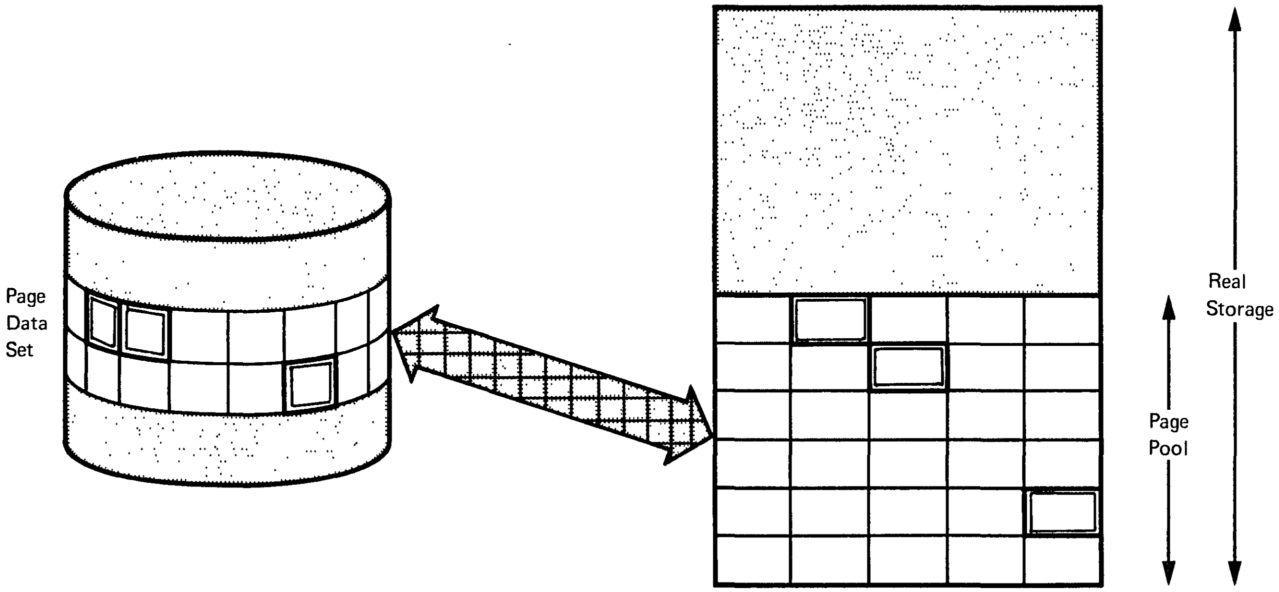


Figure 2.13. Paging between Real Storage and the Page Data Set

As execution proceeds, pages may be exchanged by the system between the page data set and page frames in the page pool.

- Address translation is accomplished as each instruction is executed by a combination of System/370 dynamic address translation and functions of the DOS/VS supervisor.
- All page frames in the page pool are available to any of the programs currently executing in the virtual mode. Designation of page frames is done by the DOS/VS supervisor which works toward keeping frequently used pages in real storage while placing new pages, as they are called in during execution, in page frames occupied by code no longer required, or, at least, less frequently required.
- Four programs executing concurrently in the virtual mode might be represented as shown in Figure 2.14.

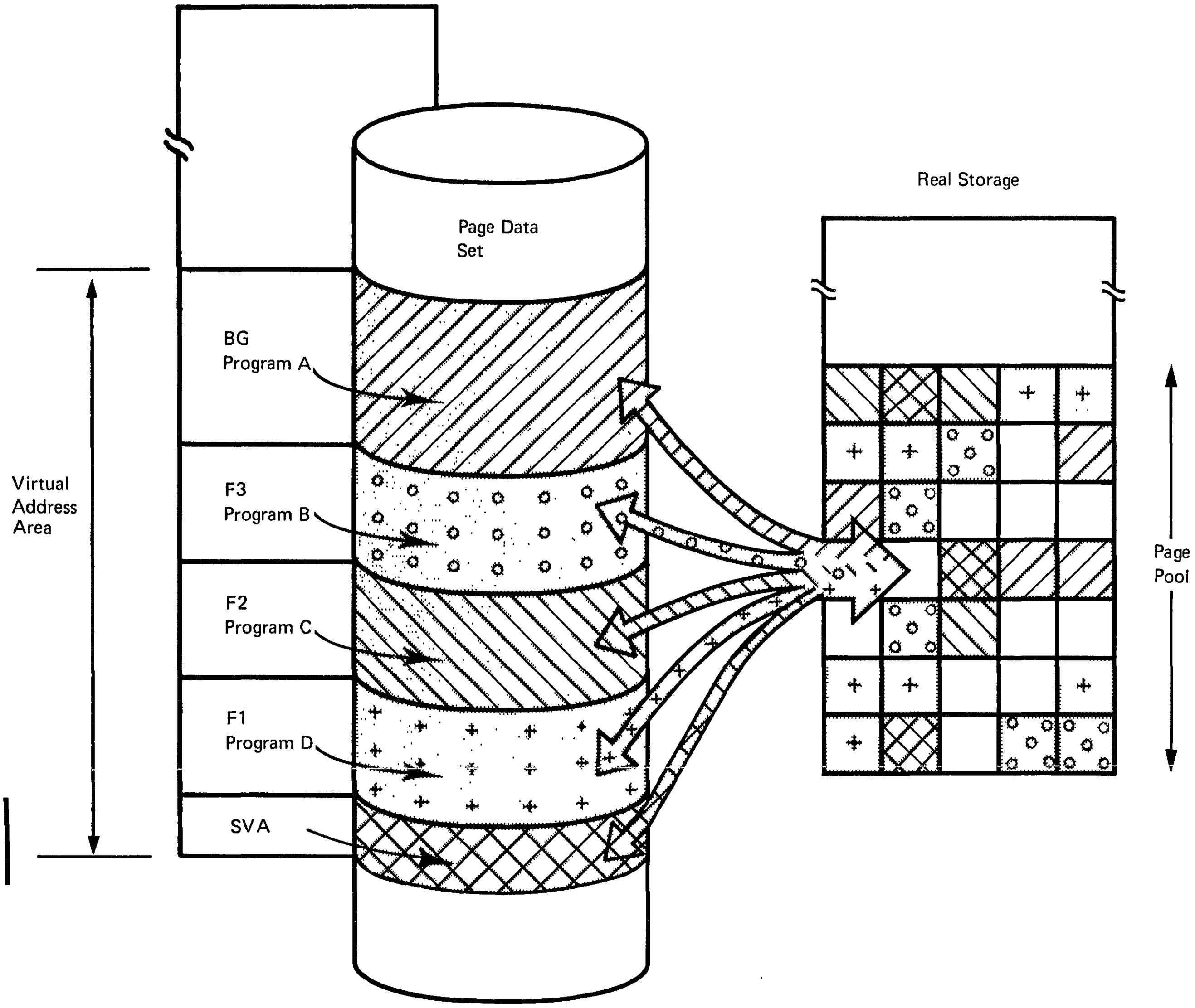


Figure 2.14. Four Programs Executing in Virtual Mode

Assignment of of page frames is done by the supervisor, which works toward keeping the most frequently used pages of each program in real storage.

Performance Considerations

System performance is usually measured in terms of system throughput, or the total amount of productive work accomplished by the system in a specific length of time. A number of factors have influenced system performance in the past, and virtual storage support introduces an additional variable which can enhance performance if used properly or degrade performance if used indiscriminately.

Performance can be **enhanced** by exploiting the DOS/VS capability of dynamically allocating the real storage making up the page pool to those programs currently executing in virtual mode. DOS/VS can more closely approach maximum utilization of real storage than was practical with earlier versions of DOS, which could accomodate only real storage partitions that were relatively fixed in size. Performance can be **degraded** when the size of programs, and the virtual partitions in which they run, is extended to a point that is disproportionate to the size of the page pool. This condition may cause excessive paging, and consequently can slow the useful production of the computer system.

Therefore, choosing this point of optimal relationship between the size of a program to run in virtual mode (plus the size of the others concurrently running in virtual mode) and the size of the page pool is important in achieving a properly balanced system. An easy solution would be to say that the sum of these program sizes should equal or only slightly exceed the page pool size. This would eliminate paging or reduce it to a minimum. But this solution would be incorrect because it considers only the factor of program **size** without considering program **characteristics.**

Programs that are well adapted to paging tend to have a modular structure, in which code and data for each subroutine or for each type of record or transaction is kept physically contiguous within the program; routines for error handling or unusual situation routines are kept as separate subprograms, away from the main section of the program. In very general terms, small programs frequently do not have such a structure and are therefore not well adapted to a paging environment. Large programs more frequently possess these characteristics (often because the sheer size of the program forces a "subroutine" approach for implementation). For this reason they tend to be better adapted to paging.

With a knowledge of the appropriate programming style and techniques, programs can be written with structures optimized for execution in a paged environment. VSAM, the virtual storage access method available in DOS/VS, is an example of coding structure adapted to paging requirements. VSAM functions require as much as 200K bytes of virtual address space, but execute efficiently in real storage only a fraction of that size.

Other factors must also be considered when balancing program sizes against available real storage. The operational requirements of an installation must be taken into account. In some cases execution efficiency and total throughput will be an installation's prime objective. In other instances, an installation may easily tolerate slower performance (because of more frequent paging) in order to have fewer constraints on program size. Larger programs may well be justified and can result from several causes:

- Use of a high level language rather than assembler. This usually results in greater programmer productivity.

- Programs that do more extensive processing in one pass of the data. This may obviate the need for additional runs.
- Functional segmentation of programs into logical units. This may allow a large application to be implemented faster by dividing the work among several programmers -- usually at the expense of code compactness.

DOS/VS can accomodate virtual partitions which the user knows will exceed the page pool size. He therefore has the options of realizing one or more of these advantages, at the expense of execution efficiency. The user must also consider the **degree** to which he "overcommits" his real storage. There is certainly a point beyond which he can not go in sacrificing execution efficiency for programmer productivity. The variables contributing to the definition of this point are numerous; some are oriented to the computer itself, such as real storage availability, CPU speed and utilization, and speed of the disk device containing the page data set. Other factors are oriented to the installation's operational environment, such as characteristics of the program library, (size of programs, I/O requirements, frequency of use and length of runs) present and projected shift usage, turn-around requirements and prescribed job sequences. In balancing all these factors to arrive at an optimal system definition, the management of an installation will probably wish to do some experimenting and tuning, varying job mixes, partition sizes and perhaps the number of active partitions. The point of departure for this experimentation should be a conservative one, in which any overcommitment of real storage is based on understanding both the justification for it and the anticipated result of it.

An Example of Partition Allocation

The examples used so far to illustrate partition allocation have not defined partition sizes. Consider one further illustration that does use specific storage allocations and in general terms defines the use of each partition. Figure 2.15 lists the partitions, their use, and the size of the real and virtual address areas allocated to each one. Figure 2.16 is a schematic representation of these storage allocations. Note that the figures in the example are not program sizes, but partition sizes. All virtual partitions and the shared virtual area are at least 64K.

Partition	Program Run Mode	Partition Use	Real Address Area Allocation	Virtual Address Area Allocation
Supervisor	Real	System Control	42 K	--
Background	Virtual	System Maintenance Large applications & tests (not frequently used)	0 K	256 K
Foreground 3	Virtual	Urgent jobs (non-scheduled, high priority jobs) Test runs	0 K	64 K
Foreground 2	Virtual	Normal Batch Production	0 K	96 K
Foreground 1	Real or Virtual	Real: POWER (36 K) Virtual: Job Control (64 K)	36 K	64 K
SVA	Virtual	Shareable programs	--	64 K
Total Allocated to Supervisor and Real Partitions.			78 K	
Total Allocated to Virtual Partitions and SVA.				544 K
Minimum Available Page Pool (When POWER is active).			72 K	
Page Pool when POWER is inactive.			108 K	

Figure 2.15. Example of Partition Definition

This example represents a Model 145 with optional features for the 3215, Integrated File Adapter, and Extended Precision Floating Point. Because of Control Storage requirements, the original 160K bytes of real storage are reduced by 10K, leaving 150K bytes for the real address area.

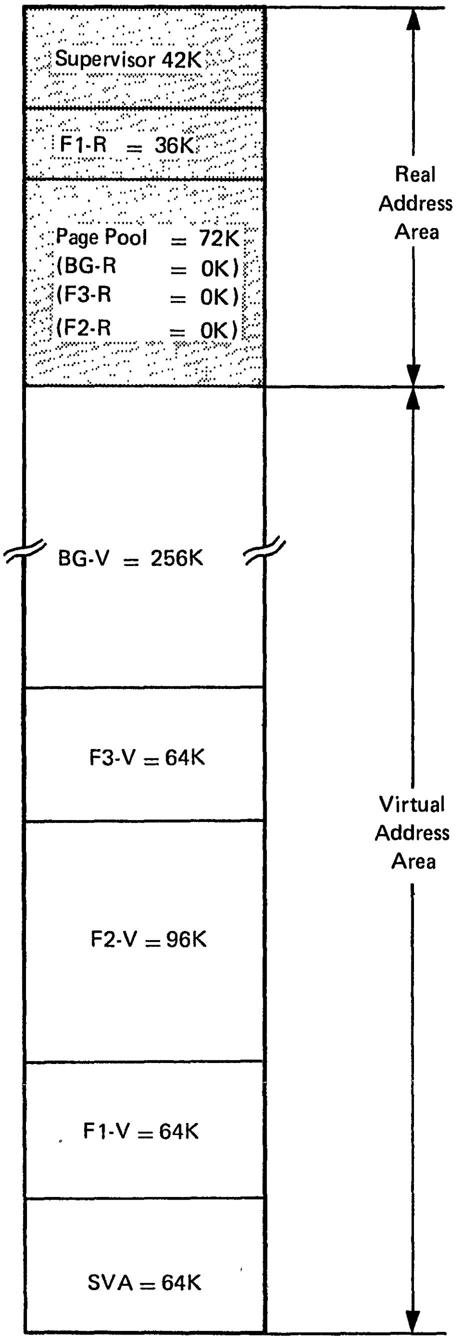


Figure 2.16. Storage Allocation Example

POWER

In all computing systems there is a large discrepancy between CPU speeds, which are electronic and therefore very high, and the speeds of card readers, punches, and printers, which are largely mechanical and therefore relatively slow. The user can lessen this discrepancy by running his jobs in the DOS/VS multiprogramming environment.

A program that can offer further improvement of system performance is POWER (Priority Output Writers, Execution Processors and Input Readers). This is an optional service program designed to reduce CPU dependence on the relatively slow speeds of card readers, punches, and printers.

POWER decreases the execution time of unit record I/O-bound jobs by servicing I/O requests addressed to such devices at disk I/O speed. The peripheral reading and punching of cards and the printing is done by POWER in parallel during the execution of other jobs. The additional CPU time used by POWER is negligible. In a typical environment of jobs with mixed characteristics, throughput may be substantially improved.

Additionally, POWER allows the user to multiprogram in up to four other partitions without the need for separate unit-record devices for each partition. The unit record devices used by POWER can provide the I/O requirements for all the partitions that are being serviced by POWER.

Processing with POWER is as follows (see Figure 2.17):

- POWER reads the job streams (job control statements, programs, and data cards) for the individual partitions and stores these in input queues on disk.
- From disk, the jobs are transferred by POWER to the partitions and executed.
- Unit-record output (printer and punch) of every job is stored on disk by POWER before it is finally printed and/or punched.

Job input as well as job output may be held in the POWER queues for execution or printing/punching at a later time. This allows the user to hold jobs that need, for example, two hours of execution or printing time until the system is less occupied.

Execution of POWER may be controlled by the operator. He may start and stop input reading and output printing/punching independently of job execution.

Whenever a card reader, a punch, or a printer becomes inoperative or fails, the system can continue processing with those jobs already in the input queues on disk and store the output of these jobs in the output queues on disk. When the I/O unit becomes available again, reading, punching, or printing can continue.

Remote Job Entry

POWER also offers a teleprocessing facility: Remote Job Entry (RJE).

With POWER RJE, the user may enter jobs for processing from remote terminals. RJE supports up to five 2770, 2780, or 3780 terminals on the same number of leased or dial-up lines. Once a job has been entered into the input job queue, execution proceeds under DOS/VS supervision. All data files required by the job are subject to DOS/VS specifications, just as if the job had been entered locally.

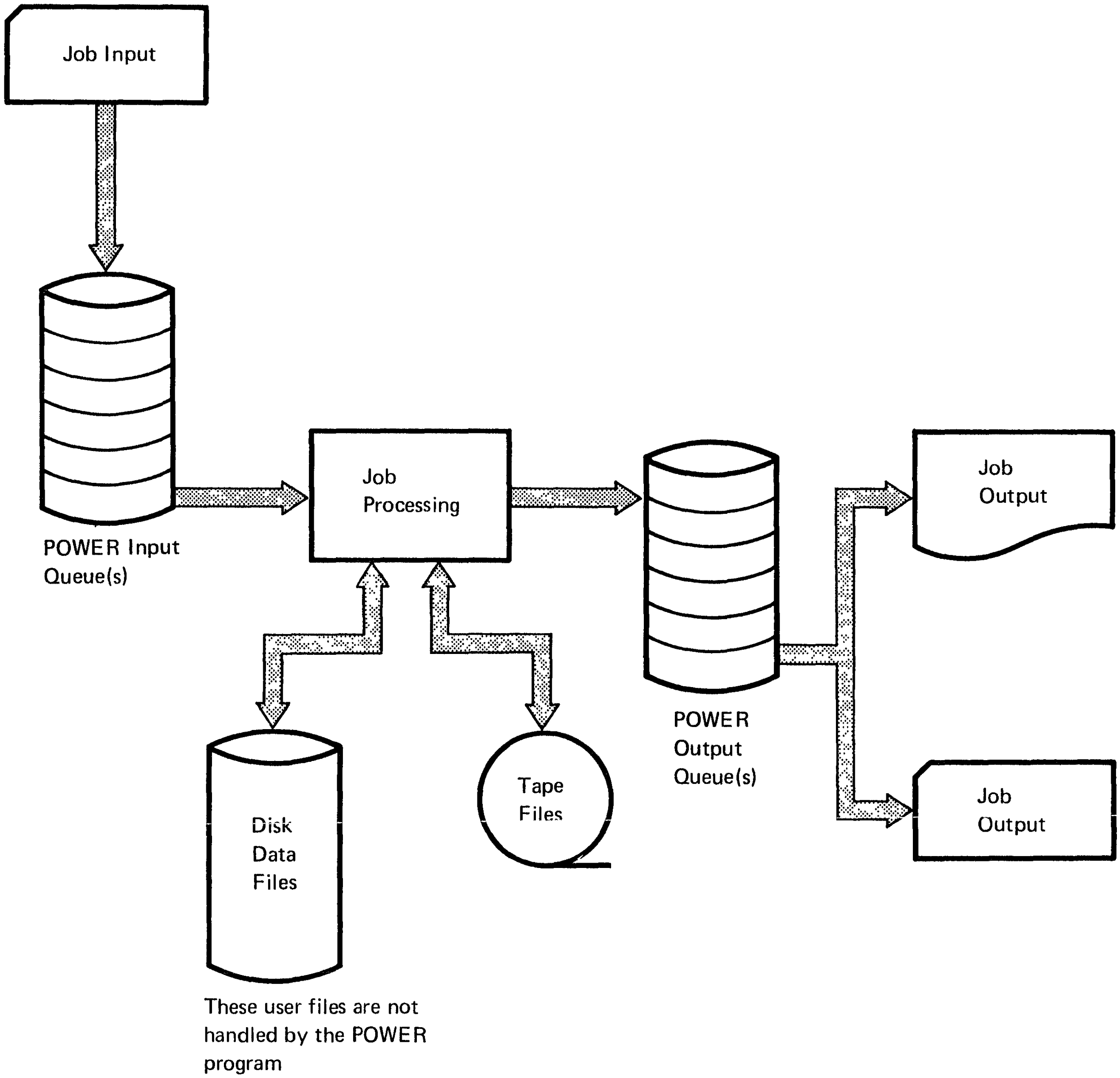


Figure 2.17. Processing with POWER

A job's normal punched-card input is read from the card reader and queued on disk before the start of a job. Similarly, a job's output is stored on disk in an output queue and printed and punched at a later stage.

RJE job output may be directed to the terminal from which the job was entered, to other terminals, or to the normal (local) output units of the system. It can be immediately transmitted to the specified unit-record device or be held in the POWER queue for printing or punching on demand.

Performance with POWER

The performance of DOS/VS with POWER is best illustrated by a typical example, shown in Figure 2.18. The upper part shows the processing of a DOS/VS job stream in a partition without the POWER facilities. Note that the first job, which needs a great deal of printing time, slows down throughput. The lower part of the figure shows the processing times of the same job stream, with POWER. Here, the total time is divided into CPU time, queue time (the time the output of the job is in the print queue, ready for printing), and printing time. (Input queuing is not considered in this example.)

Although the time elapsed between the reading of a job and the completion of its output increases under POWER, overall system performance is improved considerably. This can be seen from the difference in time between points 2 and 3, when all five jobs have finished processing. In addition the CPU is available for processing, because between points 1 and 2 the only activity here is the printing of the output queues, which requires little CPU time.

This is only a theoretical example. The actual increase in throughput that may be achieved depends, for example, on the CPU or I/O orientation of each program, the sequence of the particular jobs, the number of partitions supported, and the speed and number of unit-record devices.

Practical Considerations for Using POWER

The POWER program is distributed as a set of macro instructions and phases from which the user may assemble his own version according to his needs.

One option is to generate a reduced version that performs printing and punching only (called a writer-only system).

Through operands of a POWER generation macro, the user may specify that input and output of queues be started automatically upon initiation of POWER, rather than under operator control.

The only DOS/VS unit-record device not supported by POWER is the IBM 2596 Card Read Punch.

POWER always has to run in a real partition, which must have higher priority than any other partitions serviced by POWER.

The user has the option to store POWER output queues on tape instead of on disk.

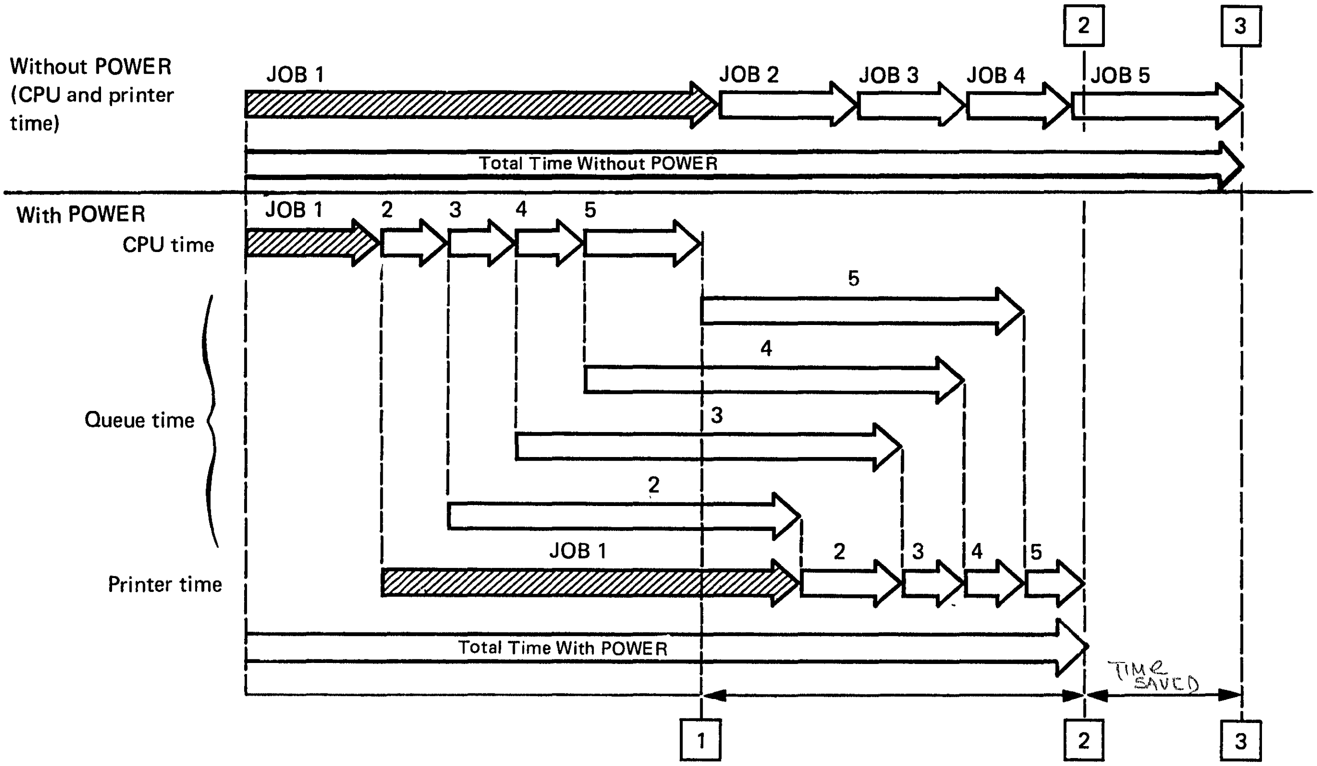


Figure 2.18. Processing Five Jobs With and Without POWER

The improvement in system performance achieved by output queueing under POWER is shown by the difference in time between points 2 and 3.

Job Accounting

A DOS/VS user can write a simple program to keep track of CPU usage and the usage of the various I/O devices by accessing the job accounting data accumulated by the system. This enables the installation manager to:

- Charge usage of the system to the various users
- Help supervise system operation
- Check on efficient use of I/O devices
- Plan for new applications, additional devices, and new systems.

The necessary information for this program is provided by the job accounting interface, an optional feature specified during system generation. With this feature, the following information is automatically gathered by the system for each job step and stored in a table in the supervisor area:

- Job name, date, and partition in which the job is running
- Start and stop times of the job, CPU time used by the program, CPU time used by the control program (overhead), and CPU idle time chargeable to that partition
- Optionally, counts of the operation of the various I/O devices.

At the end of each job step, the user's accounting program is automatically loaded into the partition where the job step has just finished processing, either to transfer the information from the job accounting table to auxiliary storage (tape or disk) for future use, or to format and print the information. The next job or job step is then started.

Each installation must provide its own accounting program to charge the various users for the computer time used, or to analyze the performance of a program or job stream. For more details on this subject refer to the *DOS/VS System Management Guide.* The job accounting procedure is summarized in Figure 2.19.

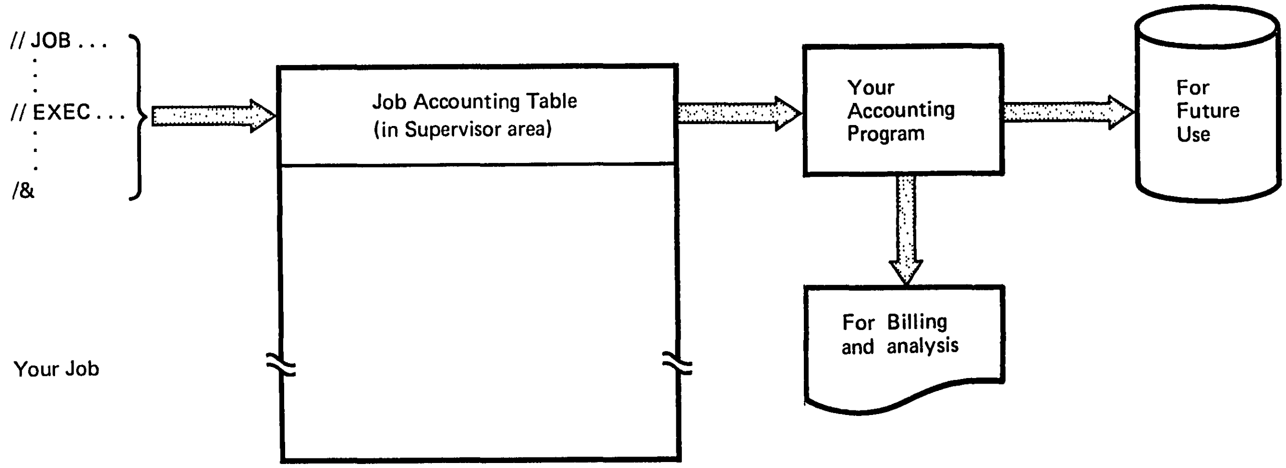


Figure 2.19. Job Accounting Procedure

An appropriate accounting program extracts and analyzes the data in the job accounting table and prints it or stores it on disk (or tape).

Libraries

One of the most powerful features of DOS/VS is its range of libraries, which enable programming data to be stored online in readily accessible form.

PROC =
PROCEDURE LIBRARY

DOS/VS supports four types of libraries: core image, relocatable, source statement, and procedure. The first three types of libraries exist in two different classes, system libraries and private libraries, whereas the procedure library exists only as a system library. The system libraries are contained in the system residence file (SYSRES), and can be accessed by all partitions. Private libraries can be contained on separate disk packs, and can be accessed only by programs in the partitions to which they are assigned. DOS/VS also supports the shared virtual area (SVA), which is closely related to the system core image library.

The system librarian program provides service and maintenance functions for all types of libraries. In addition, two system programs are related to core image libraries. These are the linkage editor and the loader, which is a part of the supervisor. DOS/VS offers the user two types of loader programs, the relocating and the non-relocating loader. The linkage editor also services the SVA if the required phase is SVA-eligible and is indicated as such in the SDL.

Using the Libraries

The DOS/VS libraries have the following main functions:

core image library

The core image library serves to catalog programs in units, called phases, that have been processed by the linkage editor and are ready for loading. Each system must contain a system core image library in order to catalog certain system programs.

In DOS/VS, phases can be in either non-relocatable or relocatable format. A non-relocatable phase is loaded directly at the address computed at link-edit time into a real or virtual partition. A DOS/VS feature now allows the linkage editor to produce relocatable phases as well. The load addresses, entry points, and the address constants of these phases can be modified by the relocating loader, and such phases can, therefore, be loaded at storage addresses different from the ones for which they were link-edited.

shared virtual area

The shared virtual area (SVA) is located in high virtual storage. It contains a system GETVIS area, a system directory list (SDL) of frequently-used phases, and code in executable format. The GETVIS area is located in the high end of the SVA. Phases that are relocatable and reenterable can be placed in the shared virtual area. These phases are also in the core image library. The name of each phase in the SVA is stored in the SDL with an indication that the phase is loaded in the SVA.

The linkage editor produces phases that are placed in the core image library and in the SVA when the user has requested it. These phases are relocatable and can be used by all partitions in the system. Because they are already in virtual storage, they need not be loaded again before execution.

relocatable library

When the linkage editor encounters a reference to a module not being processed, it searches the relocatable library for a module with the name specified in the external reference. If the search is successful, the module found is linked with the modules being processed.

Explicitly specified modules from the relocatable library can be included with modules being link-edited. In this way, sections of code that are used by a number of different programs need be written, translated and cataloged in relocatable object format only once.

source statement library

The source statement library contains sequences of source language statements, called books. The library consists of a number of sublibraries used, for example, to store macro definitions. When the assembler encounters a source statement with an unknown operation code, it tries to retrieve the book with the same name as the unknown operation code from the source statement library. It then substitutes the code found in the library for the source statement in question.

Similarly, when a compiler encounters a reference to a book in the source statement library, it gets the specified book from the library and substitutes it for the reference in the source program it is processing.

procedure library

throw away /+ & /*
cards at end
use a name card at
beginning PROC

The procedure library is used to catalog frequently used sets of job control and linkage editor statements in card image format. Depending on a system generation option, procedures may contain inline data, such as utility modifier or librarian control statements. Cataloged procedures can be included in the job control input stream and may be modified by "overwrite" statements as the job stream is processed.

private libraries

In addition to system libraries, DOS/VS offers the user the option of private core image, relocatable, and source statement libraries. These are particularly useful under the following circumstances:

- In an installation with a large number of programs or applications.
- In an environment where, for security reasons, it is desirable that certain programs be kept under lock and key.

Linkage Editor and Relocating Loader

The output of a language translator is an object module in machine language that may either be punched into cards (SYSPCH), which can be cataloged into the relocatable library, or stored in an intermediate storage area on disk (SYSLNK) if link-editing is to follow translation immediately.

Assembly or compilation of source programs is not affected by virtual storage considerations. The generated object program is the same whether it is to be executed in real or in virtual mode.

Object modules cannot yet be executed for the following reasons:

- Programs, in most cases, have to be relocated to the partitions in which they are to run.

- References to locations or labels in other modules (that is, external references) may still have to be resolved.

Relocation and resolution of references are done by the linkage editor which gets its input from SYSLNK and, optionally, from a relocatable library. The output - executable programs, called phases, with specific load addresses - are always stored in a core image library, either permanently or temporarily. If the output phases are reenterable and relocatable, they are also placed in the SVA if the user has so requested. The phases are executable, either directly or after processing by the relocating loader.

In earlier versions of DOS, whenever a user program had to be able to run in any partition, the program had to be self-relocating, or three different copies had to be present in a core image library, each link-edited for one of the three partitions, and each with a different name.

DOS/VS, however, allows the linkage editor to make its output relocatable. The phase then contains relocation information, and the relocating loader in the supervisor relocates the phase, if necessary, when loading it into the partition that the user selects at execution time.

This feature is of particular advantage (1) in a multiprogramming environment, where it can save a considerable amount of processing time and disk storage space, as shown in Figure 2.20, (2) in a single partition environment where programs are to run in real mode at one time and in virtual mode at another time, and (3) when the supervisor size has increased in which case it saves relink-editing of programs that are loaded at the end of the supervisor.

The relocating loader is an optional feature. Its inclusion in the supervisor has to be specified during system generation. Figure 2.21 summarizes the possibilities of link-editing and relocating prior to execution.

Figure 2.22 shows how the libraries, the language translators, and the linkage editor fit together in DOS/VS.

Librarian Program

DOS/VS contains a librarian program that performs maintenance and service functions for all libraries. These functions include:

- Cataloging and deleting of any element in a library.
- Printing of any element or directory.
- Punching of any element.
- Copying of elements from one library to another of the same type.
- Condensing and changing the size and location of the libraries.
- Creating a new system disk pack and private core image, relocatable, and source statement libraries.

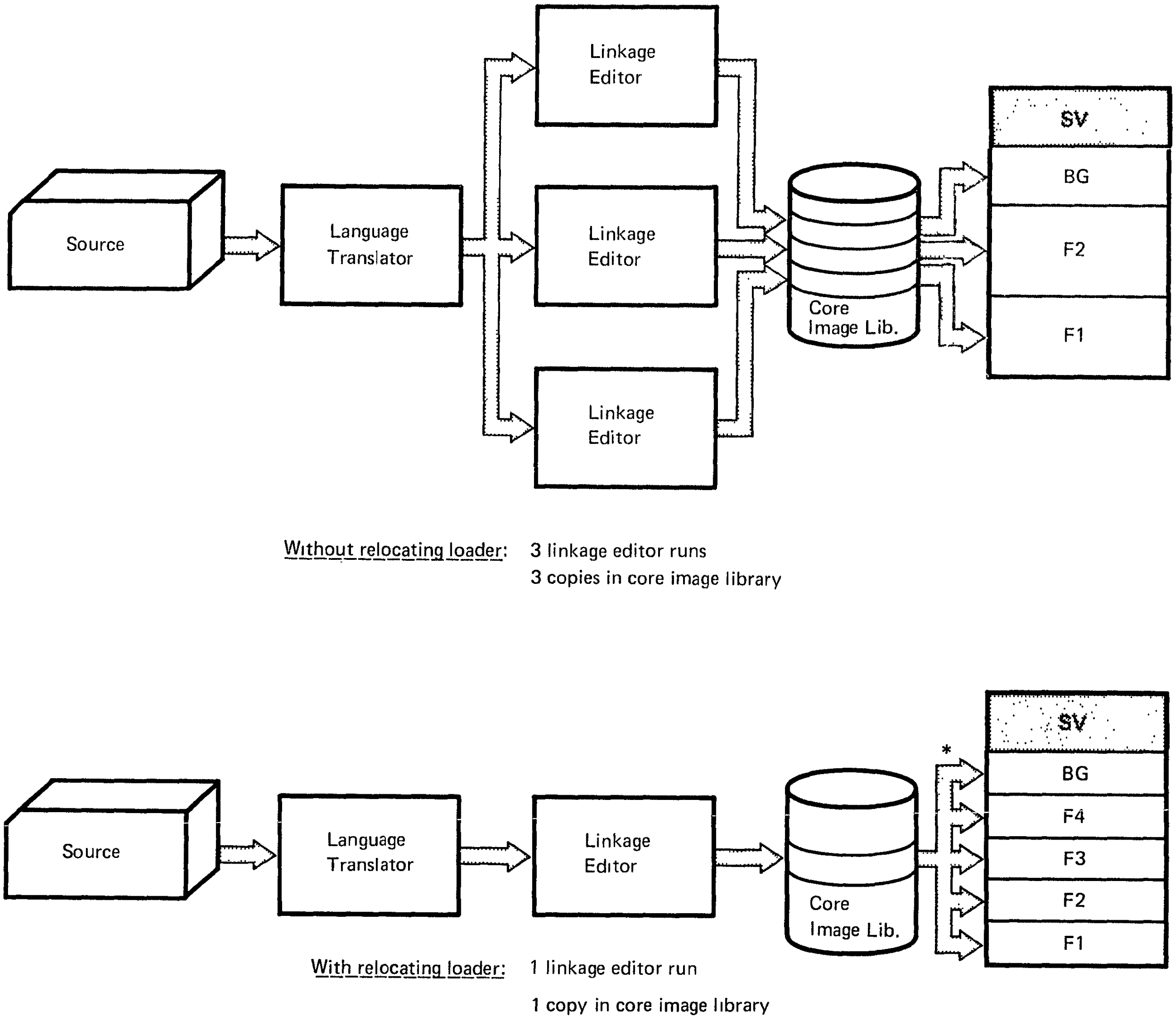


Figure 2.20. Linkage Editing With and Without Relocating Loader Feature

Contrast the multiple link-editing plus multiple core image library copies of user programs without relocating loader with the single (relocatable) core image library copy needed when the relocating loader feature is included.

* This step (program loading) is not required for programs that are contained in the SVA. Such a program is executed directly from the SVA, regardless of the partition by which it is called.

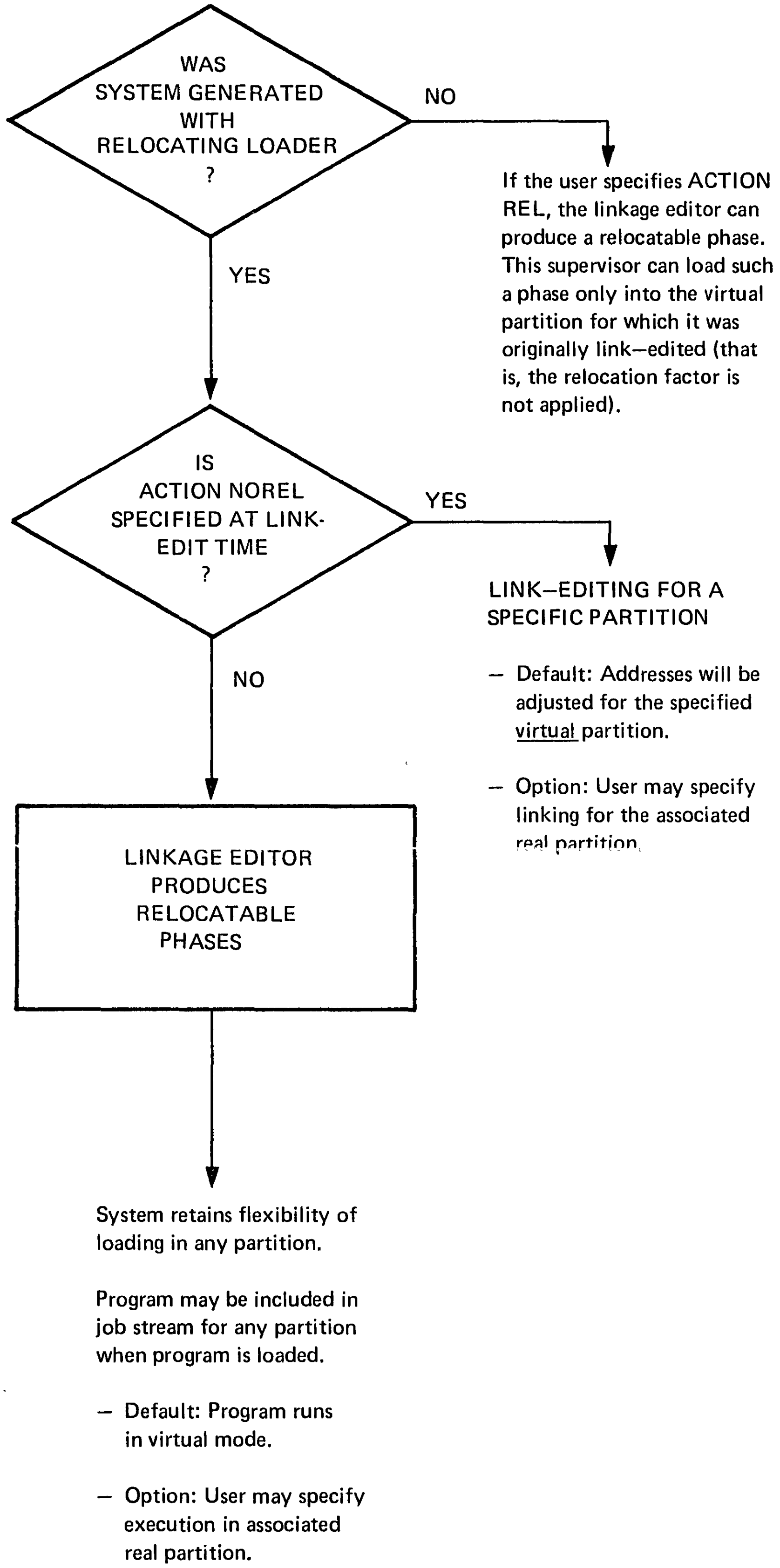
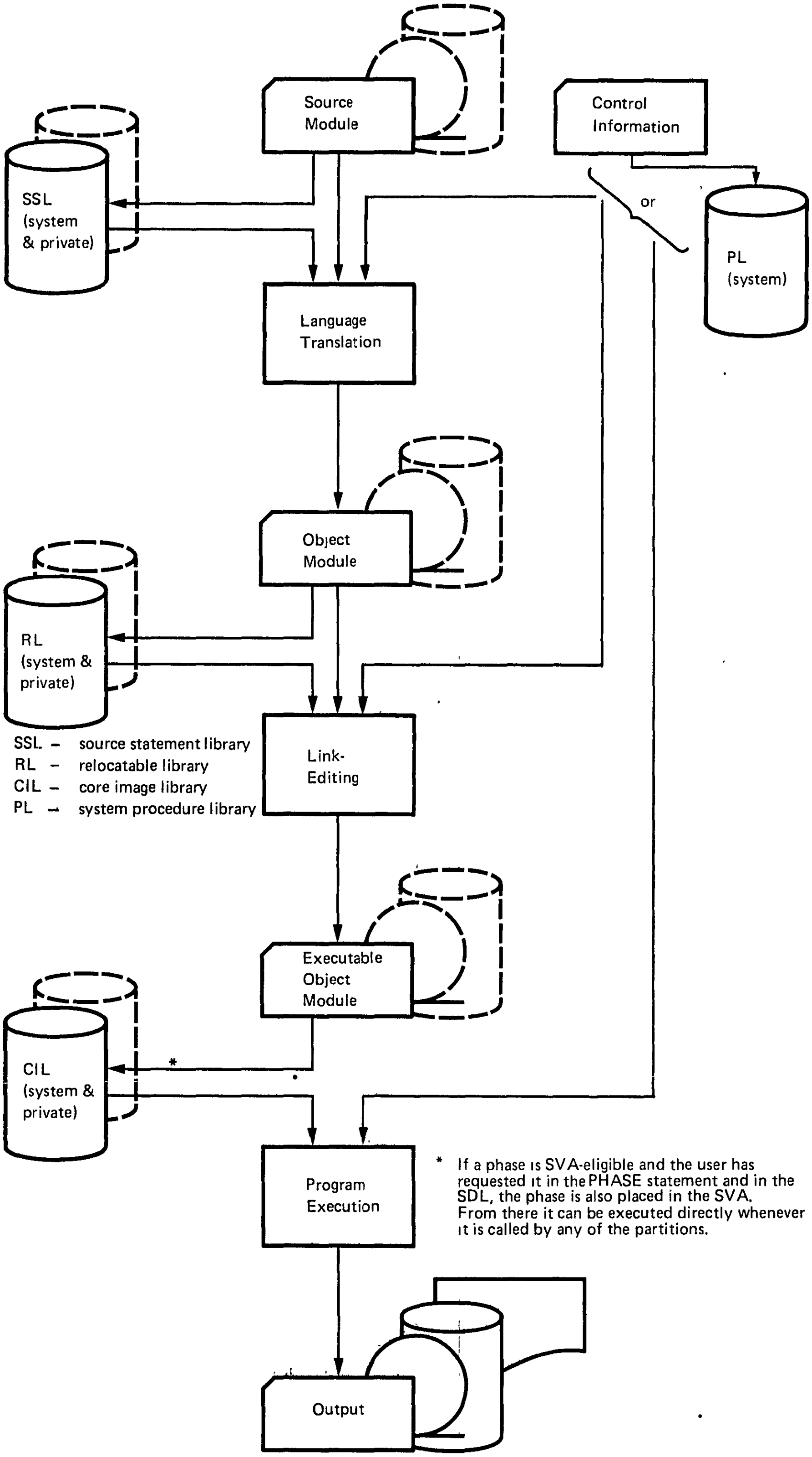

WAS SYSTEM GENERATED WITH RELOCATING LOADER ?
NO
If the user specifies ACTION REL, the linkage editor can produce a relocatable phase. This supervisor can load such a phase only into the virtual partition for which it was originally link–edited (that is, the relocation factor is not applied).
YES
IS ACTION NOREL SPECIFIED AT LINK-EDIT TIME ?
YES
LINK–EDITING FOR A SPECIFIC PARTITION
– Default: Addresses will be adjusted for the specified virtual partition.
– Option: User may specify linking for the associated real partition.
NO
LINKAGE EDITOR PRODUCES RELOCATABLE PHASES
System retains flexibility of loading in any partition.
Program may be included in job stream for any partition when program is loaded.
– Default: Program runs in virtual mode.
– Option: User may specify execution in associated real partition.

Figure 2.21. Options Available During Link-Editing



* If a phase is SVA-eligible and the user has requested it in the PHASE statement and in the SDL, the phase is also placed in the SVA. From there it can be executed directly whenever it is called by any of the partitions.

Figure 2.22. Interrelationship of Language Translators, Linkage Editor, and Libraries

Data Management

Data storage and retrieval requirements, and how the data processing department responds to those requirements are often essential concerns of everyone affected by the data processing operation.

Some basic questions involving data management are:

- What **processing requirements** is each data file to be subject to?
- What type of **file organization** is best suited to the processing requirements for the file?
- What **medium** (magnetic tape, cards, disk, etc.) is each data file to be stored on?
- What **data security** considerations should apply to the file during and after its processing cycle?

How the DOS/VS data management facilities provide a means of arriving at appropriate answers to these questions is outlined in the sections that follow.

Data Organization and Access Methods

data organization

Data Organization refers to the techniques used in placing records on an auxiliary storage device such as cards, magnetic tape or disk. It involves such considerations as:

- The choice of **storage media** best suited to the processing requirements of the data.
- The **sequence** of the individual records in the file. For example, the file could be sorted on one control field or on several, in a prescribed hierarchy; the file could be in ascending or descending order.
- The **length of records**. Records in a file can be of a fixed length or of variable length.
- The **blocking factor** for the file. This determines how many logical records constitute a physical record, and is an important factor in storage media utilization and in processing efficiency.
- The use of **indexes** with a file on a direct access device to provide an efficient means of randomly selecting specific records.
- The use of programmed **addressing techniques** to determine where a record is stored on a direct access device and the location from which it can subsequently be retrieved for processing.
- The type of **data additions** to an already existing file. It is of major importance whether many or few data additions and updates are made to a file, whether additions and updates are made in sequential or random order, whether a file is accessed by more than one program in different partitions, and whether it is a read-only file.

access methods

Access methods refer to the routines which assist the programmer in transferring records **in a particular data organization format** between storage and an I/O device. It is important to understand the relationship between data organization and access methods. Broadly speaking, how data is organized and the type of device that it is stored on largely determine the access methods that can subsequently be used to retrieve it. DOS/VS provides several methods of data organization. For each of these there is an access method which allows one or more techniques of file creation and retrieval. The following sections briefly describe these data management facilities.

Sequential Access Method (SAM) and Organization

file organization

Sequential organization means that records physically follow one another in a sequence usually determined by one or more control fields within each record. Examples of control fields are name or man-number in a personnel file, or catalog number or part number in an inventory file.

Sequential organization is the most widely used method of data organization and is supported for all device types except teleprocessing terminals. Card files, print files, diskette unit files, and magnetic tape files are always organized sequentially, simply because the physical characteristics of those devices require the reading or writing of one record after another. Data files on disk are also frequently organized sequentially, in control number sequence.

data access

If required, records are sorted into their prescribed sequence prior to, or as a part of, creating a sequentially organized file. The Sequential Access Method (SAM) can create a sequential file from the sorted records presented to it and subsequently retrieve those records for sequential processing. In addition, by utilizing certain macros, sequential files on disk or tape may be positioned to specific physical blocks prior to reading or writing. Records from sequential disk files may be "updated," meaning that each record may be written back onto its original physical location after having been changed by the program.

applications

Sequential organization and access methods are used for some files in most data processing installations since the requirements of many applications are met entirely by "batch processing". This means that transactions are held and batched until a number of them are available for processing against a master file. The batch of transactions is then sorted into the same sequence as the master file, which is then updated at periodic intervals, such as daily, weekly or monthly, depending on the volume of activity and the need for keeping the master records current. Payroll files are frequently organized sequentially; they are, typically, processed once per pay period with transactions consisting of employee time cards, piecework records or similar applicable data, producing checks and earnings statements and updating year-to-date figures in the master records.

Figure 2.23 shows how tape and disk sequential files appear.

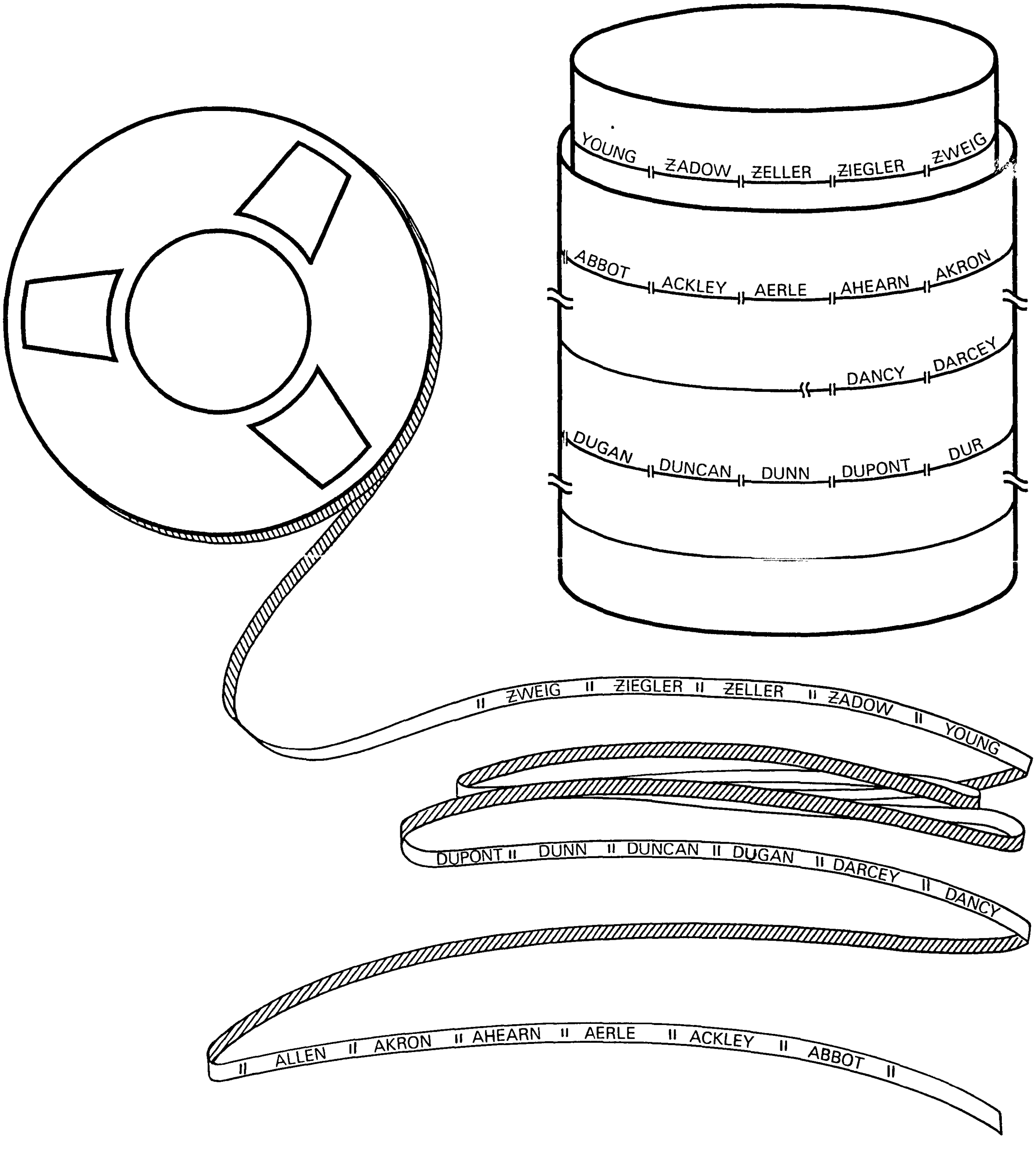


Figure 2.23. Sequential Data Organization

Records of a sequential file are arranged in the order in which they are processed. In this instance the order established is alphabetical. Note that only small segments of the file are shown and that only the control field by which the file is organized is shown. The remaining data in each record is irrelevant in this context.

Indexed Sequential Access Method (ISAM) and Organization

The physical characteristics of a disk make it practicable to retrieve a record from any location in the file, instead of having to go to the next one in physical sequence. DOS/VS exploits this capability by providing the indexed sequential access method and organization.

index and file organization

An indexed sequential file is made up of (1) records in logical sequence by control field (or key) and (2) an index, which is built when the file is created. The index itself is structured in two or three levels. Each index entry is composed of the key of a data record, or a lower level index entry, and the physical address at which the record, or lower level index entry, is located on the disk. The programmer may process indexed sequential files **sequentially** when the definition of the programming application requires this approach, or process them **randomly**, retrieving a particular relevant record from the entire file, if this approach is better adapted to the

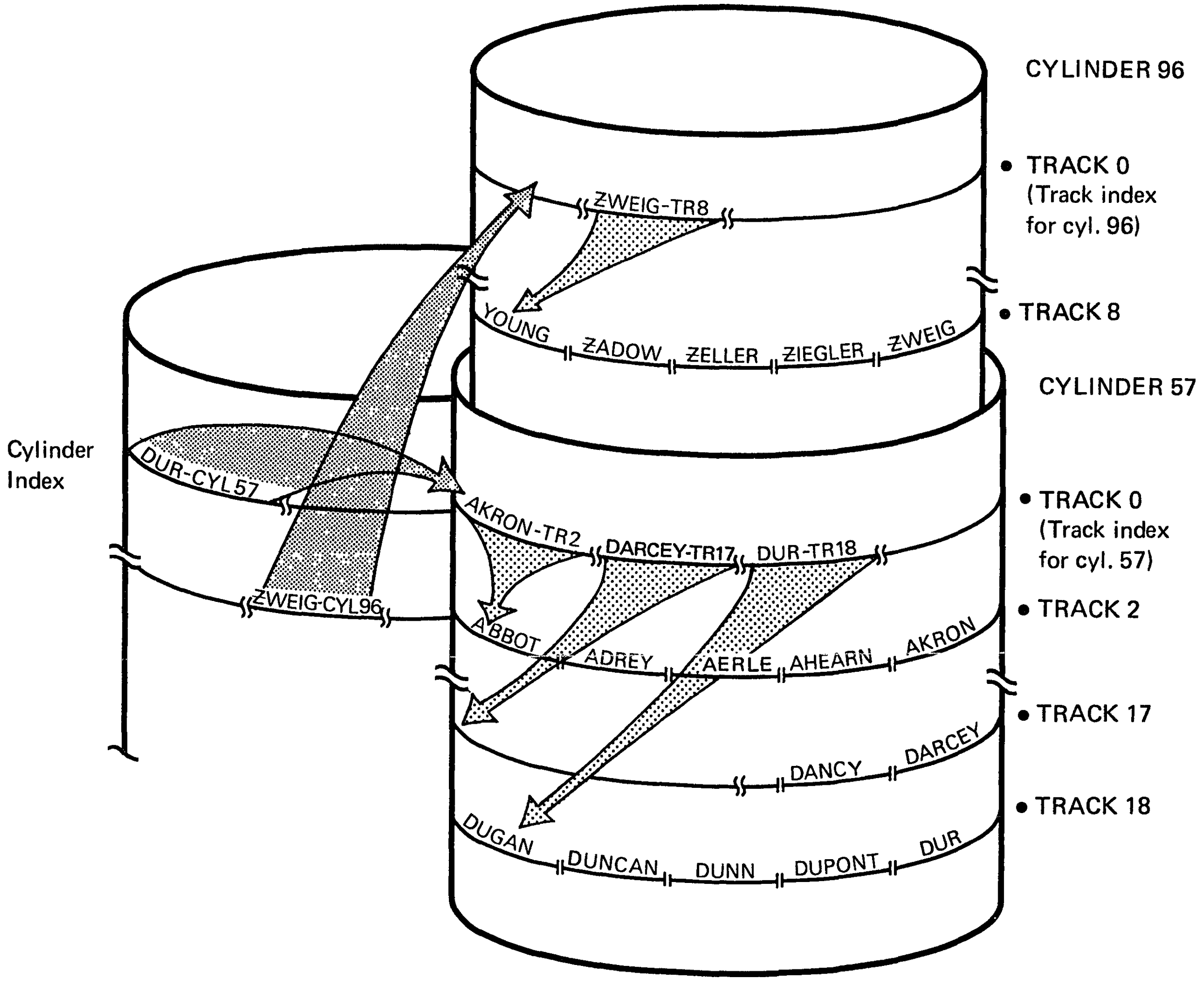


Figure 2.24. Indexed Sequential Data Organization

Records of an indexed sequential file are arranged in logical sequence by key. Indexes to these keys permit direct access to individual records. All or part of the file can be processed sequentially. In this illustration, the file starts on cylinder 57 and ends on cylinder 96.

requirements of the job. He may even combine both facilities in the same program. Both retrieval methods are easy from the programmer's viewpoint:

data access

- For **sequential** retrieval he simply issues the appropriate I/O command or macro, such as GET, at the point in his program where he requires the next record, and ISAM makes the record available to him.
- For **random** retrieval the programmer merely provides the key of the record he needs, issues the appropriate command, and ISAM presents the specific record to him for processing. Index searching, deblocking records, and handling device requirements are all accomplished internally by the access method.

overflow area

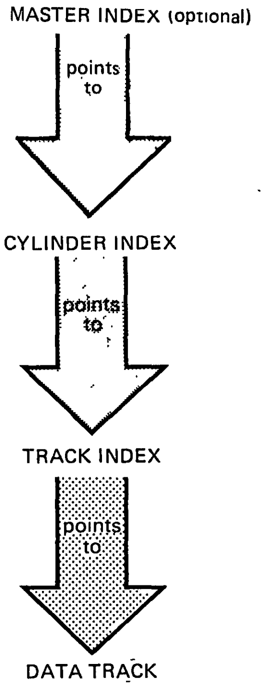
ISAM also provides the routines for creating a file from sorted input, building the index, and adding records to an existing file. For the insertion of records into an existing file, additional disk space, called overflow area, is reserved. On sequential retrieval of a file that has data in the overflow area, records are retrieved in logically sequential order.

applications

This degree of processing flexibility makes ISAM attractive in many applications. It is frequently used in inventory record maintenance, where, for example, sequential retrieval is most convenient for stock status reports and batch transaction processing of non-critical items, but where random retrieval is necessary for real-time inquiry or for stock maintenance of high turnover merchandise.

The sequential file illustrated in the previous section may be represented in indexed sequential format as shown in Figure 2.24.

As new records are added to an indexed sequential file, exceptions will occur. However, the access method handles the insertions, and both sequential and direct retrieval requirements, regardless of insertions. For direct retrieval, ISAM follows the sequence:



Virtual Storage Access Method (VSAM) and Organization

VSAM is a new access method for direct or sequential processing of fixed and variable length records on direct-access devices. It has more functions, generally better performance, better data integrity and security, improved data organization, and is easier to use and control than the DOS/VS DAM and ISAM access methods.

file organization

The records in a VSAM file can be organized either in logical sequence by a key field (key-sequence) or in the physical sequence in which they are written on the file (entry-sequence). The user can read, add, delete, and modify records in a VSAM file. He can access the records sequentially or directly by key or by address.

A key-sequenced file has an index, like ISAM; the records in a key-sequenced file can be accessed by key or by address. An entry-sequenced file does not have an index, and records can be accessed by address only. It can be processed like a sequential file or a direct file.

key-sequenced files

Like indexed sequential files, key-sequenced files are ordered according to a user-defined key field in each record. Key-sequenced files, however, can generally be processed faster than ISAM files because VSAM has a more efficient index and does not use chained record overflow.

When a key-sequenced file is created, certain portions can be left empty, that is, free space can be distributed throughout the file. When a record is inserted or when an existing record is lengthened, the free space at or near the existing records closest in key sequence is used. This eliminates the need for overflow chains and overflow areas; it also minimizes data movement. Thus performance does not degrade substantially as records are added and the file does not have to be reorganized as often as an ISAM file. VSAM reclaims space when a record is deleted or shortened, and the space released becomes free space.

The index of a key-sequenced VSAM file is more efficient than an ISAM index because it generally requires less direct-access space and less updating of index entries. Space is saved by eliminating redundant key information in the index entries (key compression) and by blocking index records. A shorter index requires less time to search and update. Updating is infrequent, however, because index entries are not usually modified when records are added to the file.

Several index options are available to speed VSAM processing:

- the index and the data can be on different devices.
- all or part of the index can be loaded into virtual storage if the user provides enough space for it.
- the lowest level of index records can be written adjacent to the data to reduce disk-arm movement, and several copies of an index record can be written on a track to reduce rotational delay.

entry-sequenced files

Records are stored in entry-sequenced files in the order in which they are submitted to or written on the device. No keys are recognized and, consequently, no indexes are built. The order of records is fixed; they are not moved. Thus, free space is not distributed throughout the file, and new records are placed at the end. Records can be shortened but they cannot be deleted. If a record is lengthened, a new copy of it is written at the end of the file. Since there is no index, the user must either access the file sequentially (in the order the records were written) or directly by record addresses. VSAM passes the user the address of each record as it is added to the file. He can maintain a table of those addresses, or supply a randomizing algorithm for addresses, to access the records.

data organization

The data organization of ISAM is based on the physical units of disk cylinder and disk track, while the data organization of VSAM is based on logical units called control intervals and control areas. A control interval is the unit of direct-access storage that is transferred to and from virtual storage. It can contain one or more records in one or more blocks. Each entry in the lowest index level of a key-sequenced VSAM file points to a control interval. Free space in a key-sequenced file is distributed in terms of control intervals. A percentage of each control interval can be free space and some control intervals can be entirely free space. Indexes are also organized in control intervals. Each contains a single index record which can have many index entries. A control area is a group of control intervals. The number of control intervals of data in a control area equals the number of index entries in each index record. VSAM data organization provides for device independence by reducing the programmer's concern about the physical characteristics of the data and the index. Figure 2.25 illustrates VSAM data organization.

data access

As with ISAM, VSAM records can be accessed either sequentially or directly. Key-sequenced files can be accessed by key or address, entry-sequenced files can be accessed by address only. The key can be that of an individual record or it can be a generic key (the front part of a key field) which specifies the keys of a group of individual records. The address is a relative address of the record from the beginning of the file, and is called the relative byte address.

VSAM allows retrieval, storage, update, and deletion of records. The access can be either sequential or direct and can be by record key or record address. With a key-sequenced file, several records in sequence can be inserted as a group at one point in the file. This is faster than inserting them one at a time with direct access, as ISAM requires. Also, the user can access several records in key-sequence and then have VSAM skip to another portion of the file and access more records in sequence without having to search the entire index to find the new group of records.

A single I/O macro, such as GET, can access one record or several records at one time. Also, more than one I/O macro can be issued to access records in different parts of the file at the same time.

VSAM catalog

VSAM keeps central control over the creation, access, and deletion of files and over the management of direct-access storage space allocated to those files. This is done by keeping information on file and space characteristics in one place, the VSAM catalog. The catalog, which is unique to VSAM, makes it easier to (1) keep track of both files and available direct-access space, (2) write job control statements to create and process VSAM files, and (3) move VSAM files to other DOS/VS systems or to OS/VS systems. Once the user has allocated a volume or portion of a volume to VSAM, each file he creates is automatically sub-allocated space on that volume by VSAM. There can be more than one VSAM catalog. However, only one catalog at a time can be connected to the system. Each catalog can keep track of VSAM files on many volumes; it is not necessary to mount a volume to determine whether or not it has space available for a VSAM file.

file and volume portability

A significant feature of VSAM files is that either the files alone, or the volumes containing them, can be moved from one DOS/VS system to another or to an OS/VS system. This is possible because VSAM data format and accessing techniques are identical under both DOS/VS and OS/VS. Also, the VSAM catalogs for the two systems are identical in format.

language support

VSAM files can be accessed through assembler language macro instructions or COBOL statements. Existing assembler language ISAM programs can access VSAM files by using the ISAM interface program which translates ISAM requests into comparable VSAM requests. New and existing programs, written in ANS COBOL, PL/I, and RPG II to use ISAM also process VSAM files through the ISAM interface program.

service programs

VSAM has an extensive service program package, called access method services, which can be used to:

- Define (create), print, copy, or reorganize VSAM files.
- Add, alter, delete, or print catalog entries.
- Convert ISAM and SAM files to VSAM files.
- Copy a VSAM file and catalog entries in a format required for transporting to another DOS/VS system or to an OS/VS system.

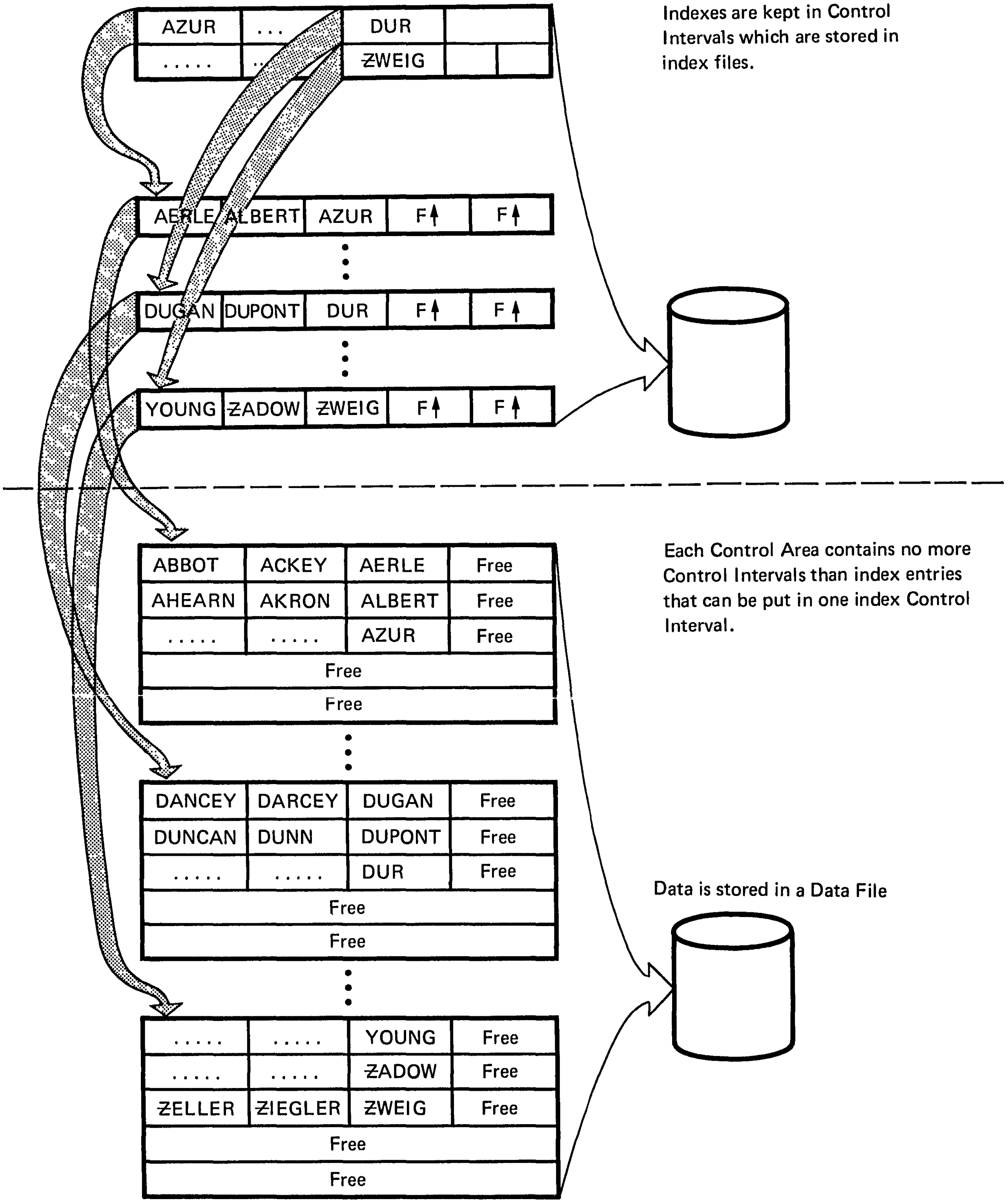
file sharing

VSAM uses the DOS/VS track hold feature to allow files to be shared across partitions. When a record is updated, the track containing the record is protected under exclusive control. The remainder of the file can be retrieved by programs running in other partitions.

Direct Access Method (DAM) and Organization

DAM offers the user a third possibility for making efficient use of the random access capabilities of disk storage. Whereas both VSAM and ISAM are comprehensive file management systems, offering both sequential and random retrieval capability, DAM is more specialized. DAM concentrates on random retrieval requirements only and provides the user with an access method that can accomplish this function efficiently. It does this by requiring the user to establish a direct relationship between the keys of the records and their physical addresses on the disk. This means that the programmer, by using the key of a record, can calculate or look up in a table the corresponding record address, and either directly store the record (on output) or directly retrieve it (on input). Greater programming burden and responsibility is placed on the user; the benefit is a potentially faster record retrieval time for specialized applications such as reservation systems, securities price and transaction inquiry programs, or selective retrieval from large tabular arrays.

A representation in DAM format of the personnel file used in previous examples might appear as shown in Figure 2.26.



Indexes are kept in Control Intervals which are stored in index files.

Each Control Area contains no more Control Intervals than index entries that can be put in one index Control Interval.

Data is stored in a Data File

Note: The length of the key and the percentage of free space to be distributed is determined by the user. The control area and the physical mapping are automatically determined by the system.

Figure 2.25. VSAM Data Organization

In the key-sequenced file shown, the records may be processed sequentially or directly. Records in a key-sequenced file can be accessed either by their key or by their address. The index file and the data file can be on different devices.

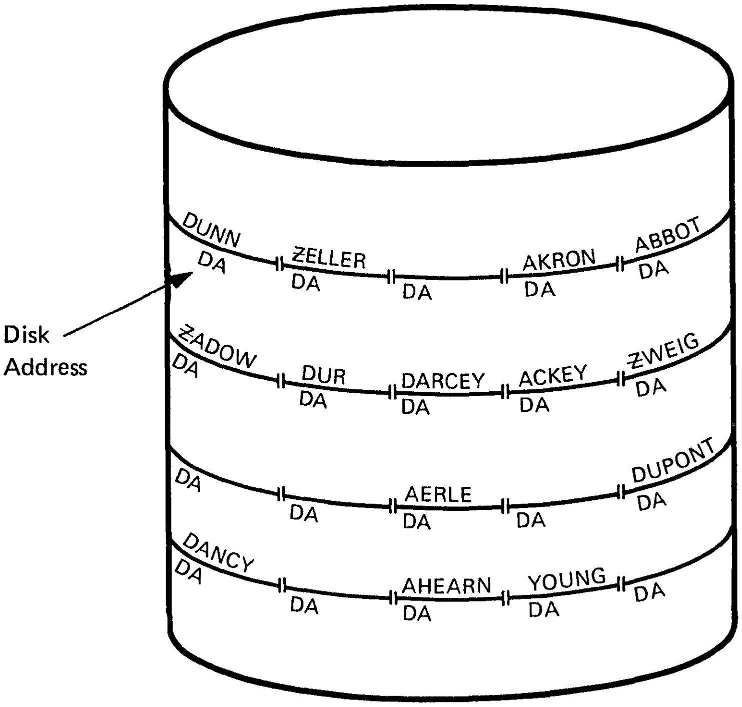


Figure 2.26. Direct Access Method Data Organization

Direct file organization implies that, for purposes of organization and retrieval, there is a direct relationship between the content of the records and their addresses on disk storage (DA in diagram).

Note that:

- Records are not in logical sequence by key.
- Tables to accomplish retrieval functions are **not** built and maintained by the access method.
- The physical location on the disk at which each record is stored and from which it is retrieved is determined by the **programmer**.
- Depending on the addressing technique used by the programmer, "gaps" may be left in the file. Also, the addressing technique could produce "synonyms," which are multiple records for the same address. The programmer is responsible for solving these problems.

Summary of Retrieval Methods for Disk

Figure 2.27 summarizes the types of retrieval available with each of the access methods supported for direct access devices.

Teleprocessing Access Methods

In teleprocessing, data processed by the computer system is obtained from other locations. Input and output of data to and from the computer is performed using terminals, comparable to I/O devices, which are connected to the CPU through communications lines.

Data Organization Method Type of Access Provided	Sequential Organization	Indexed Sequential Organization	VSAM	Direct - (or RANDOM) Organization
Sequential Retrieval/ Storage	Provided	Provided	Provided	Not provided directly; available only when implementation is programmed by user
Direct Retrieval/ Storage	Not Provided (Exception: SORT, Using a sequential disk file as input, can provide output consisting of record addresses.)	Provided	Provided	Provided

Figure 2.27. Summary of Retrieval Methods

The specialized routines provided by DOS/VS to support message transmission are contained in the Basic (BTAM) and Queued (QTAM) Teleprocessing Access Methods. Use is made of these facilities through assembler language macros. BTAM provides broad device support and wide functional flexibility. QTAM, which must run in real mode, is functionally more restrictive but relieves the programmer of many activities with which a BTAM user must be concerned. QTAM generally requires two or more tasks that may run in one or more partitions, one for a message control program that interfaces directly with the communications lines, and one or more for message processing programs that act upon information received from remote stations, and that may generate messages for those stations. Data can be processed as it arrives at the central computer, or be stored in intermediate storage, such as a disk volume, for later processing in batches.

There are five ways in which teleprocessing may be used in DOS/VS:

- Message switching: messages from one remote station are routed to another through the CPU.
- Message processing: messages received from remote terminals are processed by an application program running in the CPU.
- Remote job entry: entire jobs are submitted to the central CPU from remote terminals.

- Data collection: the CPU collects information presented by the various remote terminals for later analysis and processing.
- Inquiry/response: remote stations request information from a central source of data.

applications

Some factors that would indicate the applicability of a teleprocessing system are:

- Customer convenience. Brokerage firms, for example, require rapid execution and confirmation of customer orders.
- Inventory control. Manufacturing applications are common, but there are other "inventories" - such as airline space availability - that can be effectively controlled by a TP system.
- Credit Control. Central data files can provide assurance that a customer has funds or credit approval.
- Management control. Immediate access to centralized data files provides more timely information for control of business operations.
- Industrial control. Computer control of key production factors increases productivity of capital equipment (for example, in petroleum refineries).
- Equipment centralization. In collecting data from remote sources, either intermittently (as in production data collection) or periodically (as in central summarizing of statistical data from distant branch offices), one CPU may do the processing that would otherwise require a separate system at each remote site.
- Innovation. Some applications are just not possible without a TP system (for example, online debugging, text editing, and computer-assisted instruction).

Logical and Physical IOCS

The access methods provided by DOS/VS are designed to meet the file organization and access needs of the large majority of the system users. The data management facilities contained in these access methods are collectively called LIOCS (Logical Input/Output Control System). LIOCS functions, expressed by the user in the format of macros or high level language statements, are translated by the access methods to a physical level prior to the actual execution of machine functions. DOS/VS also allows the programmer to write at this physical level if he wants to. Data management performed in this manner is called PIOCS (Physical IOCS). PIOCS provides the user with a degree of flexibility in handling I/O operations greater than that provided by LIOCS, but at the same time, requires a greater understanding of and responsibility for the detailed aspects of input/output operations. PIOCS could be used to write programs for an I/O device not supported by DOS/VS, or for some unusual data organization or retrieval/storage requirement that is not met by the standard DOS/VS access methods.

High-Level Language Support for Data Management Functions

In addition to the assembler, IBM provides compilers in the form of program products for the following languages:

- FORTRAN
- DOS/VS COBOL, Full ANS COBOL, and Subset ANS COBOL
- PL/I
- RPG II.

Each language has data management facilities based on those provided by the standard access methods. These are summarized in Figure 2.28. The I/O statements in these languages are not macros as in assembler language, but are statements in the format prescribed by the language.

Access Method Language	SAM	ISAM	VSAM	DAM		BTAM QTAM
				Using Key and Track Reference	Using Record ID and Track Reference	
DOS/VS COBOL	YES	YES	YES	YES	YES	NO
Full ANS COBOL	YES	YES	YES	YES	YES	NO
Subset ANS COBOL	YES	YES	Through ISAM Interface Program Only	YES	YES	NO
FORTRAN	YES	NO	NO	NO	YES[1]	NO
PL/I	YES	YES	Through ISAM Interface Program Only	YES	YES[1]	NO
RPG II	YES	YES	Through ISAM Interface Program Only	YES	YES	NO
Assembler (SCP)	YES	YES	YES	YES	YES	YES

[1] The direct access support is implemented by having the user specify the relative record number within the file.

Figure 2.28. Language Support for Data Management Functions

The assembler is included in this table for comparison purposes.

Data Security and Data Integrity

In any data processing installation, it is important to protect data and files against accidental or intentional misuse or destruction. In multiprogramming, this protection is even more important, since programs in different partitions may be attempting to retrieve and update the same file or even the same record simultaneously. The user must be aware of this possibility and exercise care to safeguard the validity of the data.

In order to provide the necessary security, DOS/VS has two kinds of protection:

1. Functions that are either standard or that can be user-specified:
   - File labeling
   - Protection against duplicate assignment
   - DASD file protection
   - Track hold function
   - Data file security.
2. Resource protection macros that enable the user to provide his own protection facilities in a multitasking environment.

File Labeling

File protection is achieved by file labels. File labels are records associated with the **files** stored on tape, disk, or diskette. These labels provide unique identification for the files. In addition, tape, disk, and diskette **volumes** can be identified by labels.

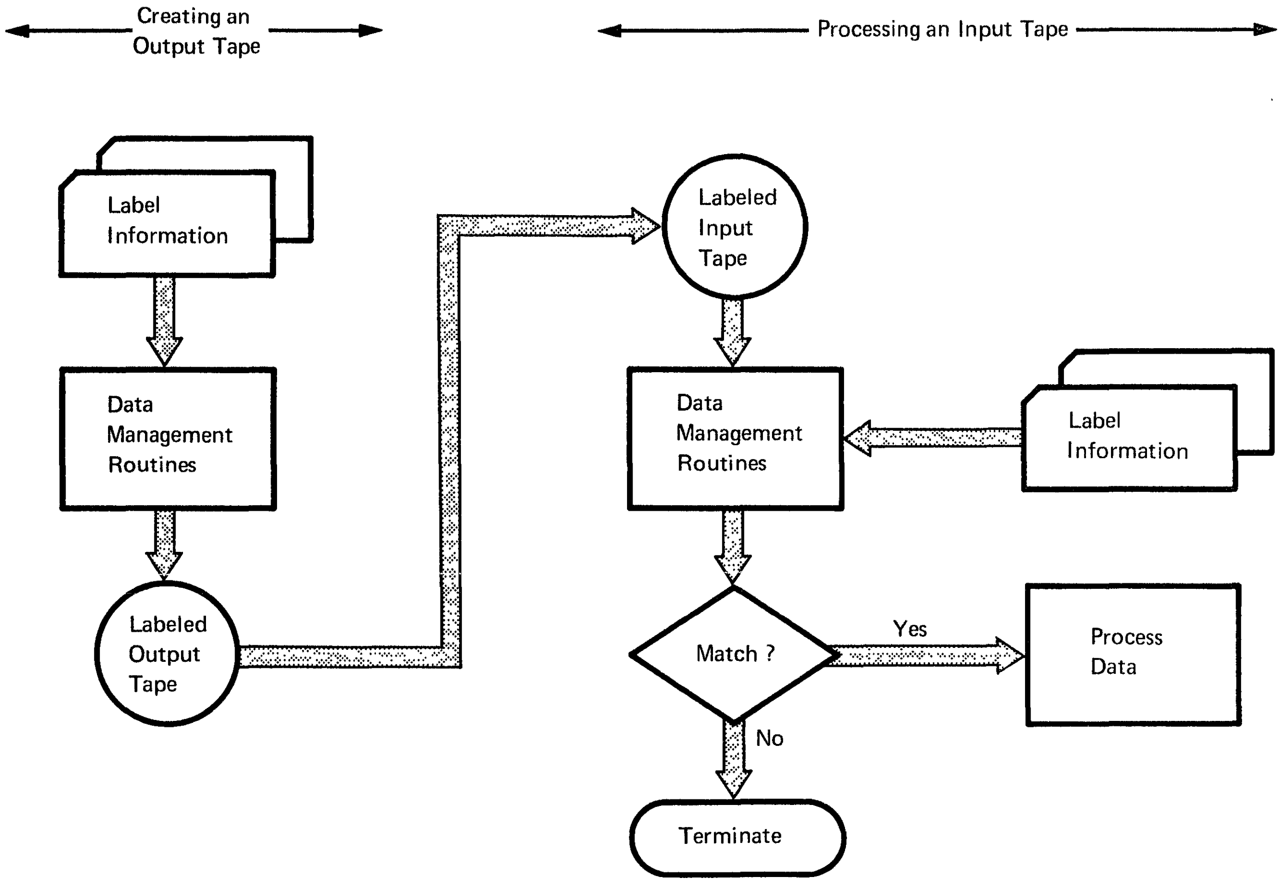


Figure 2.29. Label Processing

On output, the data management routines create labels from information specified by the user. Prior to any subsequent processing of the tape, the data management routines verify the labels against information supplied in the job control cards.

For tape storage, all labels are optional; for disk or diskette storage, each volume must have one volume label, and each file must have a file label. Additional labels can be created and processed by the user on tape or disk storage.

Whenever a file is created, the user can specify the contents of the labels. Then, when the file is processed as input (see Figure 2.29), the user must specify, in job control statements, the contents of the label, so that the data management routines can compare the specified data with the actual label. If data management detects a mismatch between the actual label and the label information, the job is terminated.

This checking function is useful only if:

- All output files are created with labels.
- All labels are unique.
- Label checking is always performed during input.

Complete information on the way DOS/VS handles tape, disk, and diskette labels is provided in the manuals *DOS/VS Tape Labels* and *DOS/VS DASD Labels.*

Protection Against Duplicate Assignments

In a multiprogramming environment, it is essential that no two programs in different partitions gain access to the same device, except for disk units. A single disk unit may contain a number of files; therefore disk units can contain files for more than one partition; as a result, the DOS/VS provides the following protection against duplicate assignments:

- Unit-record devices and tape drives assigned to a program in one partition cannot be assigned to a program in another partition.
- A tape to be used for SYSOUT data cannot be used for any other system or programmer logical unit. If this is attempted, a message is issued to the operator, who can then take corrective action.
- If additional protection for a tape unit is required, that tape should be assigned with the volume serial number specified. The system then bypasses unassigned tapes that do not contain the requested volume label. If the system cannot find a tape for the requested volume serial number, it must be mounted by the operator.
- If a disk should not be shared between partitions, it should be assigned without specifying the share option (SHR) in the ASSGN statement/command.

DASD File Protection

This is a system generation option that, in case of a programming error, prevents programs from writing outside the extents specified for their files. The other files on the DASD device are thus protected against accidental destruction.

Track Hold Function

This is a system generation option that prevents two or more different programs from accessing the same track on a DASD device concurrently. All programs that make use of this feature should specify the function. The track is then held for the first program that accesses it and is relinquished when processing is completed. Track hold is required for VSAM and the ISAM interface program.

Data File Security

Upon creation of a DASD output file, it can be specified by means of a job control statement that the file is to be secured. A secured diskette file, however, can be created by specifying a DTF parameter. Creating a secured file means:

- The operator is notified with a message when a secured file is being used as input. He can then make the decision as to whether the file should be processed.
- The label cylinder display program does not display the label information of any secured files.

Access to VSAM data files is protected by four levels of passwords. In addition, VSAM provides an exit for users to impose their own routines.

Resource Protection Macros

In a multitasking environment, there is essentially nothing to prevent a task from using the resources of another task in the same partition. These resources can be I/O devices, files, data areas, routines, real storage, and the like. When, however, all tasks in a partition use resource protection macros to protect shared resources, concurrent use is prevented.

For example, a task can gain exclusive control of a resource by issuing an ENQ macro, and relinquish control after use by issuing a DEQ macro. Any other taks which also does this for the same resource is placed in the wait state until the first task has completed its use.

I/O Devices Supported

See "Part 5. Configurations" for a complete survey of the I/O devices supported by DOS/VS.

System - Operator Interaction

Operator-system communication plays a vital part in the efficient operation of a data processing installation. DOS/VS provides facilities that ensure timely and effective interaction between man and machine.

operator's duties

The operator's primary duties in a computer installation are:

- Start up the system.
- Prepare the I/O devices. For example, mount tapes and disk packs for each individual job.
- Initiate and control the execution of jobs.
- Interpret and respond to the system's or programs' requests for information or action.

More specifically, the operator of a DOS-controlled computing system can:

- Initiate and control multiprogramming operation.
- Interrogate the system to obtain information about its status.
- Cancel the execution of a job, for instance, in the case of an unending program loop.
- Diagnose problems with the help of problem determination aids.

system action

For its part, the system:

- Alerts the operator when a condition requiring his intervention occurs.
- Provides information such as start and stop times of jobs and run times.

system-operator communication

Communication from the system to the operator is in the form of messages written on the operator communications device, which may be a keyboard console, or, for Models 115 and 125, the screen of the display operator console. The messages describe the situation that requires operator action. Each message is identified by a unique number. This number serves as an entry point into the *DOS/VS Messages* manual, where an explanation is given for each message, along with a description of the action required.

Operator responses to messages are generally short, one-word answers, such as RETRY, IGNORE, or CANCEL. Alternatively, the operator can allow the system default option to take effect in most situations requiring intervention.

Communication from the operator to the system is in the form of commands and replies to messages. There are three types of commands:

- Job control commands: almost identical in format and scope to the job control statements.
- Operator commands: statements for performing controlling duties (for example, changing the size of partitions), or for asking for specific information (for example, the assignment of I/O devices).
- Screen control commands: statements for manipulating the information displayed on the display operator console of Models 115 and 125.

The operator can interrupt processing at any time by pressing the REQUEST key on the console keyboard. He can then type in commands and signal the system to resume processing. In addition, he can instruct the system to suspend processing after the current job or job step in any partition, for example, in order to allow time for changing printer forms or device assignments.

In addition to operator-system communications, there can be direct operator-problem program communication, provided that the problem program has a special operator communications routine. The operator can then request direct communication with this routine by pressing the external interrupt key on the console (for background programs), or by issuing a command (for foreground programs).

This could be useful, for example, for inquiry to an inventory file at unpredictable times. The inquiry routine could be included in any program that always occupies a partition and be invoked through the operator's communications routine when desired.

Reliability, Availability, and Serviceability

Reliability, Availability, and Serviceability (RAS) facilities are designed to maintain a high degree of trouble-free performance of the System/370. Debugging procedures are provided to help the user to choose the debugging aid best suited to obtain information about a particular type of error or malfunction.

The facilities that make up RAS are:

- **Debugging aids** that provide the user with information about his system at the time of a program, operator, or hardware error.
- **Recovery Management Support Recorder** (RMSR) that, depending on the severity of the failure, enables the system to recover from an error condition caused by any of the following hardware failures: real storage errors, central processing unit instruction errors, input/output channel errors, control unit errors, and device errors. It also records hardware failures on the system recorder file (SYSREC) and prints messages on SYSLOG that keeps the operator informed about the condition of the system hardware.
- **Environment Recording, Edit and Print Program** (EREP) that allows the user to print the contents of the system recorder file, pre-formatted, on SYSLST. EREP has many options that provide the user and his IBM Customer Engineer with details about the state of his system hardware.
- **OnLine Test Executive Program** (OLTEP), together with a set of device test programs, constitutes the online test system. Its functions include the diagnosing of I/O errors, the verification of I/O device repairs and engineering changes, and the checking of I/O devices. OLTEP is an optional program. If it is included in the system, it must be executed in real mode in the background partition.
- **Reliability Data Extractor** (RDE) that is an option that, if specified during system generation, enables the user and his IBM Customer Engineer to record the reason for an IPL procedure as well as the number of IPL procedures carried out on a system over any specified time period, for example, during an operator's shift. RDE also enables a time stamp. End of Day (EOD), to be recorded on SYSREC during system operation.

Integrated Emulators

An emulator is a combination of programming and special machine features that permits a computing system to execute programs written for another kind of system. An emulator, for instance, can enable programs written for System/360 Model 20 to run under DOS/VS on an IBM System/370 machine. However, emulators should be used only during the short period of installation of a new system. In order to make full use of the faster processing and I/O device speeds of the new system, applications should be reprogrammed.

Integrated emulation offered with DOS/VS allows the user to emulate a number of non-DOS/VS programs on the System/370 concurrently with the execution of normal DOS/VS programs.

A program running under an integrated emulator can be executed in any partition of a multiprogramming configuration without affecting processing in the other partitions.

Figure 2.30 indicates the machine configurations that can be emulated on System/370 under DOS/VS.

Emulation Model	Emulation of 1401/1440/1460	Emulation of 1410/7010/	Emulation of System/360 Model 20
Model 155 - II and 158	yes	yes	no
Model 145	yes	yes	no
Model 135	yes	no	yes
Model 125	yes	no	yes
Model 115	yes	no	yes

Figure 2.30. Emulation on System/370 Under DOS/VS

The user should bear in mind the following general considerations:

- The size of the emulator program depends on the system to be emulated.
- DOS/VS emulators allow device independence for unit-record devices used by the emulated programs.

1401/1440/1460 and 1410/7010 Emulator Considerations:

- Disk files must be converted before they are used by the emulator program, unless the CS30 or CS40 compatibility options have been used.
- Tape programs can be in original or converted format. DOS spanned variable record (VRE) format is set as standard for the System/370 emulators.

System/360 Model 20 Emulator Considerations:

- Mixed parity/density on 7-track tapes is not supported.
- Load UCS is performed by a DOS/VS utility, not a Model 20 utility program.

System Generation

System generation is the creation of an installation-tailored operating system based on the general DOS/VS system distributed by IBM. System generation procedures include:

- Generating a supervisor adapted to the installation and its application needs.
- Assembling or compiling, and link-editing user and IBM programs as necessary, and cataloging them in the appropriate libraries, and, if possible and required, in the SVA.
- Deleting unnecessary components to free additional disk space.
- Editing, formatting, or deleting libraries as required.

The system that is shipped by IBM to the user is capable of immediate operation. Most users, however, must generate supervisors that are adjusted to meet the configuration of their installation to ensure optimum efficiency.

The system core image library, the shared virtual area, the relocatable library, the source statement library, and the procedure library may also be edited. The private libraries may be created according to the user's needs.

Briefly, the system generation process is as follows: the user, with the assistance of FE, codes a set of supervisor macro instructions describing his system configuration and functional options. These macros are assembled, resulting in the production of a new supervisor, which is then cataloged into the system core image library. Certain modules, such as IOCS modules, may be assembled from the macro definitions in the source statement library and added to the relocatable library. Finally, the user may link-edit certain system service programs and catalog these into the core image library and in the shared virtual area as well. The result of these operations is the user's operational system, to which IBM Program Products can be added.

Planning System Generation

Detailed planning for system generation saves the user unnecessary repetition during the procedure. Essentially, planning for system generation consists of:

- Planning the options and estimating the size of the supervisor. This entails selecting from the programming services provided by IBM those options to be included in the supervisor, and estimating the cost of these services in terms of bytes of real storage. The supervisor program is composed of assembled supervisor generation macros. These macro instructions describe the machine configuration, the standard I/O assignments, and standard processing options. The size of the assembled supervisor depends on the options selected in each of the supervisor generation macros.

- Planning the contents, organization, and ultimate size of the system and private libraries, and of the shared virtual area. This entails distributing the storage space available on the disk packs among the libraries needed for daily use. The user must decide on the size and number of libraries and on the size of the shared virtual area and the system GETVIS area contained in it when planning an operational system. He must take into consideration the number of disk drives available, the number and size of the programs, which programs he wants resident in the core image library and which programs he wants in the shared virtual area, the impact that the expansion of one library has on the other libraries on the same disk pack, and the plans for future space requirements.

Detailed information for planning system generation may be found in the *DOS/VS System Management Guide*. Implementation procedures will be documented in *DOS/VS System Generation*.

Shipment of DOS/VS

DOS/VS is shipped in the form of a system core image library, a system relocatable library, a system source statement library, and a system procedure library. The contents of these libraries are identified in *Attachment 1* of the *Memorandum to Users* that accompanies the shipment. The system core image library must be retained on the operational volume, because it contains the DOS/VS programs in executable format. The other libraries may be either retained or deleted.

- The core image library contains IBM programs that are link-edited and ready for execution. System control programs and system service programs are always shipped in the core image library, and so the system is immediately operational upon arrival at the user's installation. If the user wants to increase system throughput, he should set up a shared virtual area with a system directory list and place programs that are reenterable and relocatable in that area, provided he intends to use those programs often. Placing programs and program phases in the shared virtual area saves the time required for loading the phases at execution time. The core image library shipped to the user also contains an assembler program.

- The relocatable library contains IBM programs that have not been link-edited, plus data management modules. The programs include all of the IBM-supplied components, such as job control, linkage editor, and librarian. The data management modules perform input and output procedures for the IBM-supplied programs, and can also be used by the user's problem programs.

- The source statement library contains IBM-supplied macro definitions. The user may specify parameters to assemble the macros that he requires. Thus, he can generate a new supervisor from the supervisor generation macro definitions. He can also assemble the POWER modules (if he wants to operate under POWER), and assemble IOCS modules from the IOCS macro definitions. For the user's convenience, the source statement library also contains sample problems and system generation job streams that can be retrieved as needed.

- The procedure library contains a procedure that will help improve system performance. This procedure is a list of the names of IBM-supplied phases, which are to be loaded into the SDL (system directory list) in the SVA (shared virtual area).

IBM supplies the Disk Operating System on either magnetic tapes or disk packs. If DOS/VS is shipped on tape, it must be copied at the installation onto disk before the system generation procedure can begin. Program Products must be ordered separately.

Part 3. What's New in DOS/VS?

This section is primarily intended for readers who are already familiar with versions of DOS earlier than Release 28. New items in Release 28 and 29 are covered here. For ease of identification, the items new in Release 29 are marked with an asterisk.

The capabilities that DOS provides to users have continually expanded over the years. In keeping with this, DOS/VS further extends support of the System/370 line of computers and peripheral equipment. DOS/VS features also expand the programming facilities available to the user. Among the highlights:

virtual storage support

With DOS/VS, the new storage concept of the System/370 - virtual storage support - is made available to the user. DOS virtual storage support provides the user with improved processor, or real, storage management and reduces the programming constraints imposed by the limited size of this storage.

relocating loader

A relocating loader feature has been added to the system. It allows the linkage editor to create program phases that are relocatable. It also causes the relocating loader to be generated in the supervisor, which resolves the load address to any location specified by the user at execution time and updates all of the entry points and address constants in the relocatable phase. The user need retain only a single copy ot his program in a core image library for execution in any partition. Programs that are link-edited to a beginning address at the end of the supervisor, for example, are now no longer influenced by increases in supervisor size, if they are processed by the relocating loader.

new assembler

The DOS assembler D has been replaced by the DOS/VS assembler. All IBM macros are shipped in edited format. User-written macros can also be converted into pre-processed format by the assembler using the new EDECK option; or they can be retained and assembled with a COPY statement.

VSAM

VSAM is a new, easy-to-use DASD access method with advantages over existing IBM access methods. It combines an increased and extendible scope of processing functions with high performance and high reliability and data integrity. These properties of VSAM stem from (1) the introduction of a new data and free-space organization, (2) a basically new access method design, and (3) the application of new data processing technologies.

***shared virtual area (SVA)**

In a multiprogramming system, the shared virtual area is located in the high end of virtual storage. Phases cataloged in the system core image library that are relocatable and reenterable are eligible to reside in the SVA. Such phases can be shared concurrently by programs running in any partition. The system directory list (SDL) can also be contained in the SVA. The SDL contains a list of pointers to frequently-used phases. Usage of the SDL can improve performance.

***extended I/O device assignment**

The ASSGN statement/command has been extended to provide for a greater flexibility and ease of use in making symbolic assignments.

***supervisor selection**

In some installations it is convenient to have more than one supervisor, for example, one generated with teleprocessing support and another without such support. Multiple supervisors can be present on the same system residence file if they each have a different name. A new function gives the user the opportunity to specify at IPL time which supervisor he wants to use.

procedure library

To the existing three system libraries a fourth, the procedure library, has been added. The procedure library enables the user to catalog frequently used sets (procedures) of job control and linkage editor statements and to include the cataloged procedures in the job input stream with a job control statement.

integrated emulator

A new integrated emulator allows programs written for the System/360 Model 20 to run under DOS/VS on the System/370 Models 115 and 125. The 1401/1440/1460 emulator has been extended to run on the System/370 Models 115 and 125.

POWER program

POWER, which provides automatic spooling support for unit record input/output, is now a part of the DOS/VS SCP (System Control Programming). Users having jobs that are predominantly unit-record bound can benefit from performance gains due to the more efficient use of unit-record I/O devices and CPU time. POWER allows multiprogramming in up to four partitions, using only one set of unit-record I/O devices.

additional foreground partitions

DOS/VS supports one background and four foreground partitions. The BJF facilities in DOS have become standard. The single program initiator (SPI) program is no longer supported.

system residence devices

The direct access devices that may be used as system residence device are listed in Figure 5.5.

Processing Flexibility

DOS/VS requires a minimum supervisor of 26K bytes for Models 115 and 125, 28K bytes for Models 135 and 145, and 30K bytes for Models 155-II and 158 of System/370.

DOS/VS allows the user with a minimum configuration more flexibility in allocating his storage resources to obtain maximum job throughput. Multiprogramming in up to five partitions and virtual storage support allow the user to optimize CPU usage and storage space.

New processing features available under DOS/VS include:

- Virtual storage support
- The relocating loader feature
- Five-partition support
- Variable partition priority
- The shared virtual area (SVA)
- Extended I/O device assignment
- Supervisor selection
- The procedure library
- A new timing facility
- A new DOS/VS assembler
- System/360 Model 20 emulation on Models 115 and 125
- 1401/1440/1460 emulation on Models 115 and 125.

Virtual Storage Support

The rapid growth in the number and types of data processing applications has led to an increasing demand for freedom in designing applications without being functionally constrained by the physical characteristics -- system architecture, I/O device types, and CPU space -- of a particular computer system.

While System/360 with DOS already allowed the programmer a certain degree of device independence, the need for making programs fit into the available real storage still existed. This often required overlay techniques to make the program fit into a partition. Structuring these overlays added to the complexity of solving a problem. With the increased use of high-level languages, multiprogramming, expanded system control programs, and applications that require relatively larger amounts of real storage (teleprocessing, data base, etc.), the need for more real storage space and a more dynamic use of it is still growing.

To meet this need, the System/370 models provide significantly more real storage capacity than the comparable System/360 models. The availability of more storage though, did not relieve all the constraints associated with this storage. It did not eliminate the waste of storage resources through, for example, dormant code, as might be the case with an inactive or low activity teleprocessing network, or through storage fragmentation as a result of programs running in bigger partitions than required for their execution. The system had no means of dynamically utilizing the fragments of free storage space. Consider also the following situations:

1. An application is designed to operate in a 50K real storage area, which is adequate to handle current processing needs and provides room for some expansion. Some time after the application is installed, maintenance changes and the addition of new function causes one of the programs in the application to require 51K and another to require 52K. Installation of the next real storage increment cannot be justified on the basis of these two programs, so time must be spent restructuring the programs to fit within 50K.

2. An existing application has programs with an overlay structure. The volume of transactions processed by these programs has doubled. Additional processor storage is installed. However, the overlay programs cannot automatically use the additional storage. Therefore, reworking of the overlay structure programs is required to make them non-overlay and, thereby, achieve the better performance desired.

3. A simple, low-volume, terminal-oriented inquiry program that will operate for three hours a day is to be installed. If the program is written without any overlay structure, it will require 60K of real storage to handle all the various types of inquiries. However, because of a low inquiry rate, only 8K to 12K of the total program is active at any given time. The inquiry program is designed to operate in 12K with a dynamic overlay structure in order to justify its operational cost.

4. A series of new applications are to be installed that require additional compute speed and twice the amount of real storage available on the existing system. The new application programs have been designed and are being tested on the currently installed system until the new one is delivered. However, because many of the new application programs have storage design points that are larger than those of existing applications, testing has to be limited to those times when the required amount of real storage can be made available. Although another smaller scale model is also installed that has time available for program testing, it cannot be used because it does not have the amount of real storage required by the new application programs.

5. A large terminal-oriented application is to be operative during one entire shift. During times of peak activity, four times more real storage is required than during low activity periods. Peak activity is experienced about 20 percent of the time and low activity about 40 percent. The rest of the time activity ranges from low to peak. Allocation of the peak activity storage requirement for the entire shift cannot be justified and a smaller design point is chosen. As a result, a dynamic program structure must be used, certain desired functions are not included in the program, and response during peak and near peak activity periods is affected.

In the situations described, real storage is the constraining factor. However, even if more real storage were added to a system as needed, the system could not automatically make use of it. Applications would still have to be redesigned, and the waste of storage through fragmentation and dormant code would still exist.

To assist in solving these problems, the new System/370 virtual storage concept offers a means of dynamically and automatically using real storage resources, storage fragments as well as storage space added to the system at later times. With virtual storage support, programs are no longer restricted to the address space available to their partition in real storage. They may exceed this limit to a certain extent and still get the necessary real storage as it is needed for the execution of each section of the program.

The time required for any program to execute under any operating system has always been and still is dependent on such factors as the mix of programs executing concurrently, their relative priorities, system and application file placement, and in some cases on the particular data being processed. Under DOS/VS, program performance is also highly dependent on such factors as the amount of real storage overcommitment, the storage reference patterns of the program, and the speed of the paging device. The performance of each program must be evaluated in the light of at least these factors. For online or real time systems with specific performance or response requirements, particular attention must be given to assuring that adequate resources (real storage, CPU time, channels, disk arms, etc.) are available. In some cases it may be necessary to test the program using the specific user workload and configuration to verify what system resources are necessary to give adequate performance.

How virtual storage works and how the system makes use of all real storage space available is shown in the second part of this book under the heading "Virtual Storage Support".

Implementation

The user must provide accommodation for virtual storage support at a number of points during his preparation and use of the system. Details of implementation are not provided in this manual. They are found in the *DOS/VS System Management Guide*.

SYSGEN - Virtual storage support is a standard function in DOS/VS. At system generation time, the size of the real and the virtual address areas must be specified in the new VSTAB supervisor macro. The statement defining system default values for partition allocation has been extended to provide for both virtual and real partitions.

The page data set required for virtual storage support may be defined during system generation or IPL. It is initialized and preformatted during IPL. (However, it need not be reformatted at every IPL.)

Relocating Loader

The relocating loader allows all users, including those who program in high-level languages, to load single-phase and multi-phase programs at any valid problem-program address in the system. Under this option, the linkage editor is able to catalog relocatable phases into the core image library, and the relocating loader in the supervisor assigns the absolute machine addresses that are necessary for program execution. This means that the user need retain only one copy of the program in the core image library.

In previous versions of DOS, the user could vary a program's load address only by coding the program as self-relocating, by storing multiple copies in the core image libraries, or by relink-editing before execution.

Consequently, the relocating loader saves programmer time, reduces core image library storage space, and minimizes the number of linkage editor runs.

Implementation

The relocating loader is a DOS/VS option that is included in the system if RELLDR=YES is specified in the FOPT system generation macro. The following points must be considered:

- A relocatable phase differs from a non-relocatable phase only in that it has relocation information appended to it. Usually, this does not increase the library space used by the phase; in some cases, however, one or more additional library blocks may be necessary.
- If RELLDR=YES has been specified, the user may suppress this option for a particular program by specifying NOREL in the linkage editor ACTION statement.
- If the supervisor has been generated with RELLDR=NO (default value), the user can link-edit a particular phase as relocatable by specifying REL in the linkage editor ACTION statement. In a system without the relocating loader feature, a relocatable phase can only be loaded into the virtual partition for which it was originally link-edited (that is, the relocation factor is not applied).
- Regardless of relocating loader support therefore, the user can specify at link-edit time that a program is self-relocating, that it has an absolute load address, or that it be relocatable.

Additional Foreground Partitions

Users of DOS/VS can multiprogram in one background and four foreground partitions (BG, F4, F3, F2, F1). Details on multiprogramming in the virtual storage environment are found in the second part of this book under the heading "Virtual Storage Support".

Implementation

The new NPARTS parameter in the SUPVR macro is used during system generation to indicate, for a multiprogramming system, the number of partitions required. From two to five partitions can be specified. Each specified partition requires one DASD track for its standard label information, and one track for its user label information.

Variable Partition Priority

The user can modify the sequence of partition priorities. After setting the priorities during supervisor generation, he can modify them later by using an operator command.

Implementation

The user, through the PRTY parameter in the FOPT macro, specifies the relative priorities of partitions during system generation. During subsequent operation, the operator can request the system to print or change the current priorities by issuing a PRTY command.

The Shared Virtual Area (SVA)

The shared virtual area (SVA) is a new area in the high end of virtual storage in which the user can place phases that are cataloged in the core image library. For a phase to be placed in the SVA, it must be reenterable and relocatable. VSAM phases, for example, can be located in the SVA. The SVA contains a system directory list (SDL), which in turn contains the pointers to all phases in the SVA and pointers to a number of frequently-used phases not contained in the SVA (such as transients). The SVA can also contain a special area called the system GETVIS area.

The user can ask the system to place a phase in the SVA by adding the SVA parameter to the PHASE statement. The system then checks to see if the phase is SVA-eligible and listed as such in the SDL and, if so, places it in the SVA. All phases that are in the SVA are shareable between partitions.

The retrieval of phases from the SVA is faster than the retrieval of the same phases from the core image library.

Implementation

The user can specify the creation of a shared virtual area when he generates a multiprogramming supervisor for his system. To set the size of the SVA, he must specify the SVA parameter in the VSTAB macro. If no specification of the SVA is made, the system will contain an SVA of 64K bytes and a system GETVIS area of 0K bytes.

The size of the SVA can also be changed by the SET command immediately following an IPL.

Extended I/O Device Assignment

The ASSGN statement or command assigns a logical I/O unit to a physical device. Sometimes the user is not aware of the environment in which his job will run, or he may not be concerned with which particular physical device is to be used for his purposes. For these cases (among others) the following new functions are now available for making device assignments:

- Generic assignments
- Automatic volume recognition
- Clearer distinction between a permanent and a temporary assignment.

Implementation

In addition to being able to specify a physical address X'cuu', the user can specify the following parameters:

(1) A logical address (SYSyyy), which may be any system or programmer logical unit of the system

(2) A device class (READER, PRINTER, PUNCH, TAPE, DISK, or DISKETTE)

(3) A device type (for instance, 2400T9 or 3525RP)

(4) A list of physical unit addresses.

Two other new parameters of the ASSGN statement/command are:

- The VOL parameter, which provides for volume serial number recognition for tapes and disks. When this parameter is specified, the system searches for the requested volume and, if not found, issues a message to the operator.
- The PERM parameter, which is the opposite of the TEMP parameter.

Supervisor Selection

The user can have more than one supervisor on a single system residence file. At IPL time the user can specify which supervisor he wants to use.

Implementation

Before the IPL routines load the supervisor into real storage, the wait state is entered. By pressing the external interrupt key, the operator indicates that he wants to use the default supervisor. If he wants to use another supervisor, he can communicate its name via the console or a card reader. If the console is used, a message prompts the operator to type the supervisor name. If a card reader is used, a card with the supervisor name (punched in columns 1-8) is read.

The Procedure Library

The procedure library is a new system library that may be used to store - in card image format -

- Frequently used sets, procedures, of job control and linkage editor statements (basic support).
- Procedures additionally containing inline SYSIPT data, especially control statements for system utility and service programs (extended support). The inline SYSIPT data must be processed under control of the device-independent sequential IOCS or by IBM-supplied service programs and language translators.

The procedure library is part of SYSRES, so the maintenance and service functions available for the other DOS/VS libraries will also support the procedure library.

Cataloged procedures may be included in the job control input stream by a job control statement and temporarily modified by overwrite statements.

Implementation

The procedure library exists only as a system library, not as a private one. It may be generated during system generation.

The basic support is available in the DOS minimum system, while the extended support requires a supervisor with the SYSFIL option.

Interval Timer Support for Multiple Tasks

With multiprogramming, the interval timer support for multiple tasks allows for the concurrent satisfaction of timer requests of all tasks, main task as well as subtasks, in all partitions.

The timer is used by programs to schedule certain processing on a time interval basis, for example, to initiate the creation of checkpoint records every fifteen minutes, to poll a terminal every three seconds, or to change the traffic analysis algorithm every two hours.

Implementation

Through the IT parameter in the FOPT macro, the user specifies the inclusion of interval timer support during system generation. Details of implementation are not provided in this manual. They are found in the *DOS/VS System Management Guide.*

The DOS/VS Assembler

The DOS/VS assembler is a system control program that translates source programs written in System/370 assembler language into machine language. Its optimum real storage requirement is 20K, but if more storage is available, the assembler will use it to expand buffers and work areas.

The DOS/VS assembler replaces the Assembler D. In addition to supporting the System/370 instruction set, the new assembler is up to 30% faster than Assembler D. This improved performance is achieved through a new design which has been made possible by placing all library macro definitions on a separate sublibrary in edited format. (A library macro definition is a macro definition placed in a macro library and which can be called directly by a macro instruction. A source macro definition is a macro definition which is included in the source deck, either physically or by a COPY instruction.) All IBM-supplied macro definitions are delivered in edited format, and the user can use the assembler to produce his own edited macros for inclusion in the macro sublibrary.

There are three ways in which a user macro can be retained:

- A macro used only temporarily is normally maintained as part of the program, and is physically included in the source deck.
- A macro used in more than one program can be included as a separate book in the copy sublibrary and be maintained in this form. To call it, the programmer must first include a COPY statement at the beginning of his program to identify the macro as a source macro.
- A macro used more frequently should, after testing, be edited and kept in the macro library. Then it can be referenced directly by macro instructions.

A macro library used by previous assemblers can either be used as a copy sublibrary, or be converted to a macro sublibrary by letting the assembler edit all the macros and including them in a macro sublibrary.

Implementation

The assembler requires DASD space for three work files in addition to the standard DOS/VS requirements.

Edited macros residing in the macro sublibrary cannot be updated by single statements. The update is made to the original source code and, after editing by the assembler, the complete macro is replaced using the library maintenance program. If the source macro is not available, a de-editor program, supplied with the assembler, can be run to re-create the macro definition in source format from edited format.

Emulation on Models 115 and 125

A new integrated emulator allows programs written for the System/360 Model 20 to run under DOS/VS on a System/370 Model 115 or 125.

The 1401/1440/1460 emulator has been extended to run on a System/370 Model 115 or 125.

Implementation

There are several considerations that apply to the use of tape and disk files with the emulators:

- For the 1401/1440/1460 emulator, disk files must be converted by copying them to and restoring them from tape before they are used by the emulator. Tape files can be in original or converted format. DOS spanned variable record format is set as standard for the System/370 emulators.

- For the Model 20 emulator, tapes used by the Model 20 or by the Model 25 in Model 20 mode can be used by the emulator program without change, except that mixed parity or density on 7-track tapes is not supported. Disk volumes must be converted, using tape as intermediate output, to the format supported by the emulator.

I/O Flexibility

DOS/VS supports a continually expanding range of data management facilities, compilers and application programs available from IBM, and IBM peripheral equipment on the System/370 to allow the user a great deal of flexibility in tailoring data files, their processing, and I/O equipment to his specific needs.

New I/O features available under DOS/VS are:

- VSAM - a new access method.
- The POWER program.
- New I/O devices.

VSAM - a New Access Method

The Virtual Storage Access Method (VSAM) is a new data management facility which has improvements in data organization and access over other DOS/VS access methods. VSAM has key-sequenced files with an index. They are processed either sequentially or directly, like ISAM files. VSAM also has files arranged in the physical sequence in which the records are written on a direct access device. These files, called entry-sequenced, can be processed like sequential (SAM) or direct (DAM) files. An entry sequenced file does not have an index.

VSAM has the following improvements over ISAM:

- Allocation of space on the direct-access device is managed by the access method. The user simply sets aside a total area for VSAM which allocates whatever space is needed for each file when it is created from the total space VSAM controls. The access method maintains a catalog to keep track of the characteristics of and the space allocated to VSAM files and to help reduce the number of job control statements the user needs.

- An extensive service program package, called access method services, which can be used to:
  - Define (create), print, copy, or reorganize a VSAM file.
  - Convert an ISAM or SAM file to a VSAM file.
  - Add, alter, delete, or print entries in the VSAM catalog.
  - Create backup copies of VSAM files.
  - Copy a VSAM file and catalog entries in a format which makes it easily portable to another DOS/VS system or to an OS/VS system.

  Access method services functions are requested through job control statements, like utilities. A series of access method services commands can be executed in one job step, and commands can be modified depending on the results of previous commands.

- Free space can be distributed throughout a key-sequenced file when it is created for subsequent use when records are added. Also, free space is made available when records are deleted. Use of distributed free space replaces the chained record overflow of ISAM. Therefore, VSAM performance does not degrade significantly when records are inserted into a key-sequenced file, and those files do not have to be updated as often as ISAM files. Further performance improvement results because VSAM indexes are usually not modified when records are added.
- The security and integrity of data is improved for VSAM files. A file can be protected against unauthorized use by up to four levels of passwords, one of which the user must know and issue in order to access the file. The password levels grant authority to read a file, authority to read and update a file, or authority to read and update both a file and the VSAM catalog. Data integrity is improved by (1) minimizing data movement and index updating when records are added to a file, (2) preserving both new and old index paths to data until an update is completed, and (3) special formatting to indicate the end of a file as it is being created or extended.
- VSAM files contain variable-length as well as fixed-length records. Blocking and deblocking is done by the access method, which optimizes block length to suit the device on which the file is written. Also, VSAM's data organization allows a great deal of device independence; files can be moved easily from one type of device to another.
- When accessing key-sequenced files by record key, the key can be that of an individual record or a generic key (one or more rightmost bytes of the key field are omitted) that specifies a group of records. Several records in sequence can be inserted as a group at one point in a key-sequenced file. This is faster than inserting them one at a time with direct access as the user must do with ISAM. Also, VSAM can process sequentially, skipping to different parts of a key-sequenced file without searching the entire index.
- VSAM files can be moved to another DOS/VS system. This is made possible by the identical data organization and access techniques of DOS VSAM and OS VSAM and by the VSAM catalog, described above.

Implementation

Support of VSAM is provided through macros in the assembler language and DOS/VS COBOL. Most existing ISAM programs written in assembler language and ISAM programs written in PL/I, ANS COBOL, and RPG II can process statements into comparable VSAM statements. The interface routines are incorporated as a file is opened, so the program does not have to be compiled or link-edited again. However, since only the ISAM statements of a program are mapped into comparable VSAM statements, the full scope of VSAM functions cannot be obtained through the interface. VSAM files are created or converted from ISAM or SAM files by access method services.

VSAM uses DOS/VS virtual storage facilities to load the required access method modules and to assign storage for input/output buffers.

The POWER Program

POWER has been included in DOS/VS as a component of the system control programming. It supports spooling of unit record input/output for up to four partitions. All unit record input/output devices supported by DOS/VS are also supported by POWER. For a complete list of these devices, see the lists of punched card devices and printers in *Part 5. Configurations.* In addition, POWER supports the IBM 3540 Diskette Input/Output Unit as an input device.

POWER is designed to increase throughput by reducing the time the CPU waits for the completion of unit-record I/O operations. With POWER, the unit-record input and output for all jobs in all partitions is performed using a DASD as intermediate storage. A job's normal punched-card input (including job control statements) is read from the card reader and stored on disk by POWER in input queues before the start of the job. Similarly, a job's output is stored on disk in an output queue and printed and punched at a later stage.

Normally when multiprogramming with DOS/VS, separate unit-record devices must be assigned to each partition. A significant advantage of POWER is that this requirement for multiple card devices and printers has been removed by allowing a single input or output device to serve up to four partitions.

Remote Job Entry

POWER also offers the remote job entry (RJE) capability. This is a teleprocessing feature that allows jobs to be entered simultaneously into the system from up to five remote terminals.

Implementation

User programs do not need to be modified in order to run under POWER control. However, the DOS/VS supervisor must be generated with the POWER=YES operand. In addition, the POWER program must be generated from IBM-supplied macros and phases. POWER must be activated by the operator in order for any program to run under its control.

A number of considerations apply to the use of POWER under DOS/VS:

- The POWER program occupies one user partition and can service from one to four other user partitions.
- The size of the POWER program depends on the options used. Without remote job entry, the writer-only POWER program that supports one partition requires a minimum of 20K bytes of real storage; with remote job entry, it requires about 70K bytes of real storage with one terminal supporting four partitions. POWER must always run in real mode.

- Although POWER builds the input and output queues on a DASD, output for individual jobs can be directed to magnetic tape as intermediate storage.

New Devices Supported

Among the new I/O devices supported by DOS/VS are:

- All models of the IBM 3330/3333 and 3340 disk storage devices for attachment to CPU Models 125, 135, 145, 155-II, and 158. The 3340 can also be attached to the Model 115.
- The display operator console, which is the standard operator console on Models 115 and 125. This console allows the system to display messages at high speed on a cathode ray tube (CRT) screen. Various options, such as the redisplay feature and the facility for obtaining a printed copy of all the displayed messages on the 5213 Console Printer enhance operating flexibility. The display also replaces the manual control switches.
- The multi-function card unit (IBM 5425), which attaches to a Model 115 or 125 CPU, and which handles 96-column cards.
- The multifunction card machine (IBM 2560), which attaches to the Model 115 or 125.
- The IBM 5203 Printer, which attaches to the Model 115.
- The IBM 3203 Printer, which attaches to the Model 115 and 125.
- The IBM 3540 Diskette Input/Output Unit.
- The IBM 3881 Optical Mark Reader.

A complete list of the I/O equipment supported by DOS/VS is given in "Part 5. Configurations."

Compatibility

Current DOS users can transfer to DOS/VS support of the System/370 series with approximately the same effort normally required for a new version.

- Current DOS data files can be processed by DOS/VS, if compatible I/O devices are used. Programs written for a 2311 can be processed on a 3333, 3330, or 3340 attached to a System/370 Model 125 through the use of the 2311 Compatibility Feature of the System/370 Model 125. Programs written for the 1052 Console Printer-Keyboard can be processed on the video display and 5213 Console Printer of the Model 125 through the use of the 1052 Compatibility Feature of the System/370 Model 125.

- Existing assembler language source programs can be assembled by the DOS/VS Assembler, provided that no user written macros are called. If such macros are called, the user must either supply COPY instructions for the macro definitions at the beginning of all source decks in which the macros are used, or convert his library macros to edited macros and include them in the macro sublibrary.

- Existing high level language source programs can be compiled if the appropriate compiler is available for DOS/VS. COBOL D programs must be changed or converted with the Language Conversion Program before they can be processed by an ANS COBOL compiler. RPG programs must be adapted to RPG II.

- Previously compiled or assembled DOS object programs can be link-edited without modification under DOS/VS. User programs will execute provided the following points have been taken into account:

  - Programs that reference supervisor control blocks or manipulate data in the supervisor area of the first 512 bytes of storage (including specific bits such as the former PSW ASCII bit) may not run properly with DOS/VS. Because these areas are subject to change, user program compatibility can be protected only when user written routines employ STXIT and other appropriate IBM macro instructions.

  - Devices specified by the program must be available on the system on which DOS/VS will run.

  - Programs that depend on CPU circuitry not supported on System/370 may not execute properly.

  - Proper processing of time dependent programs run under earlier versions of DOS is not ensured.

  - Programs that deliberately create program checks may not run properly.

- Supervisor sizes will generally be larger in DOS/VS than with previous DOS versions. Therefore, in most cases programs will have to be relink-edited if they were not written to be self-relocating.

- User written I/O appendage routines must either run in real mode or adhere to certain restrictions which are described in the *DOS/VS System Management Guide.*

- Programs with self-modifying channel programs must run in real mode.

Part 4. DOS/VS Component Programs

The DOS/VS component programs are divided into two general categories: the system control programming and program products (see Figure 4.1).

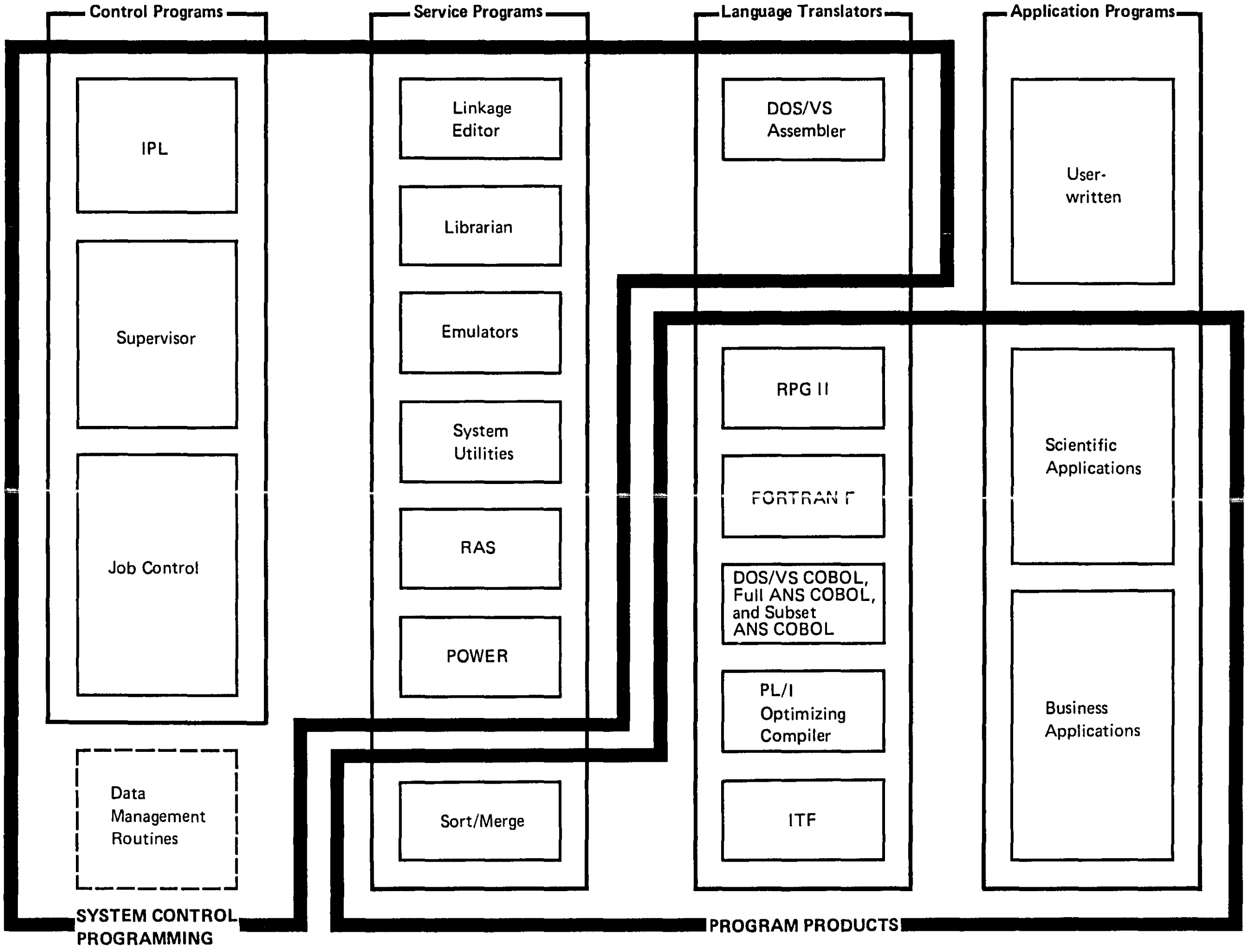


Figure 4.1. Program Structure Within DOS/VS

System Control Programming

This programming performs all functions necessary to generate, control, and service the Disk Operating System Virtual Storage. It controls the operation of program products and user-written problem programs. Because the functions that it performs are basic to the operation of the computing system, system control programming is supplied to all DOS/VS users without additional charge.

These programs and their functions are described in "Part 2. The Functions and Facilities of DOS/VS".

IBM also provides a number of SCP system utility programs that are essential to proper functioning of the system. These include:

- Alternate track assignment program for disks
- Programs for clearing disks
- Copy and restore programs for back-up data transfer between disk and tape, card, or other disk
- Initialization programs for disk and tape
- A program that displays a disk volume table of contents.

Program Products

A program product executes under supervision of the system control programming and is directly concerned with the processing of user programs or user data. Program products are optional and available for additional charge from IBM. Program products include compilers, a sort/merge program, an interactive terminal facility, and various business and scientific applications. The last category will not be dealt with in this manual.

RPG II Compiler

RPG II is a programming language in which a wide variety of commercial data processing applications may be implemented. The RPG II language is an expanded version of the previous RPG language provided with DOS. RPG II offers performance improvement in two areas:

- Improved storage efficiency for object programs
- Improved throughput performance for CPU-bound programs.

In addition, many new functions have been added to the RPG II language. These functions make RPG II:

- Easier to use than the previous version
- More flexible, so that the programmer can choose the method of coding most suited to his needs
- More powerful, so that many programs once difficult or impossible to code using RPG, can now be coded using RPG II.

For more details on this program product and the manuals available for it, consult the *RPG II General Information Manual,* GC21-5021.

FORTRAN Compiler

FORTRAN, a language specifically designed for the solution of scientific and computational problems, is supported under DOS/VS by the DOS FORTRAN F Compiler, a Type I program. Together with this compiler, the DOS FORTRAN IV Library Option 1 must be used if new I/O device support is desired. This is a program product providing support for the new I/O devices of the System/370.

For more details on the FORTRAN library, consult the *FORTRAN IV Library (Option 1) Program Product Specifications,* GC28-6882.

COBOL Compilers

IBM offers to DOS/VS users three COBOL compilers as program products: DOS/VS COBOL, full American National Standard (ANS) COBOL, and Subset ANS COBOL. These compilers translate ANS COBOL programs directly into object code, including most of the functions necessary for processing the user's data files. The remaining I/O functions are taken from the relocatable library by the linkage editor.

In order to be in a form suitable for the ANS COBOL compilers, COBOL D source programs can be converted by the Language Conversion program. The amount of conversion that must be done by direct programming varies with each application program. The COBOL D Compiler will also operate under DOS/VS. However, the current device support for object programs compiled by COBOL D has not been extended.

DOS/VS COBOL

This program product compiles programs written in the ANS COBOL language, and is designed for use under DOS/VS. DOS/VS COBOL contains all the functions of DOS COBOL compiler, version 3, as well as VSAM, 3886 OCR, 3340 direct access storage, 3540 diskette input output unit, 5202/3203 advanced printer support and the FIPS flagger, which identifies areas of the user's program that do not conform to the Federal Information Processing Standard. This compiler requires a partition size of at least 60K.

For more detail on this program product and the manuals available for it, consult the *DOS/VS COBOL General Information Manual,* GC28-6473.

Full ANS COBOL Compiler, Version 3

This program product compiles programs written using the full ANS COBOL language. COBOL Version 3 provides improvements that reduce the size of the object modules, extensive debugging facilities, alphabetized cross reference lists, and ASCII support. This compiler can run in a partition of at least 54K and requires use of the Object Library Program Product for execution of compiled programs.

For more details on this program product and the manuals available for it, consult the *Full ANS COBOL and Library General Information Manual,* GC28-6421.

Subset ANS COBOL Compiler, Version 1

This program product supports a subset of the ANS Standard COBOL language. Level 2 of the National Bureau of Standards is implemented, together with additional features taken from higher levels. The subset is more powerful than the COBOL D language, adheres to the ANSI standards, and has been given additional flexibility through special IBM extensions. This compiler can run in a partition of at least 20K and thus provides powerful features in a limited amount of space.

Programs written for the Subset COBOL compiler can also be compiled by the Full ANS COBOL compiler.

For more details on this program product and the manuals available for it, consult the *Subset ANS COBOL and Library General Information Manual,* GC28-6402.

PL/I Compiler

PL/I is a general purpose programming language which can be used to program both commercial and scientific applications, and is particularly useful for applications that require a combination of techniques to be used in a program.

PL/I support under DOS/VS is provided by the PL/I optimizing compiler and by two object-time libraries, the resident and transient libraries.

Source programs which were written for the PL/I D compiler can be. compiled by the new compiler provided that those programs are expressed in valid PL/I language.

A facility is provided by the new compiler to allow communication between PL/I modules and existing modules produced by certain FORTRAN and COBOL compilers.

The PL/I D compiler will also operate under DOS/VS. However, the current device support for object programs compiled by PL/I D has not been extended.

PL/I Optimizing Compiler

The PL/I optimizing compiler is designed to provide optimized object programs from a comprehensive level of PL/I. It provides a high level of PL/I language, diagnostics at both compile-time and object-time, and optimized object programs.

If optimization is specified, the compiler will process the PL/I source program, reorganizing it if necessary, so as to produce an efficient object program. If optimization is not specified, compilation time will be reduced.

The language level implemented by the optimizing compiler contains extensions beyond the PL/I D and the F level subsets.

For more details on this program product and the manuals available for it, consult the *PL/I Optimizer, Resident Library, and Transient Library General Information Manual,* GC33-0004.

PL/I Libraries

Two object-time libraries are required for the execution of object modules produced by the optimizing compiler. These libraries contain subroutines which must be combined with the object module to produce an executable program (the PL/I resident library), and other subroutines which are required dynamically as the object program is being executed (the Pl/I transient library).

Both the resident and transient libraries are separate program products.

Interactive Terminal Facility

The interactive terminal facility (ITF) allows engineers, analysts, executives, and other non-professional programmers to work directly with the computer through terminals. With ITF, the user keys in statements or problems and receives replies or answers immediately.

Two languages are available: ITF:BASIC and ITF:PL/I, a subset of PL/I.

For more details on this program product and the manuals available for it, consult the *ITF PL/I and BASIC General Information Manual,* GC28-6825.

DOS/VS Sort/Merge

A sort/merge program enables the user to sort multiple files of logical data records into a predetermined sequence, or to merge files of previously sequenced records.

The DOS/VS Sort/Merge Program Product, besides giving improved performance in virtual mode over DOS Sort/Merge, offers a number of additional functions. These include:

- Support of 3340 Disk Storage for input, output, and work files.
- Support of key-sequenced and entry-sequenced VSAM files for input to and output from the sort/merge.
- Ability to incorporate a user-written routine to read input for merging. This feature was previously available only for sorting applications.
- New control statements:
  - For specifying selection of records to be included in the sort/merge
  - For specifying reformatting of records
  - For requesting a summary of records
  - For specifying a user-defined collating sequence.

For more details on this program product and the manuals available for it, consult the publication *IBM DOS/VS Sort/Merge General Information,* GC33-4030.

Part 5. Configurations

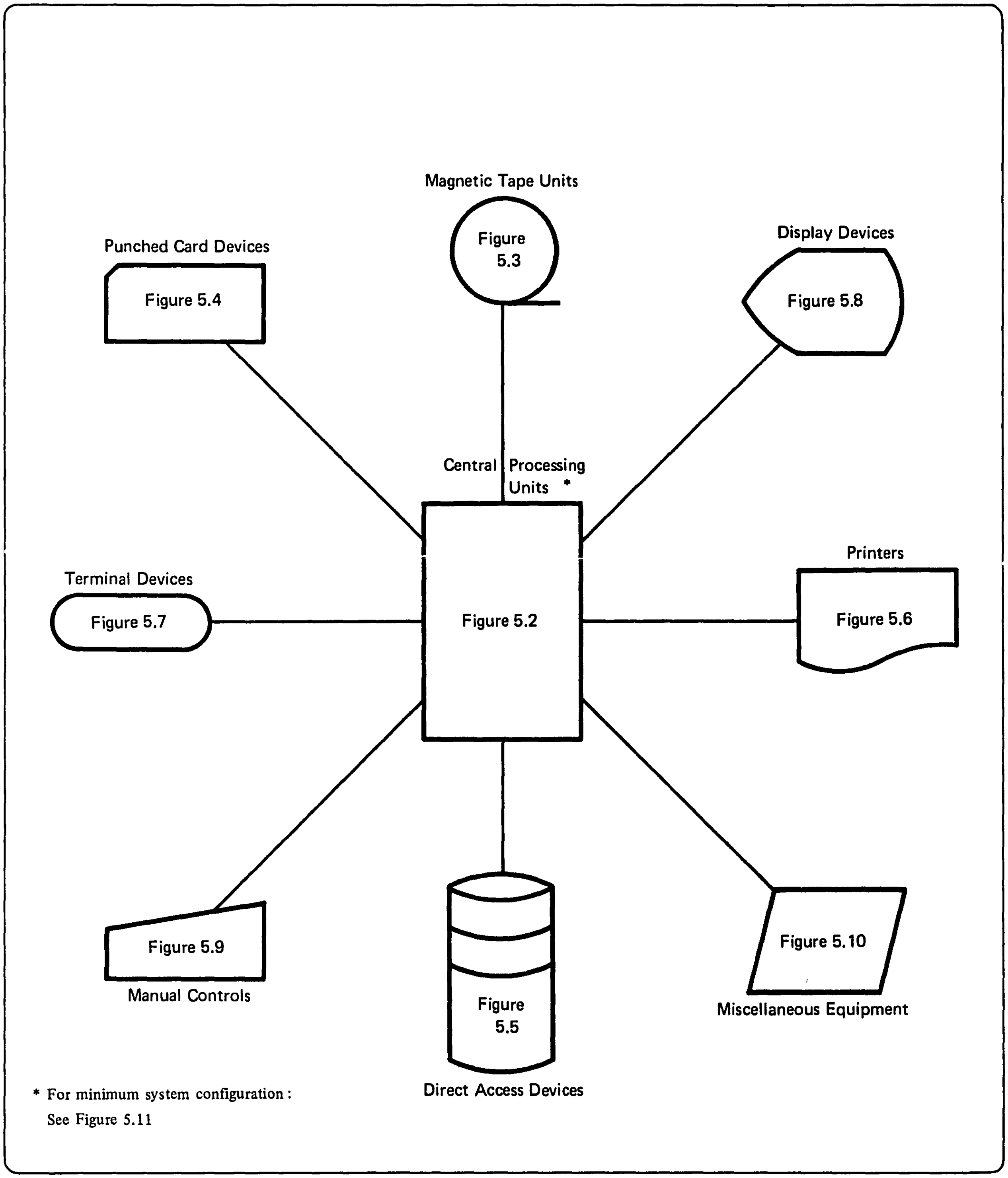


Figure 5.1. IBM System/370 Configurations Supported by DOS/VS

In each of the figures referenced there is a full list of devices supported.

System/370		Central Processing Units	
No.	Model	Storage size in bytes	Remarks
3115	F	65,536	
	FE	98,304	
	G	131,072	
	GE	163,840	
3125	FE	98,304	
	G	131,072	
	GE	163,840	
	GF	196,608	
	H	262,144	
3135	FE	98,304	
	GD	147,456	
	GF	196,608	
	DH	245,760	
	H	262,144	
	HF	327,680	
	HG	393,216	
	I	524,288	
3145	GE	163,840	
	GFD	212,992	
	H	262,144	
	HG	393,216	
	I	524,288	
	H2	262,144	
	HG2	393,216	
	I2	524,288	
	IH2	786,432	
	J2	1,048,576	
3155	H	262,144	
II	HG	393,216	
	I	524,288	
	IH	786,432	
	J	1,048,576	
	JI	1,572,864	
	K	2,097,152	
3158	I	524,288	
	J	1,048,576	
	JI	1,572,864	
	K	2,097,152	
	KJ	3,145,728	
	L	4,194,304	
	MP1	524,288	
	MP2	1,048,576	
	MP3	1,572,864	
	MP4	2,097,152	
	MP5	3,145,728	
	MP6	4,194,304	

Figure 5.2. Central Processing Units

System/370		Magnetic Tape Devices				
No.	Model	Name	Maximum Data Rates		Remarks	Control Unit
			Kilobytes per second	Bytes per inch		
2401	1	Magnetic Tape Unit	30	800	Not attachable to a Model 115 or 125	2803 or 2804
	2	Magnetic Tape Unit	60	800		
	3	Magnetic Tape Unit	90	800		
	4	Magnetic Tape Unit	60	1600		
	5	Magnetic Tape Unit	120	1600		
	6	Magnetic Tape Unit	180	1600		
	8	Magnetic Tape Unit	60	800		
2415	1-3	Magnetic Tape Unit and Control	15	800		none
	4-6	Magnetic Tape Unit and Control	30	1600		
2420	5	Magnetic Tape Unit	160	1600		2803
	7	Magnetic Tape Unit	320	1600		
2495	1	Tape Cartridge Reader	.9	20		none
3410	1	Magnetic Tape Unit	20	1600	See Note 1	3411
	2	Magnetic Tape Unit	40	1600		
	3	Magnetic Tape Unit	80	1600		
3420	3	Magnetic Tape Unit	120	1600	Not attachable to a Model 115 or 125	Note 2
	4	Magnetic Tape Unit	470	6250		Note 3
	5	Magnetic Tape Unit	200	1600		Note 2
	6	Magnetic Tape Unit	780	6250		Note 3
	7	Magnetic Tape Unit	320	1600		Note 2
	8	Magnetic Tape Unit	1250	6250		Note 3

Note 1: Three models of the 3411 Magnetic Tape unit with Control are available. These are identical to the 3410 Magnetic Tape Unit models except that the control unit is built in.

Note 2: Attaches to the 3803 Models 1 and 2.

Note 3: Attaches to the 3803 Model 2 only.

Figure 5.3. Magnetic Tape Units

System/370		Punched Card Devices					
No.	Model	Name	Maximum Speed			Control Unit	Remarks
			Reading	Punching			
			Cards per minute	Cols. per second	Cards per minute		
1442	N1	Card Read Punch	400	160	-	none	
	N2	Card Punch	-	160	-		
2501	B1	Card Reader	600	-	-	none	
	B2	Card Reader	1000	-	-		
2520	B1	Card Read Punch	500	-	500	none	
	B2	Card Punch	-	-	500		
	B3	Card Punch	-	-	300		
2540	1	Card Read Punch	1000	-	300	2821	
2560	A1	Multifunction Card Machine	500	260	-	none	For Models 115 and 125 only. (See Note 3)
2596		Card Read Punch (Note 1)	500		120	none	96-column card machine
3504	A1 A2	Card Reader Card Reader	800 1200			none	For Model 125 only (See Note 2)
3505	A1,B1	Card Reader	800	-	-	none	
	A2,B2	Card Reader	1200	-	-		
3525	P1	Card Punch	-	100	-	3505 Card Reader	See Note 2
	P2	Card Punch	-	200	-		
	P3	Card Punch	-	300	-		
5425	A3	Multifunction Card Unit	250	-	60	none	For Models 115 and 125 only. (See Note 3)
	A4	Multifunction Card Unit	500	-	120	none	96-column card machine.

Note 1: DOS/VS and POWER do not support SYSRDR and SYSPCH files on this device.

Note 2: The following devices may be attached natively to a Model 125, either:
1) One 5425, or
2) One 2560 and one 3504 Model A1 or A2, or
3) One 3504 Model A1 or A2 and one 3525.

Note 3: To a Model 115 either one 2560 or one 5425 can be attached.

Figure 5.4. Punched Card Devices

System/370		Direct Access Devices			
No.	Model	Name	Million Bytes Capacity (Max.)	Control Unit	Remarks
2311	1	Disk Storage Drive	7	2841	See Note 1
2312	A1	Disk Storage	29	2314	
2313	A1	Disk Storage	117	2314	
2314	A1,B1	Direct Access Storage Facility	233	none	Note attachable to a Model 115 or 125
2318	A1	Disk Storage	58	2314	
2319	A-series	Disk Storage	87	2314	
	B-series	Disk Storage			
2321	1	Data Cell Drive	400	2841	See Note 1
3333	1	Disk Storage	200	Note 2	Not attachable to a Model 115
3330	1	Disk Storage	200	Note 3	
	2	Disk Storage	100	-	
3340	A2	Direct Access Storage	140	Note 4	
	B1	Direct Access Storage	70	Note 5	
	B2	Direct Access Storage	140	Note 5	

Note 1: The 2311 and 2321 are not supported as the system residence device.

Note 2: Attaches to the integrated file adapter (IFA) on Model 135, to the 3345 Control Storage on Model 145, to the integrated storage control (ISC) on Model 158, or to the 3830-2 Storage Control Unit on Models 135 and 145. It also attaches directly to the Model 125.

Note 3: Attaches to the 3333-1 and 3830-1.

Note 4: Attaches directly to Models 115 and 125, to the integrated file attapter (IFA) on Model 135, to the integrated storage control (ISC) on Models 145 and 158, or to the 3830-2 Storage Control Unit on Models 135, 145, 155-II, and 158.

Note 5: Model B1 or B2 of the 3340 attaches directly to a Model A2 or to another B model.

Figure 5.5. Direct Access Devices

System/370		Printers			
No.	Model	Name	Control Unit	Max. Print Speed	Remarks
1403	2, 7 3, N1	Printer Printer	2821	600 lines per minute 1100 lines per minute	Selective Tape Listing feature is excluded
1443	N1	Printer	none	240 lines per minute	
3211	1	Printer	3811	2000 lines per minute	Not attachable to Model 115
3213		Console Printer	none	85 chars per second	For Model 158 only (Notes 1 and 4)
5213	1	Console Printer	none	85 chars. per second	For Model 125 (Note 2) and Model 115 (see also Note 4)
3203	1 2	Printer Printer	none none	600 lines per minute 1200 lines per minute	For Models 115 and 125 only
5203	3	Printer	none	300 lines per minute	For Model 115 only (Note 3)

Note 1: The 3213 is required to operate the Model 158 display console in 3215 mode.

Note 2: The 5213 is required to operate the Model 125 display console in 3052 mode.

Note 3: If the 5203 is the only printer on the system, it must have at least 120 print positions.

Note 4: POWER does not support the 3213 or the 5213.

Figure 5.6. Printers

System/370		Terminal Devices		
No.	Model	Name	Control Unit	Remarks
1030		Data Collection System	2701 2702 2703 3704 3705	
1050		Data Communication System		
1060		Data Communication System		
2721		Portable Audio Terminal	7770	
2740	1, 2	Communication Terminal	2701 2702 2703 3704 3705	
2760		Optical Image Unit		
2770		Data Communication System	2701 2703 3704 3705	
2780	1 - 4	Data Transmission Terminal		
2790		Data Communication System	2715	The 2790 with the 2715 can be attached to the CPU either directly or via a 2701/2703/3705/ICA
2972		Banking Terminal	2701 2703 3704 3705	
3735		Programmable Buffered Terminal		
3780		Data Communication Terminal		

Note: In addition to the above terminal devices, DOS/VS supports TP attachment to the CPUs of the systems 1130, 1800, System/3, System/7, System/360 and System/370. Devices supported by former releases are also supported by DOS/VS. All the specified devices, except the 7770, can be attached via the ICA.

Figure 5.7. Terminal Devices

System/370		Display Devices			
No.	Model	Name	Characters	Control Unit *	Remarks
2260	1	Display Station	960	2848	Local or remote attachment
	2	Display Station	480		
2265		Display Station	960	2845	For remote attachment only
3270		Information Display System		3272	Local or remote attachment

* For remote attachment a 2701 control unit is required.

Figure 5.8. Display Devices

System/370		Manual Controls			
No.	Model	Name	Speed	Control Unit	Remarks
3210	1, 2	Console Printer-Keyboard	15.5 cps	none	Not attachable to System/370 Models 115, 125, and 158 *
3215	1	Console Printer-Keyboard	85 cps	none	

* 3215 operation mode on the System/370 Model 158 display console requires the 3213 printer plus attachment features.

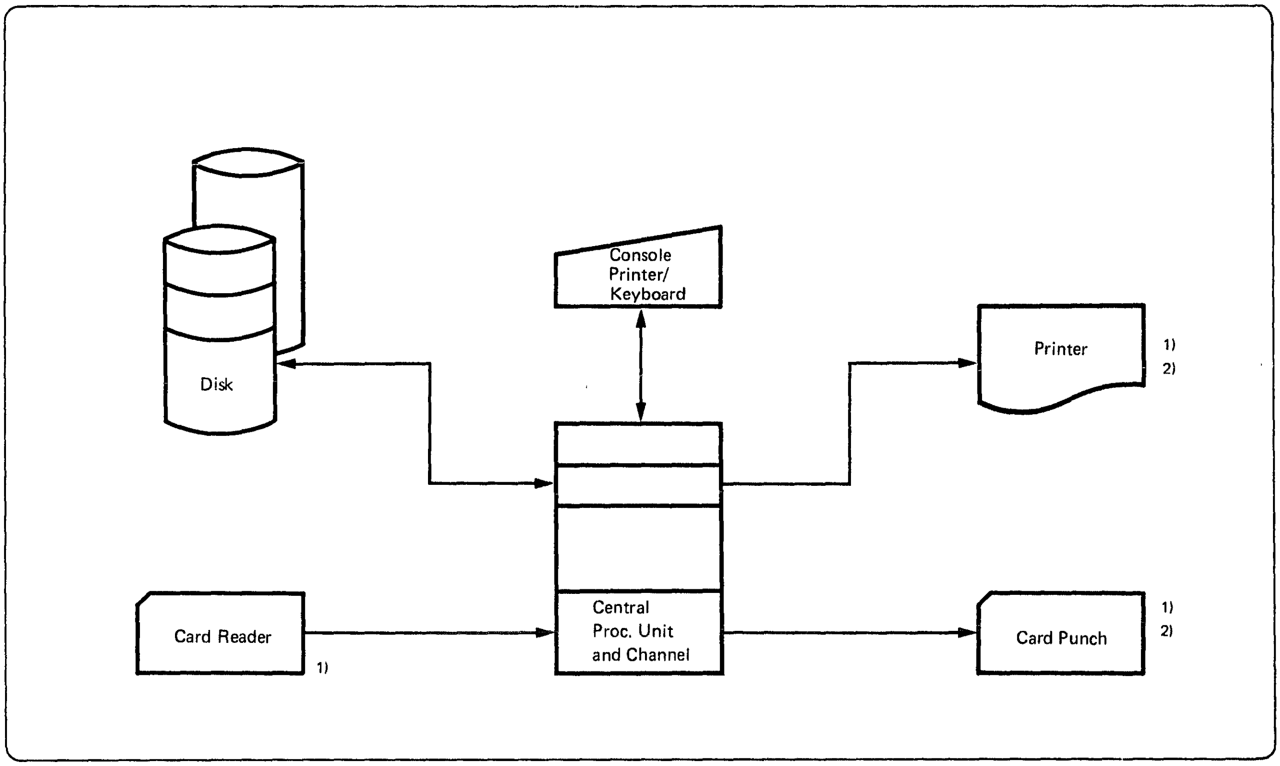
Figure 5.9. Manual Controls

System/370		Miscellaneous Equipment			
No.	Model*	Name	Max. Speed	Control Unit	Remarks
1017	1,2	Paper Tape Reader	120 cps	2826	
1018	1	Paper Tape Punch	120 cps	2826	
1255	1-3	Magnetic Character Reader	750 dpm	none	
1259	2	Magnetic Character Reader	600 dpm	none	
1270	1-4	Optical Character Reader	750 dpm	none	
1275	2,4	Optical Character Reader	1600 dpm	none	
1287	1-5	Optical Reader	665 dpm	none	
1288	1	Optical Page Reader		none	
1419	1	Magnetic Character Reader	1600 dpm	none	
2671	1	Paper Tape Reader	1000 cps	2822	
2816	1	Tape Switching Unit		2803	
3540	B1,B2	Diskette I/O Unit	3636 rpm input 2212 rpm output		none
3881	1	Optical Mark Reader	**	none	For Models 135 and 145 only
3886	1	Optical Character Reader	100 dpm	none	
7770	3	Audio Response Unit		none	

* For the additional models that are available outside of the United States of America, refer to the current sales manual.

** For 8 1/2" x 11 1/2" documents, approximately 4,000 documents can be read per hour. Higher throughput rates occur for documents shorter than 11 inches. Approximately 6,000 documents can be read per hour for 3" documents.

Figure 5.10. Miscellaneous Equipment



Notes: 1) The card reader, card punch, and printer can each be replaced by a magnetic tape unit or by a disk extent. This does not apply to the card reader during IPL.

2) The printer and the card punch may be replaced by one magnetic tape unit.

Figure 5.11. Minimum Practical System Configuration

Part 6. DOS/VS Documentation

A full set of manuals and educational courses is available to describe the Disk Operating System Virtual Storage and its use. Like DOS/VS itself, the documentation is a functional enhancement of previous releases and editions. The DOS/VS library has been revised to consolidate coverage of each major subject into the proper manual, thus reducing the time needed to find a specific topic or item.

Education

IBM offers an array of education courses and manuals answering the needs of both users new to data processing and users new only to DOS/VS or some of its applications. For a summary of the educational help offered, consult the *IBM Data Processing Education Selector Guide,* GR20-1055, and the *IBM DOS Education Selector Guide,* GR20-2330.

The DOS/VS SCP Library

The DOS/VS SCP library is a set of manuals describing the functions and uses of the DOS/VS system control programming and the operation of the system. It is divided into several topical groups and logical types of manuals.

Topical Groups in the SCP Library

The SCP library can be divided into the following eight topical groups:

System Generation
System Control and Service
Data Management
Operation
System Utilities
Teleprocessing
Emulation
Assembler

System Control
Programming Library
(SCP Library)

Titles of the manuals that fall into these topical groups can be found in Figure 6.1.

Types of Manuals in the SCP Library

Wherever appropriate in the SCP library, a distinction is made between several levels of information, each level serving a different purpose:

1. **Descriptive Information**
   Descriptive information is aimed at developing full understanding of a component or group of components and the part they play in the working of the system. In developing a topic, a descriptive manual or section attempts to address practical implications by means of examples and careful explanation. This information is contained in manuals called Guides.

2. **Reference Information**
   This type of information represents the concise specifications for using a feature or component, and is contained in manuals that are reference sources in both name and design. Accompanying explanatory text is reduced to a minimum, allowing rapid retrievability of information. Figure 6.1 shows a number of manuals, particularly in the group dealing with basic system functions, classified entirely as guides or reference manuals. The *DOS/VS System Management Guide,* for example, and the *DOS/VS Data Management Guide* contain the necessary descriptive information on the basic functions of the system while *DOS/VS System Control Statements* and *DOS/VS Supervisor and I/O Macros* are quick-reference sources for the corresponding control statements and macro instructions.

   In other instances, both descriptive and reference information may properly be contained within a single manual, one that fully covers a topic, such as VSAM Access Method Services, within one volume.

Manuals in the Program Product Library

The *IBM System/360 and System/370 Bibliography,* GA22-6822, lists the specific manuals available with each program product.

Program Logic Information

Logic manuals, in presenting the internal details of system programs and components, mix reference and descriptive/tutorial information. In their reference role, logic manuals contain information such as register usage by the program, or layout of data areas. They take on a more descriptive role when presenting the module-by-module logic and the overall flow of control within the program. Constructed in this way for readers who need to know (or learn) program internals, they consolidate logic information on a particular programming topic into one volume.

Figure 6.1, Part 1: The DOS/VS System Library Manuals. *The overall subject of DOS/VS is logically divided into several major topics such as Operation, Data Management, or Teleprocessing, and dealt with in descriptive, reference, and logic manuals as appropriate.*	**Descriptive**	**Reference**	**Logic**
	Develops a full understanding of the subject and the part it plays in the overall working or use of the system.	**The concise specifications for using the components of the system, organized for ease of access.**	**Internals information to round out the user's understanding of the flow of logic in the system and its components.**
System Generation How to prepare, build, and maintain a Disk Operating System	DOS/VS System Generation GC33-5377		
Maintenance Serviceability aid intended for persons involved in program maintenance		DOS/VS Handbook SY33-8571	
System Control & Service Use of the system, its libraries, and control and service functions for the processing of programs in both batch and multiprogramming environments.	DOS/VS System Management Guide GC33-5371	DOS/VS System Control Statements Reference GC33-5376 DOS/VS System Utilities Reference GC33-5381 DOS/VS Supervisor and I/O Macros Reference GC33-5373	DOS/VS Supervisor Logic SY33-8551 DOS/VS Linkage Editor Logic SY33-8556 DOS/VS Error Recovery and Recording Transients Logic SY33-8552 DOS/VS POWER Logic SY33-8565 DOS/VS Logical Transients Logic SY33-8553 DOS/VS POWER RJE Logic SY33-8566 DOS/VS IPL & Job Control Logic SY33-8555 DOS/VS System Serviceability Aids Logic SY33-8554 DOS/VS Librarian Logic SY33-8557 DOS/VS System Utilities Logic SY33-8558
Data Management Use of storage media and data organization and access methods for the preparation and manipulation of data.	DOS/VS Data Management Guide GC33-5372	DOS/VS Tape Labels Reference GC33-5374 DOS/VS DASD Labels Reference GC33-5375 DOS/VS System Utilities: Access Method Services GC33-5382	DOS/VS LIOCS Vol. 1 Introduction and Imperative Macros Logic SY33-8559 DOS/VS LIOCS Vol. 3 DAM and ISAM Logic SY33-8561 DOS/VS LIOCS Vol. 2 SAM Logic SY33-8560 DOS/VS LIOCS Vol. 4 VSAM Logic SY33-8562 DOS/VS Access Method Services Logic SY33-8564

Figure 6.1. DOS/VS Documentation (Part 1 of 3)

Figure 6.1, Part 2: The DOS/VS System Library Manuals. *The overall subject of DOS/VS is logically divided into several major topics, such as Operation, Data Management, or Teleprocessing, and dealt with in descriptive, reference, and logic manuals as appropriate.*	**Descriptive**	**Reference**	**Logic**
	Develops a full understanding of the subject and the part it plays in the overall working or use of the system.	**The concise specification for using the components of the system,organized for ease of access.**	**Internals information to round out the user's understanding of the flow of logic in the system and its components.**
Operation Running monitoring, and directing the system for the processing of jobs, and initiatinq the proper steps for recovery from errors.	DOS/VS Operating Procedures GC33-5378 DOS/VS Serviceability Aids & Debugging Procedures GC33-5380	IBM System/370 Model 1xx Operation Procedures DOS/VS Messages Reference GC33-5379 DOS/VS On-Line Test Executive Program (OLTEP) Reference GC33-5383	DOS/VS On-Line Test Executive Program (OLTEP) Logic SY33-8568
Assembler Using all available machine instructions directly, and striking the best balance between storage utilitization and speed of program execution.		OS/VS DOS/VS VM/370 Assembler Language Guide GC33-4010 Guide to the DOS/VS Assembler GC33-4024	DOS/VS Assembler Logic SY33-8567
Teleprocessing Processing of data obtained from remote terminals using communications access methods.		DOS/VS Basic Telecommunications Access Method - Reference GC27-6989 QTAM Message Control Program Guide GC27-6986 QTAM Message Processing Program Services GC27-6985	DOS/VS BTAM Logic SY27-7251 DOS/VS QTAM Message Control Program Logic SY27-7249

Figure 6.1. DOS/VS Documentation (Part 2 of 3)

Figure 6.1, Part 3: The DOS/VS System Library Manuals. *The overall subject of DOS/VS is logically divided into several major topics such as Operation, Data Management, or Teleprocessing, and dealt with in descriptive, reference, and logic manuals as appropriate.*	**Descriptive**	**Reference**	**Logic**
	Develops a full understanding of the subject and the part it plays in the overall working or use of the system.	**The concise specifications for using the components of the system organized for ease of access.**	**Internals information to round out the user's understanding of the flow of logic in the system and its components.**
Emulation Use of data and program on System/370 that were developed for another system.		1401/1440/1460 DOS/VS Emulator on System/370 GC33-5384	1401/1440/1460 DOS/VS Emulator on System/370: Logic SY33-7008
		1410/7010 DOS/VS Emulator on System/370 GC33-5385	1410/7010 DOS/VS Emulator on System/370: Logic SY33-7009
	Moving from Model 20 to DOS/VS GC33-5386	Model 20 Emulator on System/370 : Reference GC33-5388	Model 20 Emulator on System/370 Using DOS and DOS/VS: Logic SY33-7010
	Moving from System/3 to DOS/VS GC33-5389		
	IBM Emulator for Honeywell Series 200 on System/370 using DOS and DOS/VS: Transition Guide GH20-1153	IBM Emulator for Honeywell Series 200 on System/370 using DOS and DOS/VS: Reference GA24-3604	IBM Emulator for Honeywell Series 200 on System/370 using DOS and DOS/VS: Logic LY24-3606
	IBM Emulator for RCA 301 on System/370 using DOS and DOS/VS: Transition Guide GH20-1152	IBM Emulator for RCA 301 on System/370 using DOS and DOS/VS: Reference GA24-3605	IBM Emulator for RCA 301 on System/370 using DOS and DOS/VS: Logic LY24-3607

Figure 6.1. DOS/VS Documentation (Part 3 of 3)

Glossary

IBM is grateful to the American National Standards Institute (ANSI) for permission to reprint its definitions from the American National Standard Vocabulary for Information Processing (Copyright 1970 by American National Standards Institute, Incorporated), which was prepared by Subcommittee X3K5 on Terminology and Glossary of American National Standards Committee X3. ANSI definitions are preceded by an asterisk (*).

access method: A technique for moving data between virtual storage and input/output devices.

***address:** (1) An identification, as represented by a name, label, or number, for a register, location in storage, or any other data source or destination such as the location of a station in a communication network. (2) Loosely, any part of an instruction that specifies the location of an operand for the instruction.

alternate track: One of a number of tracks set aside on a disk pack for use as alternatives to any defective tracks found elsewhere on the disk pack.

application program: A program written by a user that applies to his own work.

assembler language: A source language that includes symbolic machine language statements in which there is a one-to-one correspondence with the instruction formats and data formats of the computer.

attach: (1) To create a task and present it to the supervisor. (2) A macro instruction that causes the control program to create a new task and indicates the entry point in the program to be given control when the new task becomes active.

auxiliary storage: Data storage other than real storage; for example, storage on magnetic tape or disk. Synonymous with external storage, secondary storage.

blocking: Combining two or more logical records into one block.

blocking factor: The number of logical records combined into one physical record or block.

book: A group of source statements written in any of the languages supported by DOS/VS and stored in a source statement library.

buffer: An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. Synonymous with I/O area.

byte: A sequence of eight adjacent binary digits that are operated upon as a unit and that constitute the smallest addressable unit of the system.

card punch: A device to record information in cards by punching holes in the cards to represent letters, digits, and special characters.

card reader: A device which senses and translates into machine code the holes in punched cards.

catalog: To enter a phase, module, book, or procedure into one of the system or private libraries.

***central processing unit**: A unit of a computer that includes the circuits controlling the interpretation and execution of instructions. Abbreviated CPU.

channel: (1)* A path along which signals can be sent, for example, data channel, output channel. (2) A hardware device that connects the CPU and real storage with the I/O control units.

compile: To prepare a machine language program from a computer program written in a high level language by making use of the overall logic structure of the program, or generating more than one machine instruction for each symbolic statement, or both, as well as performing the function of an assembler.

compiler: A program that translates high level language statements into machine language instructions.

configuration: The group of machines, devices, etc., which make up a data processing system.

control area: A group of control intervals used as a unit for formatting a file before adding records to it. Also, in a key-sequenced file, the set of control intervals pointed to by the lowest level index; used by VSAM for distributing free space and for placing a low-level index adjacent to its data.

control interval: A fixed-length area of auxiliary storage space in which VSAM stores records and distributes free space. It is the unit of information transmitted to or from auxiliary storage by VSAM, independent of blocksize.

control program: A program that is designed to schedule and supervise the performance of data processing work by a computing system.

control unit: A device that controls the reading, writing, or display of data at one or more input/output devices.

core image library: A library of phases that have been produced as output from link-editing. The phases in the core image library are in a format that is executable either directly or after processing by the relocating loader in the supervisor.

CPU busy time: The amount of time devoted by the central processing unit to the execution of instructions.

data file: A collection of related data records organized in a specific manner. For example, a payroll file (one record for each employee, showing his rate of pay, deductions, etc. or an inventory item, showing the cost, selling price, number in stock, etc.) See also file.

data integrity: See integrity.

data management: A major function of DOS/VS that involves organizing, storing, locating, retrieving, and maintaining data.

data security: See security.

deblocking: The action of making the first and each subsequent logical record of a block available for processing one record at a time.

default value: The choice among exclusive alternatives made by the system when no explicit choice is specified by the user.

deletion of an I/O device: Removal of the I/O unit from the supervisor configuration tables.

diagnostic routine: A program that facilitates computer maintenance by detection and isolation of malfunctions or mistakes.

dial-up terminal: A terminal on a switched teleprocessing line.

direct access: (1) Retrieval or storage of data by a reference to its location on a volume, other than relative to the previously retrieved or stored data. (2)* Pertaining to the process of obtaining data from, or placing data into, storage where the time required for such access is independent of the location of the data most recently obtained or placed in storage. (3)* Pertaining to a storage device in which the access time is effectively independent of the location of the data. Synonymous with random access.

direct organization: Direct file organization implies that for purposes of storage and retrieval there is a direct relationship between the contents of the records and their addresses on disk storage.

directory: An index that is used by the system control and service programs to locate one or more sequential blocks of program information that are stored on direct access storage.

disk pack: A direct access storage volume containing magnetic disks on which data is stored. Disk packs are mounted on a disk storage drive, such as the IBM 3330 Disk Storage Drive.

diskette: Consists of a flexible magnetic oxide coated disk, permanently enclosed in a semi-rigid protective plastic jacket approximately 8 inches square. During data processing operations, the disk turns freely within the jacket. It is capable of storing 1898 128-character data records.

distributed free space: Space reserved within the control intervals of a key-sequenced file for inserting new records into the file in key sequence; also, whole control intervals reserved in a control area for the same purpose.

dump: (1) To copy the contents of all or part of virtual storage. (2) The data resulting from the process as in (1).

dynamic address translation (DAT): (1) The change of a virtual storage address to an address in real storage during execution of an instruction. (2) A hardware function that performs the translation.

entry sequence: The order in which data records are physically arranged in auxiliary storage, without respect to their contents (contrast with key sequence).

entry-sequenced file: A VSAM file whose records are loaded without respect to their contents, and whose relative byte addresses cannot change. Records are retrieved and stored by addressed access, and new records are added to the end of the file.

error message: The communication that an error has been detected.

error recovery procedures: Procedures designed to help isolate, and, when possible, to recover from errors in equipment. The procedures are often used in conjunction with programs that record the statistics of machine malfunctions.

***file**: A collection of related records treated as a unit. For example, one line of an invoice may form an item, a complete invoice may form a record, the complete set of such records may form a file, the collection of inventory control files may form a library, and the libraries used by an organization are known as its data bank.

hard copy: A printed copy of machine output in a visually readable form, for example, printed reports, listings, documents, and summaries.

***hardware**: Physical equipment, as opposed to the computer program or method of use, for example, mechanical, magnetic, electrical, or electronic devices. Contrast with software.

***idle time**: That part of available time during which the hardware is not being used.

index: (1)* An ordered reference list of the contents of a file or document, together with keys or reference notations for identification or location of those contents. (2) A table used to locate the records of an indexed sequential file.

indexed-sequential organization: The records of an indexed sequential file are arranged in logical sequence by key. Indexes to these keys permit direct access to individual records. All or part of the file can be processed sequentially.

Initial Program Load (IPL): The initialization procedure that causes DOS/VS to commence operation.

integrity: Preservation of data or programs for their intended purpose.

***interface**: A shared boundary. An interface might be a hardware component to link two devices or it might be a portion of storage or registers accessed by two or more computer programs.

***I/O**: An abbreviation for input/output.

ISAM interface program: A set of routines that allow a processing program coded to use ISAM to gain access to a key-sequenced file with an index.

job: (1)* A specified group of tasks prescribed as a unit of work for a computer. By extension, a job usually includes all necessary computer programs, linkages, files, and instructions to the operating system. (2) A collection of related problem programs, identified in the input stream by a JOB statement followed by one or more EXEC statements.

job accounting interface: A function that accumulates, for each job step, accounting information that can be used for charging usage of the system, planning new applications, and supervising system operation more efficiently.

job control: A program that is called into storage to prepare each job or job step to be run. Some of its functions are to assign I/O devices to certain symbolic names, set switches for program use, log (or print) job control statements, and fetch the first program phase of each job step.

job (JOB) statement: The job control statement that identifies the beginning of a job. It contains the name of the job.

job step: The execution of a single processing program.

K: 1024.

***key**: One or more characters associated within an item of data that are used to identify it or control its use.

key sequence: The collating sequence of data records, determined by the value of the key field in each of the data records. May be the same as, or different from, the entry sequence of the records.

key-sequenced file: A file whose records are loaded in key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in the file in key sequence by means of distributed free space. Relative byte addresses of records can change.

label: Identification record for a tape or disk file.

language translator: A general term for any assembler, compiler, or other routine that accepts statements in one language and produces equivalent statements in another language.

leased facility: A circuit of the public telephone network made available for the exclusive use of one subscriber.

librarian: The set of programs that maintains, services, and organizes the system and private libraries.

library: A collection of files or programs, each element of which has a unique name, that are related by some common characteristic. For example, all phases in the core image library have been processed by the linkage editor.

linkage editor: A processing program that prepares the output of language translators for execution. It combines separately produced object modules, resolves symbolic cross references among them, generates overlay structures on request, and produces executable code (a phase) that is ready to be fetched or loaded into virtual storage. The linkage editor also produces relocatable phases.

load: (1)* In programming, to enter instructions or data into storage or working registers. (2) In DOS/VS, to bring a program phase from a core image library into virtual storage for execution.

message: See error message, operator message.

microprogramming: A method of working of the CPU in which each complete instruction starts the execution of a sequence of instructions, called microinstructions, which are generally at a more elementary level.

multiprogramming system: A system that controls more than one program simultaneously by interleaving their execution.

multitasking: The concurrent execution of one main task and one or more subtasks in the same partition.

object code: Output from a compiler or assembler which is suitable for processing to produce executable machine code.

***object module:** A module that is the output of an assembler or compiler and is input to a linkage editor.

object program: A fully compiled or assembled program. Contrast with source program.

***online:** (1) Pertaining to equipment or devices under control of the central processing unit. (2) Pertaining to a user's ability to interact with a computer.

operand: (1)* That which is operated upon. An operand is usually identified by an address part of an instruction. (2) Information entered with a command name to define the data on which the command processor operates and to control the execution of the command processor.

operator command: A statement to the control program, issued via a console device, which causes the control program to provide requested information, alter normal operations, initiate new operations, or terminate existing operations.

operator message: A message from the operating system or a problem program directing the operator to perform a specific function, such as mounting a tape reel, or informing him of specific conditions within the system, such as an error condition.

***overflow:** (1) That portion of the result of an operation that exceeds the capacity of the intended unit of storage. (2) Pertaining to the generation of overflow as in (1).

overlay: (1) A program segment (phase) that is loaded into virtual storage. It replaces all or part of a previously retrieved section. (2) The process of replacing a previously retrieved program section in virtual storage by another section.

page: (1) A fixed-length block of instructions, data or both, that can be transferred between real storage and the page data set. In DOS/VS, a page is 2K bytes in length. (2) To transfer instructions, data, or both between real storage and the page data set.

page data set: An extent in auxiliary storage, in which pages are stored.

page frame: A block of real storage that can contain a page.

page in: The process of transferring a page from the page data set to real storage.

page out: The process of transferring a page from real storage to the page data set.

page pool: The set of all page frames available for paging virtual-mode programs.

paging: The process of transferring pages between real storage and the page data set.

***parameter**: A variable that is given a constant value for a specific purpose or process.

peripheral equipment: A term used to refer to card devices, magnetic tape and disk devices, printers, and other equipment bearing a similar relation to the CPU.

phase: The smallest complete unit that can be referred to in the core image library.

printer: A device that expresses coded characters as hard copy.

priority: A rank assigned to a partition that determines its precedence in receiving CPU time.

private library: A user-owned library that is separate and distinct from the system library.

problem determination aid: A program that traces a specified event when it occurs during the operation of a program. Abbreviated PDAID.

problem program: Any program that is executed when the central processing unit is in the problem state; that is, any program that does not contain privileged instructions. This includes IBM-distributed programs, such as language translators and service programs, as well as programs written by a user.

processing program: (1) A general term for any program that is not a control program. (2) Synonymous with problem program.

processor storage: The general purpose storage of a computer. Processor storage can be accessed directly by the operating registers. Synonymous with real storage.

queue: (1) A waiting line or list formed by items in a system waiting for service; for example, tasks to be performed or messages to be transmitted in message switching system. (2) To arrange in, or form, a queue.

random processing: The treatment of data without respect to its location in auxiliary storage, and in an arbitrary sequence governed by the input against which it is to be processed.

real address: The address of a location in real storage.

real address area: In DOS/VS, the area of virtual storage where virtual addresses are equal to real addresses.

real mode: In DOS/VS, the mode of a program that may not be paged.

real partition: In DOS/VS, a division of the real address area of virtual storage that may be allocated for programs that are not to be paged, or virtual programs that contain pages that are to be fixed.

real storage: The storage of a System/370 computing system from which the central processing unit can directly obtain instructions and data, and to which it can directly return results. Synonymous with processor storage.

relocatable library: A library of relocatable object modules and IOCS modules required by various compilers. It allows the user to keep frequently used modules available for combination with other modules without recompilation.

relocatable phase: Output of the linkage editor containing relocation information. The relocating loader in the supervisor uses this information to relocate the phase into any partition the user selects at execution time.

restore: To return a data file created previously by a copy operation from cards, disk, or magnetic tape to disk storage.

***routine**: An ordered set of instructions that may have some general or frequent use.

secondary storage: Same as auxiliary storage.

security: Prevention of access to or use of data or programs without authorization.

sequential organization: Records of a sequential file are arranged in the order in which they will be processed.

service program: A program that assists in the use of a computing system, without contributing directly to the control of the system or the production of results.

shared virtual area: An area located in the highest addresses of virtual storage. It can contain a system directory list (SDL) of frequently-used phases, resident programs that can be shared between partitions, and an area for system GETVIS support.

software: A set of programs, concerned with the operation of the hardware in a data processing system.

source: The statements written by the programmer in any programming language with the exception of actual machine language.

***source program**: A computer program written in a source language. Contrast with object program.

source statement library: A collection of books (such as macro definitions) cataloged in the system by the librarian program.

spanned records: Records of varying length that may be longer than the currently used blocksize, and which may therefore be written in one or more continuous blocks. A spanned record may occupy more than 1 track of a disk device.

spooling: The reading and writing of input and output streams on auxiliary storage devices, concurrently with job execution, in a format convenient for later processing or output operations.

stand-alone dump: A program that displays the contents of the registers and part of the real address area and that runs independently and is not controlled by DOS/VS.

standard label: A fixed-format identification record for a tape or disk file. Standard labels can be written and processed by DOS/VS.

storage protection: An arrangement for preventing access to storage

supervisor: A component of the control program. It consists of routines to control the functions of program loading, machine interruptions, external interruptions, operator communications and physical IOCS requests and interruptions. The supervisor alone operates in the privileged (supervisor) state. It is loaded by the IPL program and occupies the lowest area of real storage throughout system operation.

switched line: A communication line in which the connection between the computer and a remote station is established by dialing. Synonymous with *dial line*.

system directory list: A list containing directory entries of frequently-used phases and of all phases resident in the shared virtual area. This list is placed in the shared virtual area.

system residence device: The direct access device on which the system residence volume is located.

system residence volume: The volume on which the basic operating system and all related supervisor code is located.

task: A unit of work for the central processing unit from the standpoint of the control program.

teleprocessing: The processing of data that is received from or sent to remote locations by way of telecommunication lines.

terminal: (1)* A point in a system or communication network at which data can either enter or leave. (2) Any device capable of sending and receiving information over a communication channel.

throughput: The total volume of work performed by a computing system over a given period of time.

***track:** The portion of a moving storage medium, such as a drum, tape, or disk, that is accessible to a given reading head position.

transient area: An area in real storage used for temporary storage of transient routines.

UCS: Universal character set.

unit record: A card containing one complete record; a punched card.

universal character set: A printer feature that permits the use of a variety of character arrays. Abbreviated UCS.

unrecoverable error: A hardware error which cannot be recovered from by the normal retry procedures.

user label: An identification record for a tape or disk file; the format and contents are defined by the user, who must also write the necessary processing routines.

utility program: A problem program designed to perform a routine task, such as transcribing data from one storage device to another.

virtual address: An address that refers to virtual storage and must, therefore, be translated into a real storage address when it is used.

virtual address area: In DOS/VS, the area of virtual storage whose addresses are greater than the highest address of the real address area.

virtual mode: In DOS/VS, the mode of a program which may be paged.

virtual partition: In DOS/VS, a division of the virtual address area of virtual storage that may be allocated for programs that may be paged.

virtual storage: Addressable space that appears to the user as real storage, from which instructions and data are mapped into real storage locations. The size of virtual storage is limited by the addressing scheme of the computing system and by the amount of auxiliary storage available, rather than by the actual number of real storage locations.

Virtual Storage Access Method (VSAM): VSAM is an access method for direct or sequential processing of fixed and variable length records on direct access devices. The records in a VSAM file can be organized either in logical sequence by a key field (key sequence) or in the physical sequence in which they are written on the file (entry-sequence). A key sequenced file has an index, an entry-sequenced file does not.

volume: (1) That portion of a single unit of storage media which is accessible to a single read/write mechanism, for example, a drum, a disk pack, or part of a disk storage module. (2) A recording medium that is mounted and dismounted as a unit, for example, a reel of magnetic tape, a disk pack, a data cell.

VSAM access method services: A multifunction utility program that defines VSAM files and allocates space for them, converts indexed sequential files to key-sequenced files with indexes, facilitates data portability between operating systems, creates backup copies of files and indexes, helps make inaccessible files accessible, and lists file and catalog entries.

VSAM catalog: A key-sequenced file, with an index, containing extensive file and volume information that VSAM requires to locate files, to allocate and deallocate storage space, to verify the authorization of a program or operator to gain access to a file, and to accumulate usage statistics for files.

work file: A file on a secondary storage medium reserved for intermediate results during execution of the program.

Index

A

abnormal program termination 22
access method 55
  summary of 64
access method services (VSAM). 89
address area
  real 29
  virtual 29
address relocation 50
address space 29
address translation (see dynamic address translation)
allocating storage 30, 31
application program 10
assembler program 90
assembling a program 20
auxiliary storage 55

B

background partition 23
basic teleprocessing access method (BTAM) . 65
book 50
BTAM 65

C

catalog (VSAM) 61
cataloged procedure 50, 89
cataloging programs
  permanently 18
  temporarily 18
central processing unit
  models supported by DOS/VS 108, 7
COBOL compiler 104
command
  job control 71
  operator 71
communication
  operator-system. 71
compatibility 97
compiler
  RPG II 103
  FORTRAN 103
  COBOL 104
  PL/I 105
component program 99, 10
configuration
  configurations supported by DOS/VS . . . 107
control field 55
control interval (VSAM) 61
control program. 9
core image library 49, 18
CPU model
  models supported by DOS/VS 108
CPU capacity 108
CPU storage 29
CPU usage . . . .
  in single-partition system 25
  in multiprogramming system 27
CPU wait state 24

D

DAM (see direct access method)
DASD file protection 69
DAT (see dynamic address translation)
data integrity 67
data management 55
data management routine 10
data organization
  sequential 56
  indexed sequential 58
  direct 62
  virtual storage 60
data security 67
debugging aid 73
device assignment 14
  extended 81, 88
  permanent 16
  standard 16
  temporary 16
direct access device . . . .
  models supported by DOS/VS 111
direct access method (DAM) 62
disk operating system/virtual storage . .
  component programs of the 10
  summary of concepts of the 10
  System/370 CPU models supported . . 108, 7
display device
  models supported by DOS/VS 114
documentation (DOS/VS) 117, 119
DOS/VS (see disk operating system/virtual storage) . . . .
DOS/VS Assembler 90
dump 22
duplicate assignment (of I/O devices) . . . 69
dynamic address translation (DAT) 36

E

education 117
emulation on System/370 . . . . 75, 91
emulator 75, 91
entry-sequenced file . . 60
Environment Recording, Edit and Print program (EREP) 73
executable program 18, 49
executing a program 18, 31
  in real mode 31

- in virtual mode 31, 38
- performance considerations for 39

- exit routine 23

F

- file 55
- file label 68
- file organization
  - direct 62
  - indexed sequential 58
  - sequential 56
  - virtual storage 60
- file portability 62
- file protection 69
- file sharing 62
- foreground partition 25, 86
- FORTRAN compiler 103
- free space (VSAM) 61

G

- glossary 123

I

- index
  - for ISAM 58
  - for VSAM 60
- indexed sequential access method (ISAM) 58
- initial program loader (IPL) 9
- input queue (POWER) 43
- integrated emulator 75
- interactive terminal facility 106
- internal storage 29
- interval timer 90
- I/O interrupt 27
- ISAM (see indexed sequential access method)
- ISAM interface program 62

J

- job 13
- job accounting 46
- job control program 9, 18
- job control statement 13
  - examples of 19
- job step
  - single 14
  - multiple 14
- job stream 13

K

- key
  - DAM 63
  - ISAM 58
  - VSAM 60
- key-sequenced file 60

L

- label
  - tape 68
  - disk 68
  - processing 68
- language translator 103, 9
- language support
  - for data management (summary) 67
  - for VSAM 62
- librarian program 49
- library 49
  - core image 49
  - private 50
  - maintenance and service of 51
  - procedure 50
  - relocatable 50
  - source statement 50
  - system 49
- link-editing 20, 50
  - with relocating load 50, 52, 53
  - without relocating load 52, 53
- linkage editor 50
- LIOCS (see logical IOCS)
- logical IOCS 66
- logical unit 14
  - system 17
  - programmer 17

M

- magnetic tape unit
  - models supported by DOS/VS 109
- main storage 29
- main task 28
- maintenance and service of library 51
- manual control
  - models supported by DOS/VS 114
- manuals for DOS/VS 117
- message 71
- module 50
- multiprogramming 24
- multitasking 28

O

- object program 18, 50
- On-Line Test Executive Program (OLTEP) 73
- operator command 71
- operator-system communication 71
- output queue (POWER) 43
- overcommitting real storage 40
- overflow area 59

P

- page 34
- page data set 34
- page fault 34
- page frame 34
- page in operation 34
- page out operation 34
- page pool 34
- paging 38, 34
- partition

  - allocating storage to a 30, 33, 41
  - background 23
  - changing size of 24
  - foreground 25, 86
  - number of tasks in 28
  - priority of 25, 87
  - real 31
  - virtual 31
- performance
  - with POWER 45
  - with virtual storage 39
- permanent device assignment 16
- phase
  - relocatable 50
  - non-relocatable 50
- physical IOCS 66
- physical unit 14
- PIOCS (see physical IOCS)
- PL/I compiler 105
- PL/I library 106
- POWER 43, 95
- POWER RJE 44, 95
- PP (see program product)
- printer
  - models supported by DOS/VS 112
- priority (partition)
  - default priority 25
  - changing 25
- private library 50
- problem-program area 24
- procedure 50
- procedure library 50, 89
- processing program 9
- processor storage 29
- program
  - assembling a 20
  - cataloging a 18
  - executable 18, 49
  - executing a 18
  - link-editing a 20
  - object 18, 50
  - processing with POWER 44
  - self-relocating 51
  - terminating a 22
- program product 103
- program run mode
  - real 31
  - virtual 31
- program termination 22
  - abnormal 22
- programmer logical unit 17
- programming language
  - Assembler 90
  - COBOL 104
  - PL/I 105
  - RPG 103
  - FORTRAN 103
  - ITF-Basic 106
  - ITF-PL/I 106
- protection
  - file 69
  - storage 23
- publications 117
- protection macro 70
- punched card device
  - models supported by DOS/VS 110

Q

- queue (POWER)
  - input 43
  - output 43
- queued teleprocessing access method (QTAM) 65

R

- RAS 73
- real address area 29
- real mode 31
- real storage 29
- record 55
- Recovery Management Support Recorder (RMSR) 73
- Reliability, Availability, Serviceability (RAS) . 73
- Reliability Data Extractor (RDE) 73
- relocatable library 50
- relocating loader 50, 86
- relocation of addresses 50
- remote job entry 44, 95
- remote terminal 64
- resource protection macro 70
- retrieval of data
  - summary of retrieval methods 65
- RDE (see Reliability Data Extractor)
- RJE (see remote job entry)
- RMSR (see Recovery Management Support Recorder)
- RPG II compiler 103

S

- SAM (see sequential access method)
- SCP (see system control programming)
- security of data 67
- self-relocating program 51
- sequential access method (SAM) 56
- sequential data organization 56
- service program 9
- shared virtual area 81, 87
- shipment of DOS/VS 78
- single-partition system 23
- sort/merge program 106
- source statement library 50
- standard device assignment 16
- storage
  - allocation of 30, 31
  - auxiliary 55
  - CPU 29
  - fragmentation of 29
  - internal 29
  - main 29
  - management of 29

organization of 23
processor 29
real 29
virtual 29
storage allocation
in real address area 30
in virtual address area 30
storage fragmentation 29
storage management 29
storage organization 23
sublibrary 50
subtask 28
supervisor 9, 77
supervisor area 24
supervisor selection 82, 88
symbolic device name 14
list of 17
SYSCAT 17
SYSCLB 17
SYSIN 17
SYSIPT 17
SYSLNK 17
SYSLOG 17
SYSLST 17
SYSOUT 17
SYSPCH 17
SYSRDR 17
SYSRES 17
SYSRLB 17
SYSSLB 17
system action 71
system configuration (see configuration)
system control programming 101
system generation 77
system library 49
system logical unit 17
system-operator communication 71
SYSVIS 17
SYS000...SYSnnn 17

T

task
main 28
sub 28
teleprocessing
access method 64
with POWER 44
temporary device assignment 16
terminal device
models supported by DOS/VS 113
track hold function 70

U

unit-record device 43
user exit routine 22
utility program 101

V

virtual address area 29
virtual mode 31, 36
virtual storage 29
virtual storage access method (VSAM) . . 60, 93
virtual storage support 29, 83
volume portability 62
VSAM (see virtual storage access method)
VSTAB supervisor macro 85

W

wait state 24

Introduction to DOS/VS

GC33-5370-2

READER'S COMMENT FORM

This sheet is for comments and suggestions about this manual. We would appreciate *your* views, favorable or unfavorable, in order to aid us in improving *this* publication. This form will be sent directly to the author's department. Please include your name and address if you wish a reply. Contact your IBM branch office for answers to technical questions about the system or when requesting additional publications. Thank you.

Your comments* and suggestions:

*** We would especially appreciate your comments on any of the following topics:**

Clarity of the text	Accuracy	Index	Illustrations	Appearance	Paper
Organization of the text	Cross-references	Tables	Examples	Printing	Binding

GC33-5370-2

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

CUT ALONG THIS LINE

Fold   Fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
1133 Westchester Avenue
White Plains, N.Y. 10604

Attention: Department 813 U

Fold   Fold

IBM

**International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)**

**IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)**

GC33-5370-2

Introduction to DOS/VS (S370 - 20) Printed in U.S.A. GC33-5370-2

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

GC33-5370-2

Introduction to DOS/VS (S370 - 20) Printed in U.S.A. GC33-5370-2

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)