Introduction to DOS/VS

Release 34

GT00-0474-0 (formerly GC33-5370-5)
File No. S370-20

Systems

Introduction to DOS/VS

Release 34

IBM

Sixth Edition (April, 1977)

This is a major revision of, and obsoletes, GC33-5370-4.

For a summary of amendments, refer to Part 3 of this manual under *New in Recent Releases* which documents the amendments for Release 32, 33, and 34 of DOS/VS. This edition also contains a number of maintenance changes.

All changes and additions to the text are indicated by a vertical line to the left of the change.

This edition applies to Version 5, Release 34, 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 latest *IBM System/370 Bibliography,* GC20-0001, for the editions that are applicable and current.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

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

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/VS 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 Different about DOS/VS?** summarizes the features of DOS/VS that differ from those of DOS. Furthermore, the new features introduced with the last three releases of DOS/VS are highlighted. This part of the manual is intended primarily for users already familiar with the system.

Part 4: **Licensed IBM Programs for DOS/VS** provides brief descriptions of licensed IBM programs available with DOS/VS through a license agreement.

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,* together with *IBM System/360 Principles of Operation.*

Table of Contents

List of Figures

Part 1. What is the Disk Operating System?

DOS/VS (Disk Operating System/Virtual Storage) is a comprehensive collection of program components 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.

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

DOS/VS operates on central processing units (CPUs) of System/370 Models 115 through 158. Detailed information about the supported CPUs as well as the supported I/O devices is given in Part 5 of this manual.

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/VS, 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.

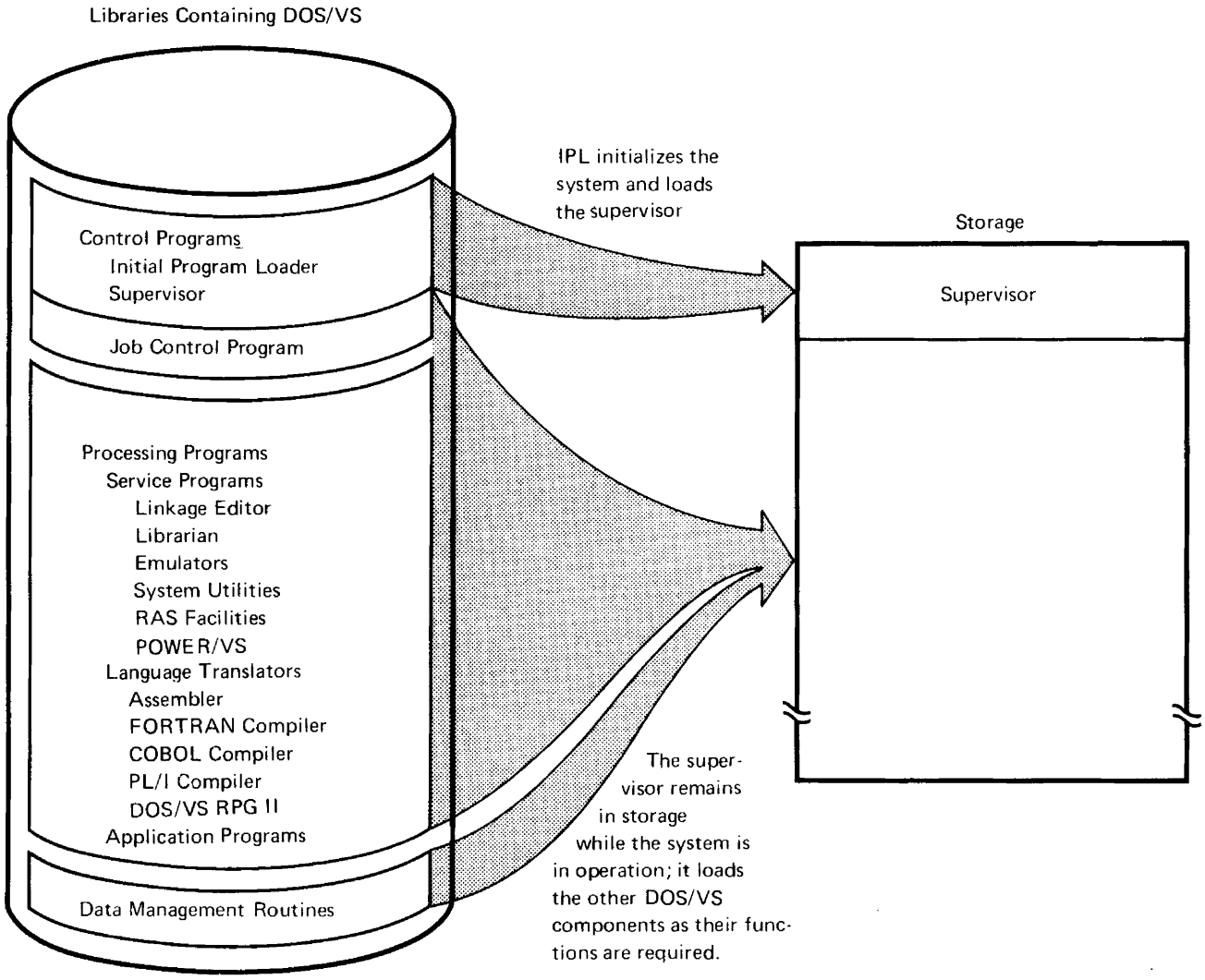


Figure 1.1. DOS/VS Component Programs

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 complex computing system, DOS/VS provides a number of routines for the detection, analysis, and recovery of machine and system malfunctions.

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

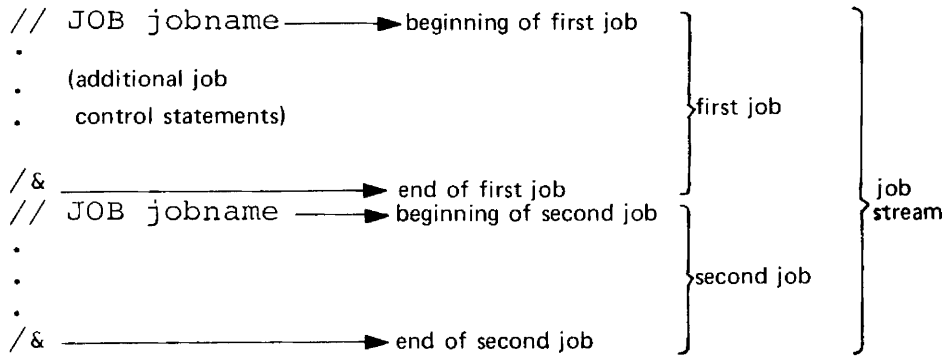
A job and the environment in which it is to run must be defined to the system by means of job control statements. Job control statements specify, for example, whether user programs are to be compiled, link-edited, and/or executed, from which library or device the system or user programs are to be loaded,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

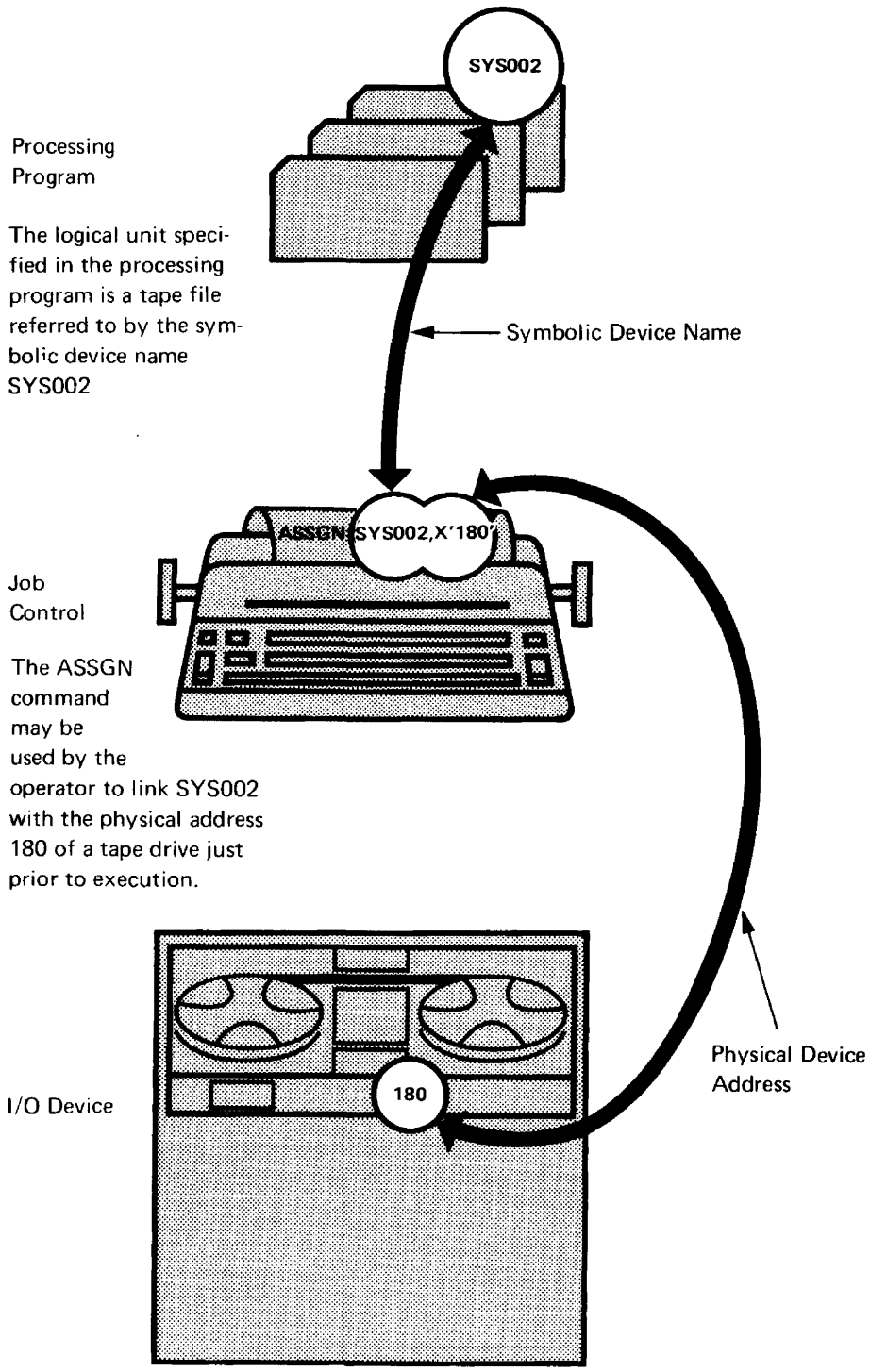


Figure 2.1. Assigning an Actual I/O Device to a Symbolic Device Name by the Operator

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 device-type, device-class, or address-list are used in assignments, the job control statements are to a large extent independent of the configuration used.
- 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.	communication between the system and the operator and for logging job control statements. It can also be assigned to a printer.
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 master 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, and as hard copy file
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.

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.

Programs that can be shared between partitions and that are used frequently may be loaded into the shared virtual area (SVA) at IPL time. More information on the SVA and the requirements of the programs that can be contained in it can be found in the section *Allocating Storage to Partitions.*

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 completion of the previous job. In the case of a background partition (described under *Storage Organization*), the job control program is automatically loaded after IPL. In the case of a foreground partition, the partition must be activated by using the BATCH or START commands.
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.

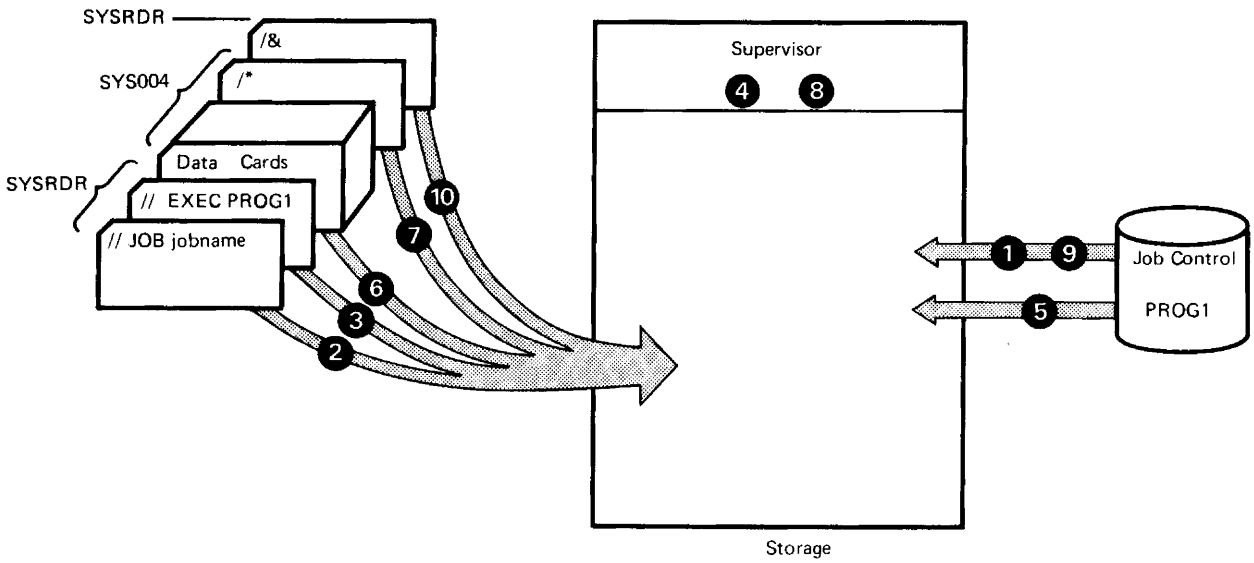


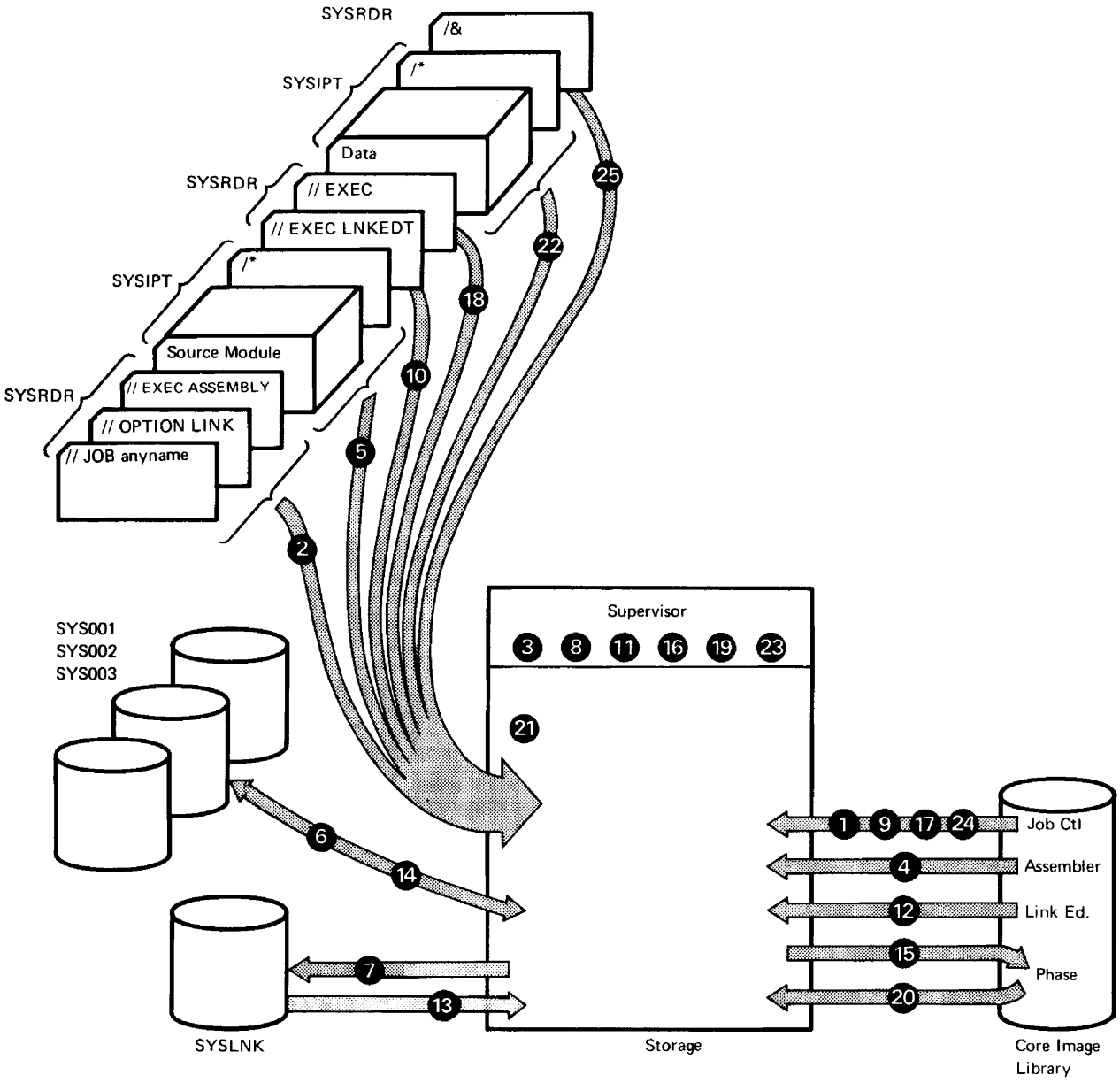
Figure 2.3. Loading and Executing from the Core Image Library (Part 2)

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 (9).
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 POWER/VS.

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 high flexibility 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.

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 POWER/VS.

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 high flexibility 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.

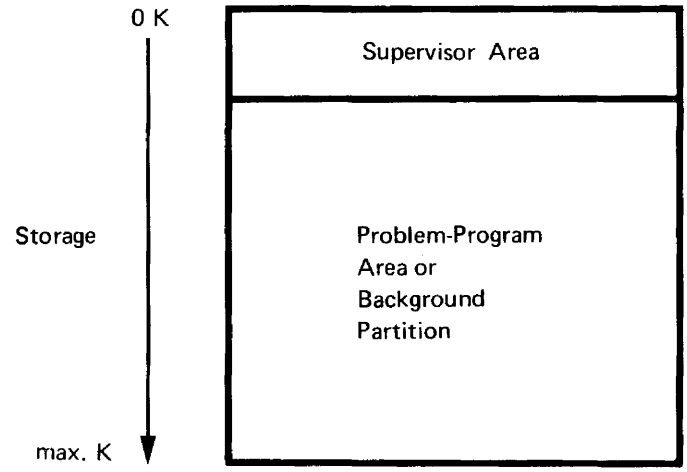


Figure 2.5. Single-Partition Storage Organization

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.)

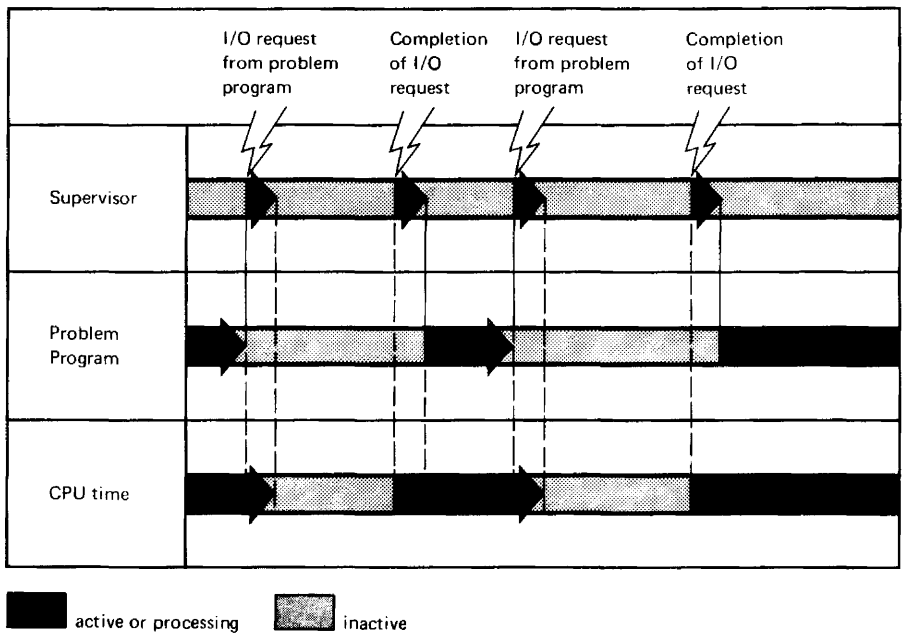


Figure 2.6. CPU Usage in a Single-Partition System

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).

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 background (virtual) 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 (see Figure 2.7) for the partitions are, from low to high, in the order of their position relative to 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.

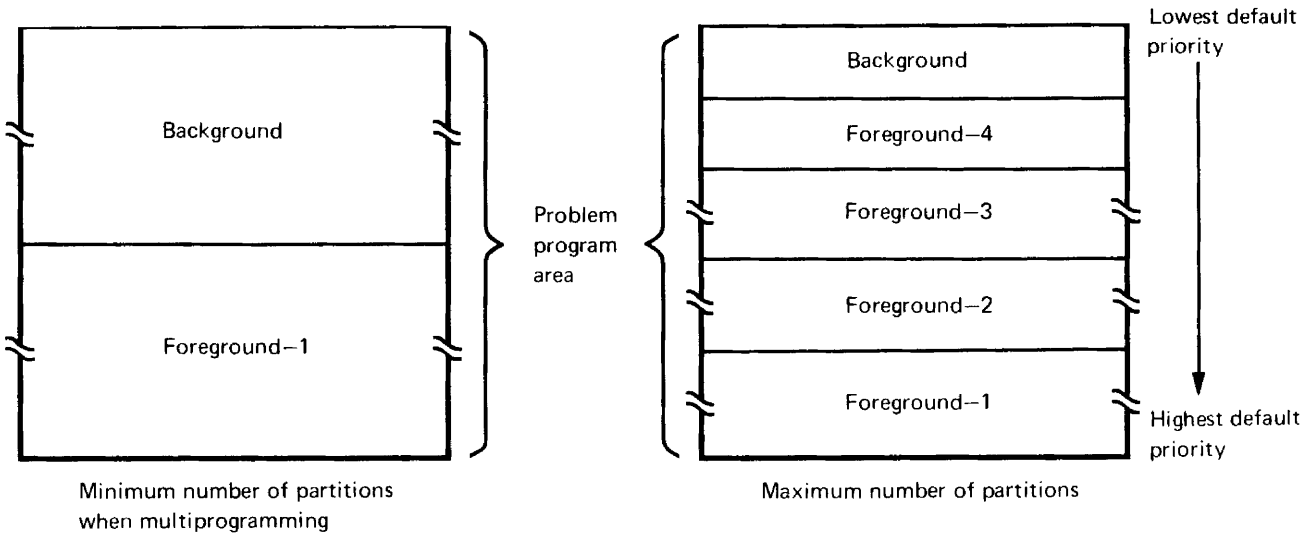


Figure 2.7. Default Priorities in a Multiprogramming Environment
The default priorities assigned by the system may be changed through an operator command.

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 (refer to Figure 2.6).

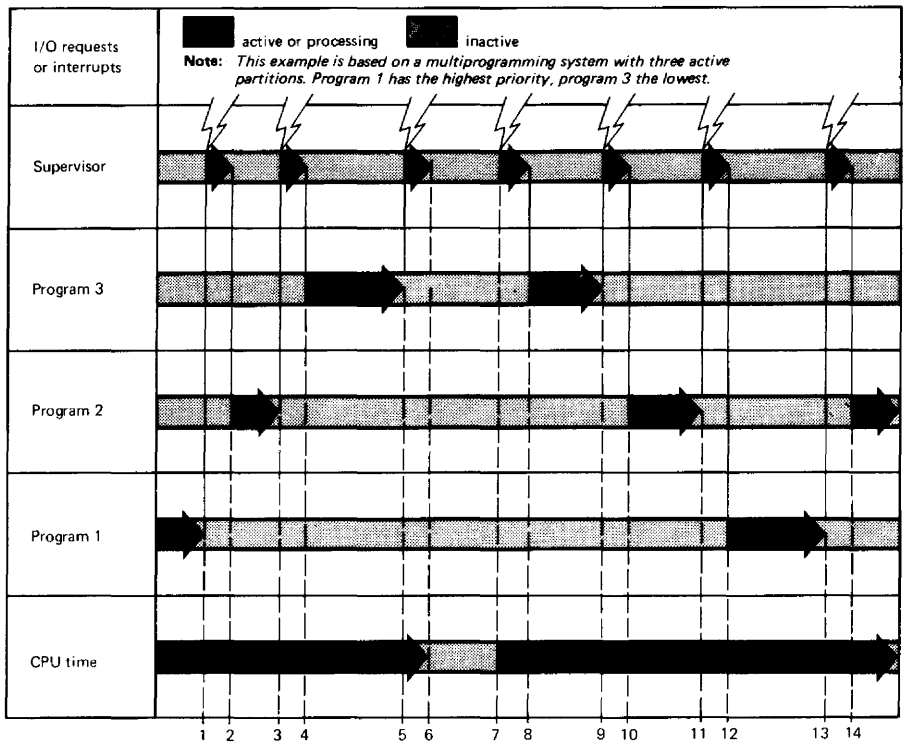


Figure 2.8. CPU Usage in a Multiprogramming System

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 priority of a subtask in the partition is normally determined by the sequence in which it is attached by the main task. The subtask attached first has the highest priority, and so on, unless the priorities are modified by means of a system macro within the user's program.

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

The concept of virtual storage as implemented in DOS/VS is the most important improvement on the features available in DOS.

Users of DOS are restricted to an address space limited by the storage physically contained in the computer system. The total space required by the supervisor and assigned partition(s) cannot exceed this contraint;

programs are confined to the **real storage** partition limits. Also, because the size of a partition depends on the size of the largest program to be executed in it, running any smaller program in such a partition will result in some storage space being unused. DOS/VS removes these limitations of DOS.

The concept and implementation of virtual storage support are described below.

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, the size of the SVA (shared virtual area), 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 and the SVA. 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.

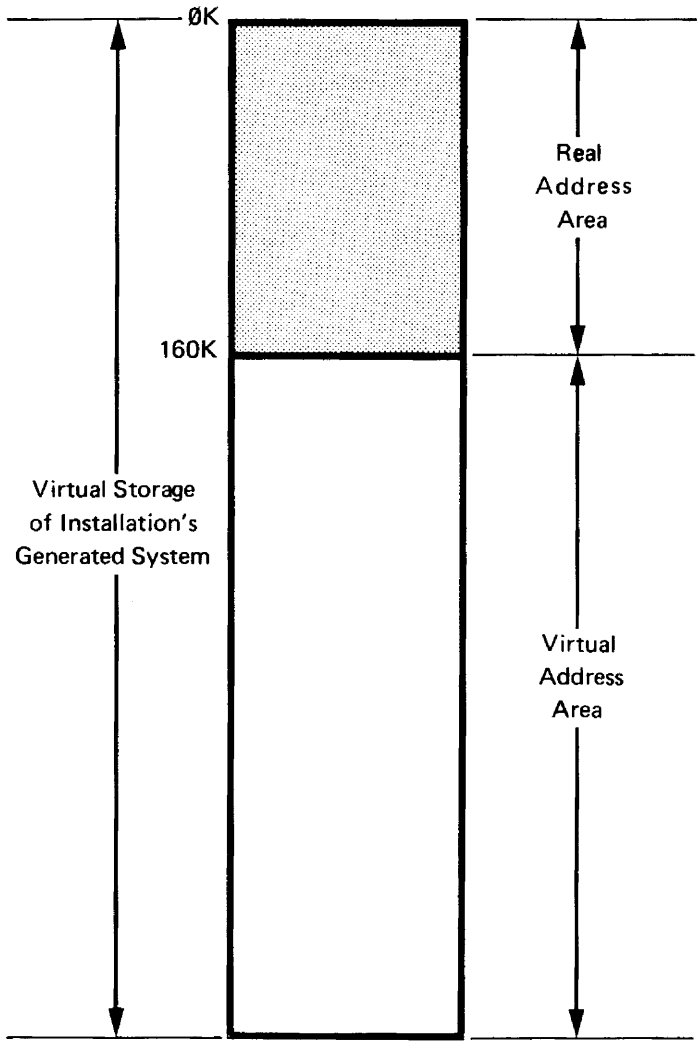


Figure 2.9. Virtual Storage

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.

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. 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 and the highest addresses are allocated to the SVA, which must be at least 64K.

(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.)

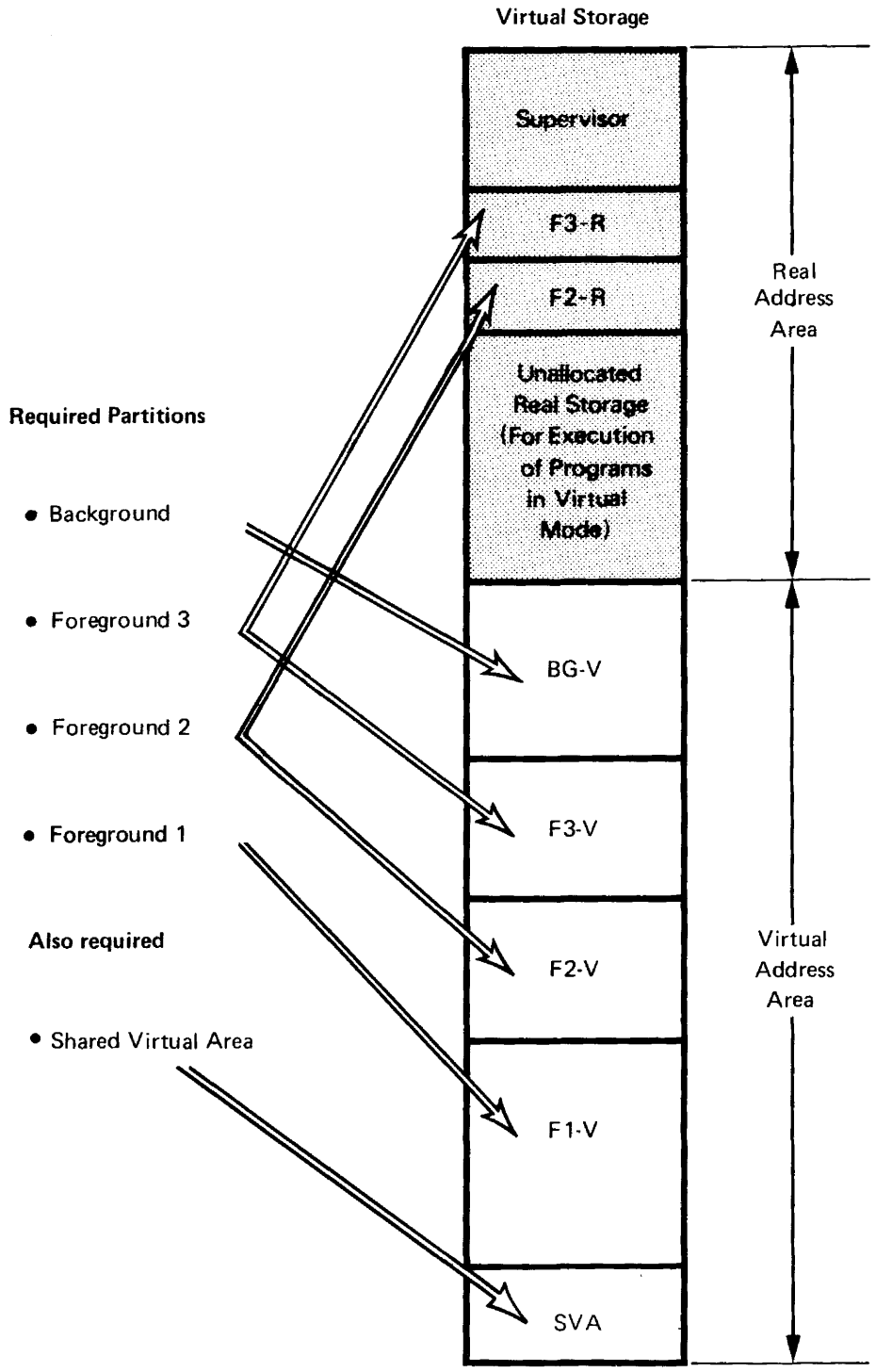


Figure 2.10. An Example of Storage Allocation

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:

- The main page pool, that is, real storage not occupied by the supervisor plus any real storage not allocated to partitions. The main page pool must be at least 18K bytes unless all programs are running in real mode. If all programs run in real mode, the main page pool 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.

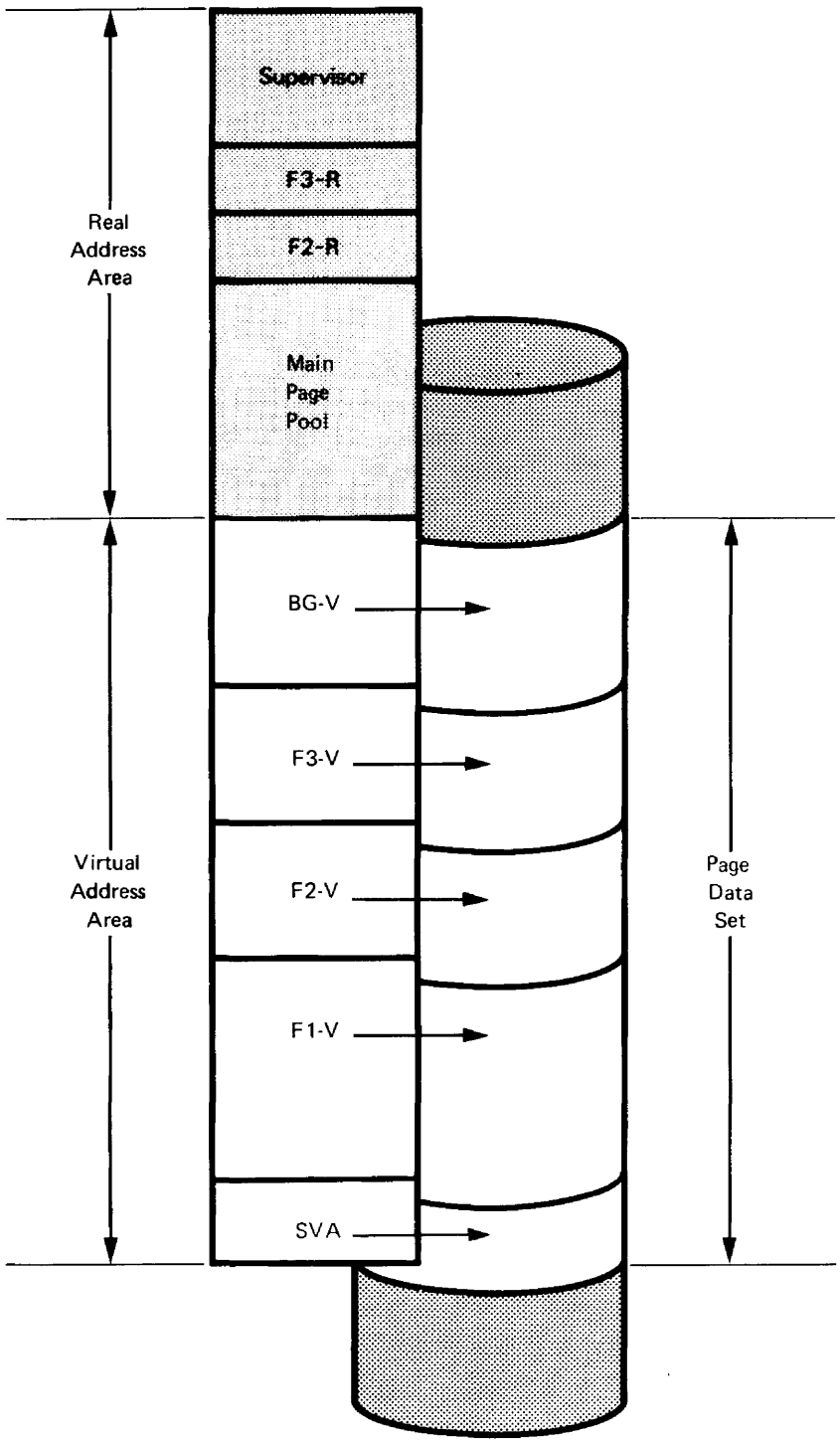


Figure 2.11. Page Data Set

The capacity of the page data set is equivalent to the size of the system's virtual address area. Programs in the virtual partitions will actually be executed in the page pool which consists of the main page pool and any available address space of allocated real partitions.

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 an I/O operation. There are other parts of some programs 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 fixes as soon as possible, to make the page frames available to all programs running in virtual mode. The user should do the same.

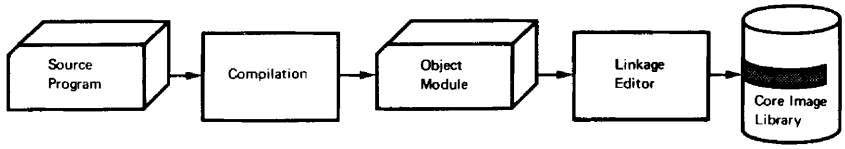
dynamic address translation

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 takes place **dynamically** during program **execution.** This is accomplished by the dynamic address translation hardware feature which is a basic part of System/370 design.

virtual mode execution

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 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 reloaded from the core image library but 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 page to be replaced 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.)

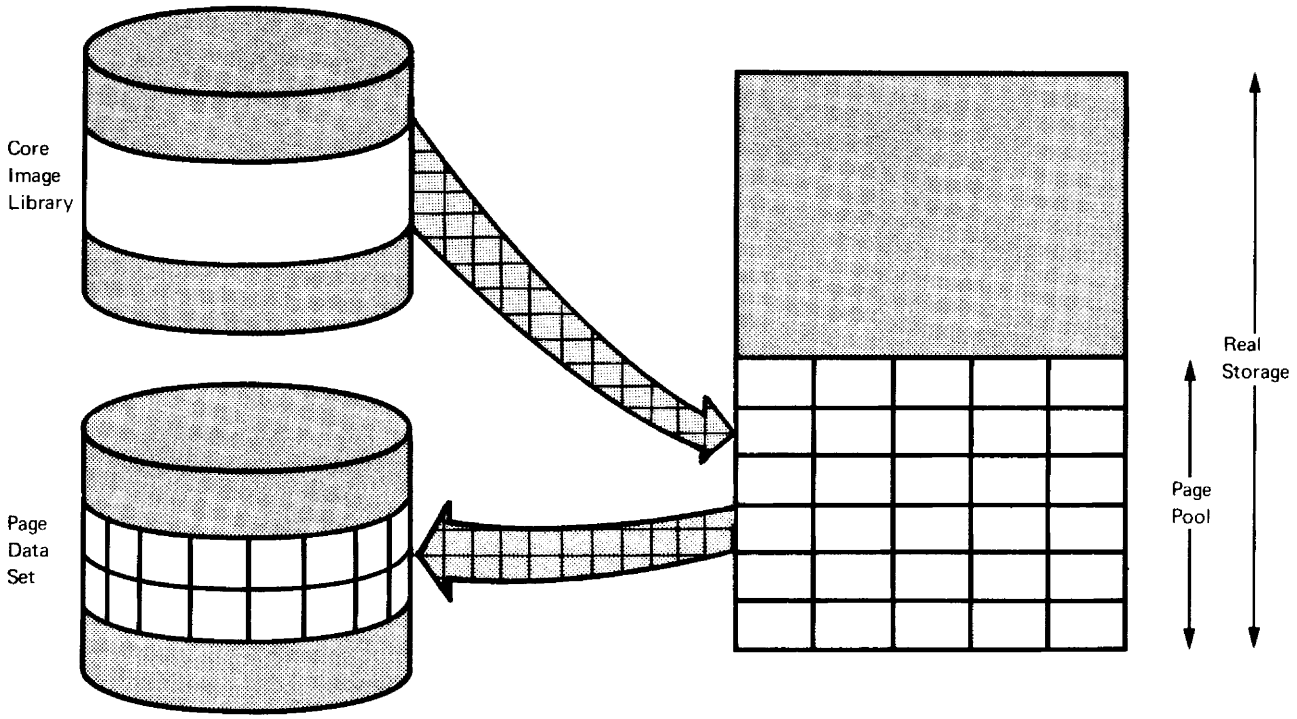


Figure 2.12. Loading a Program for Execution in Virtual Mode

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

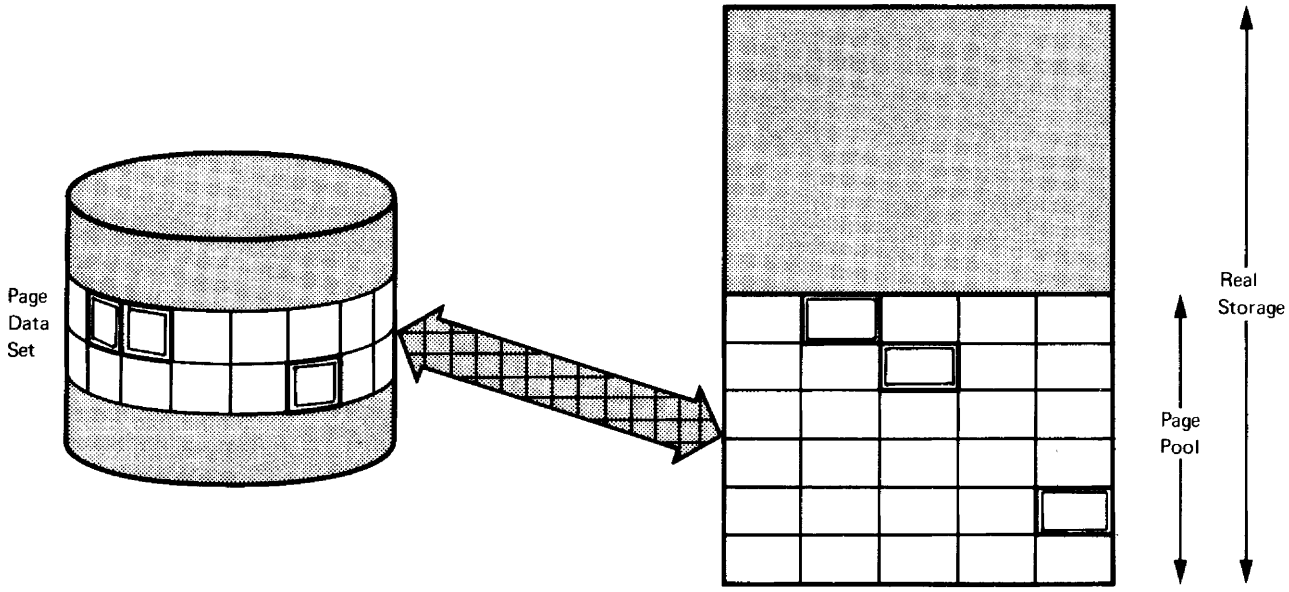


Figure 2.13. Paging between Real Storage and the Page Data Set

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 tables stored in the DOS/VS supervisor.
- All page frames in the page pool are available to any of the programs currently executing in 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 in page frames occupied by code no longer required.
- Four programs executing concurrently in virtual mode might be represented as shown in Figure 2.14.

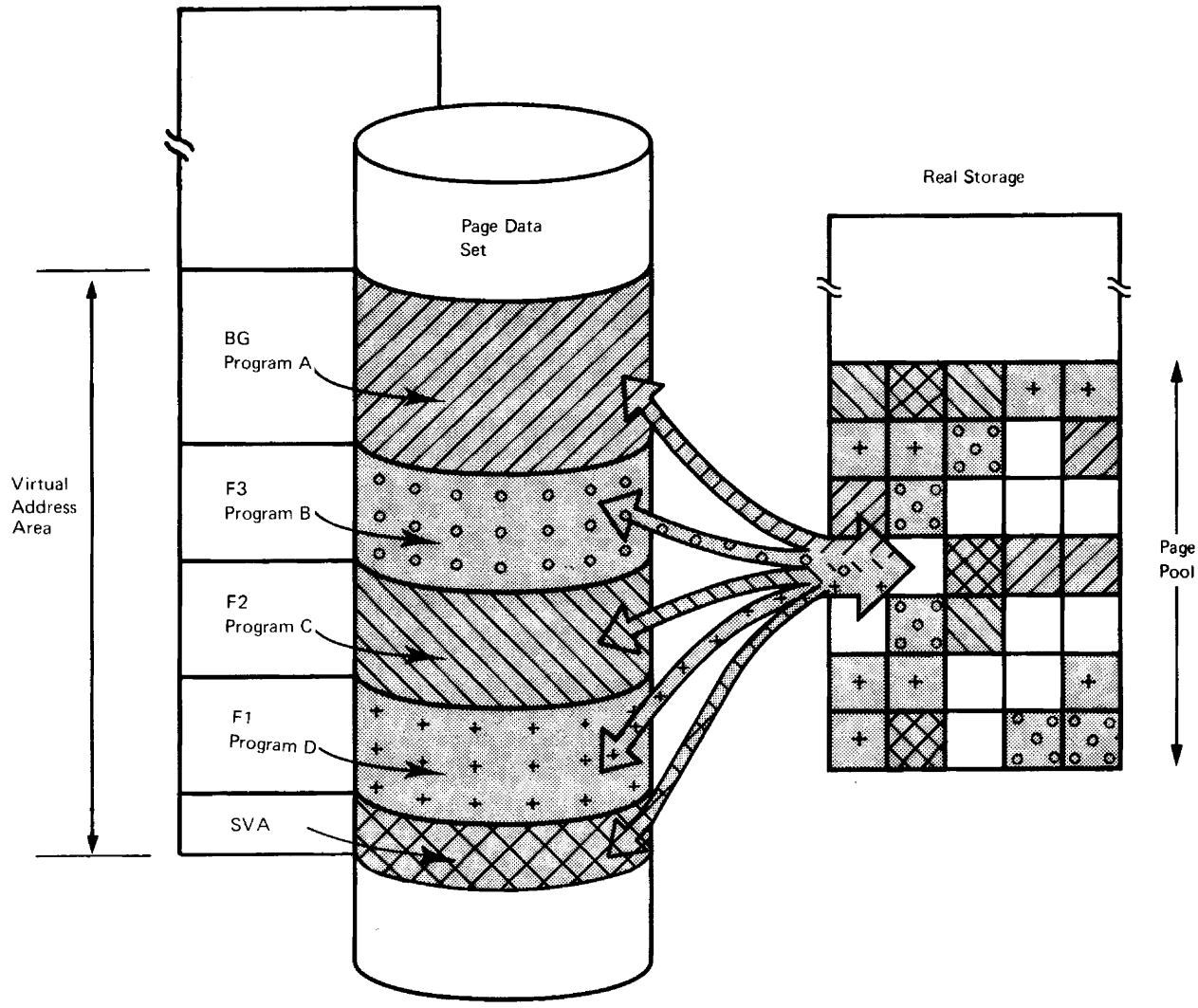


Figure 2.14. Four Programs Executing in Virtual Mode

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

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 302K 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 accommodate virtual partitions which the user knows will exceed the page pool size. He therefore has the option 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.

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	54 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	Virtual	POWER/VS without Remote Job Entry (also requires allocation of associated real partition. Unused page frames in the real partition are made available to the page pool.)	24 K	162 K
SVA	Virtual	Shareable programs		64 K
	Virtual	VSAM		302 K
Total Allocated to Supervisor and Real Partitions.			78 K	
Total Allocated to Virtual Partitions and SVA.				944 K
Main page pool (not explicitly allocated).			72 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.

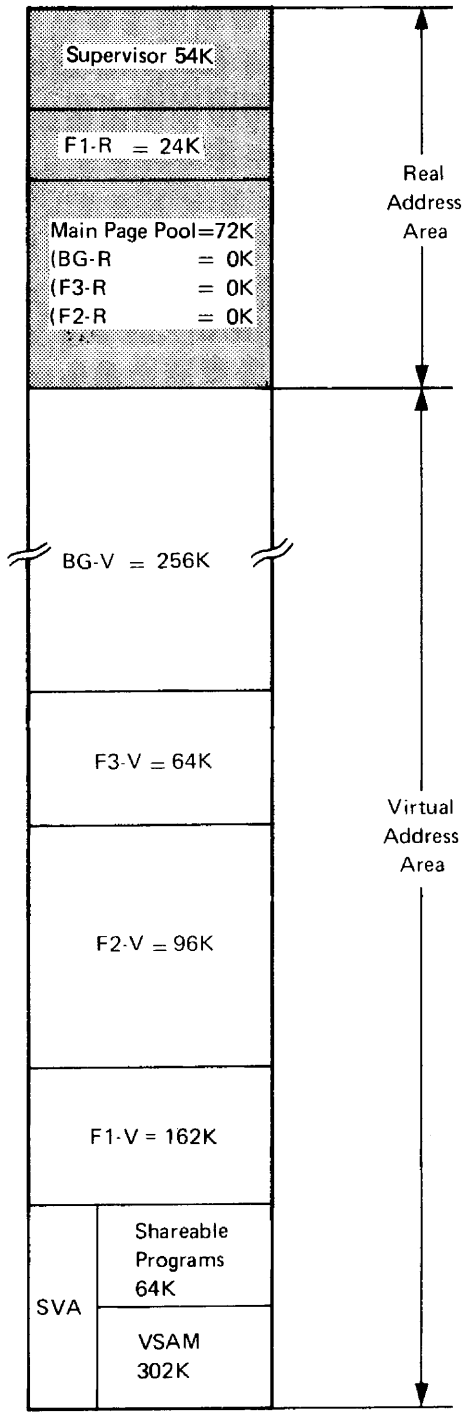


Figure 2.16. Storage Allocation Example

POWER/VS

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/VS, an optional service program designed for virtual storage to reduce CPU dependence on the relatively slow speeds of card readers, punches, and printers.

POWER/VS 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/VS in parallel during the execution of other jobs. The additional CPU time used by POWER/VS is negligible. In a typical environment of jobs with mixed characteristics, throughput may be substantially improved.

POWER/VS requires one of the generated virtual paritions and allows the user to execute programs in up to the maximum number of partitions minus one without the need for separate unit-record devices for each partition. The unit record devices used by POWER/VS can provide the I/O requirements for all the partitions that are being serviced by POWER/VS.

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

- Input. POWER/VS reads the job streams (job control statements, programs, and data cards) for the individual partitons and stores these in input queues on disk. The input may be entered from:
  - A local card reader or diskette device.
  - A remote BSC terminal.
  - A remote SNA (SDLC) work station with a console and an input device (card reader or 80-column card image device).
  - An outside partition not controlled by POWER/VS.
- Execution. From disk, the jobs are transferred by POWER/VS to the designated POWER/VS-controlled partitions and executed.
- Output. Unit-record output (printer and punch) of every job is stored on disk (or tape) by POWER/VS before it is finally processed as output:
  - For a local printer or punch device.
  - For a remote BSC terminal.
  - For a remote SNA (SDLC) work station where the output can either be produced by an appropriate work station unit-record device or where it can be temporarily stored on disk or diskette. In the latter case, the user can perform remote spooling of his job output.
  - For an outside partition not controlled by POWER/VS.
- Control. Throughout the different steps of input, execution, and output, the jobs running under POWER/VS are within the management of the user through the following command language facilities:

- Job entry control language (JECL). Along with his input job decks, the user may insert JECL commands to describe individual job execution or I/O requirements.
- Central operator commands. From input until the final list and punch output, the central operator may discharge control functions such as starting and stopping job execution or displaying or altering execution and I/O performance characteristics.
- Remote terminal operator commands. The remote terminal operator may perform similar functions as above for all jobs submitted by him or routed to him.
- Cross-partition communication macros. These macros enable a problem program running outside of POWER/VS control to supervise the execution of POWER/VS jobs in a way similar to POWER/VS commands or JECL.

Job input as well as job output may be held in the POWER/VS 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.

Turnaround time for jobs with extensive printed or punched output can be improved by segmenting the output, which means that parts of the output are printed or punched before the entire job is finished. The user can request output segmentation at POWER/VS generation, in the job stream, or in his problem program.

Whenever a unit-record input or output device 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.

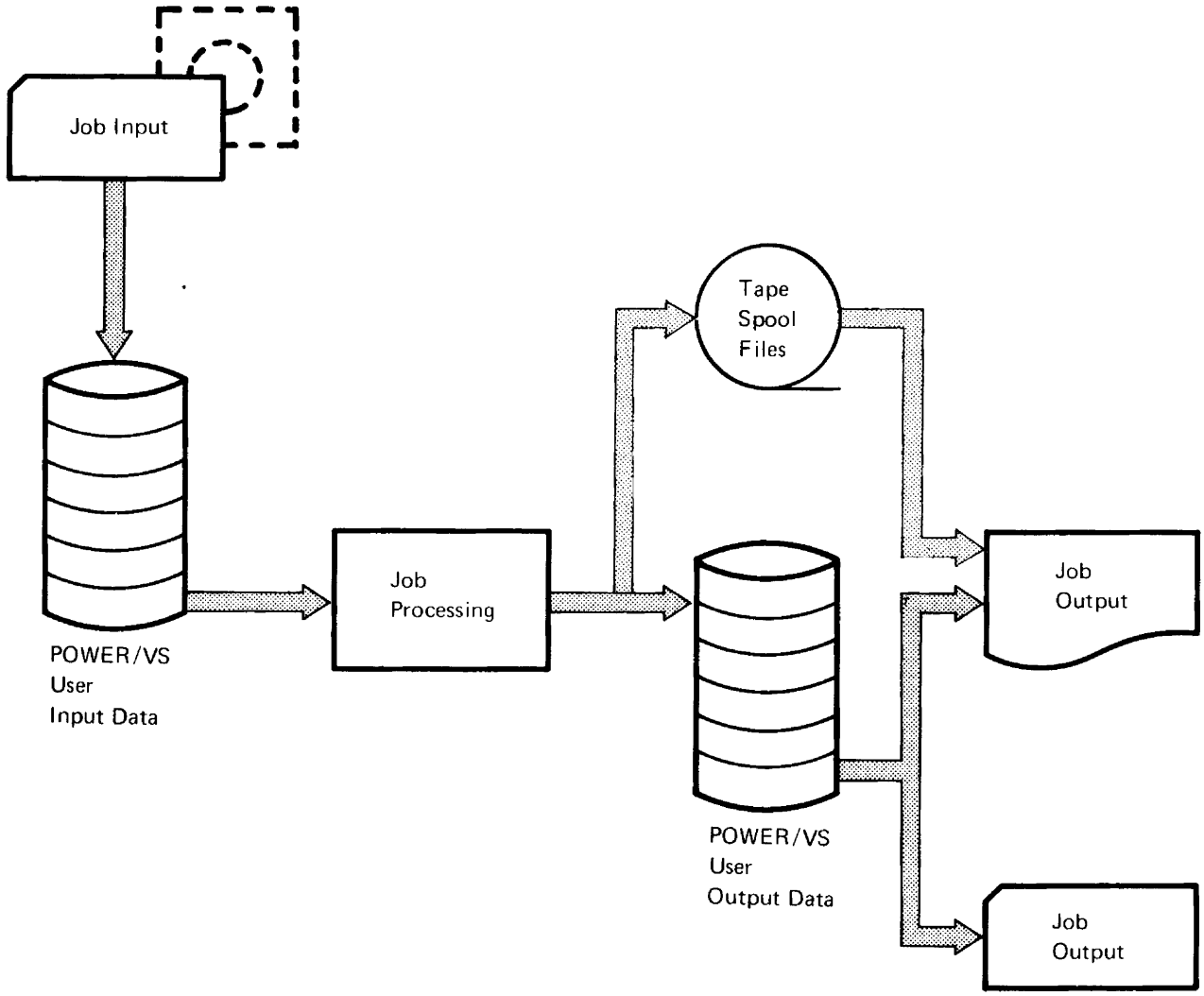


Figure 2.17. Processing with POWER/VS

A job's normal punched-card input is read from the card reader or from the diskette and queued on disk before the start of a job. Similarly, a job's output is stored on disk (or tape) in the output queue and printed and punched at a later stage. Punch output can also be directed back to the input queue.

Performance with POWER/VS

The performance of DOS/VS with POWER/VS 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/VS 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/VS. 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/VS, 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.

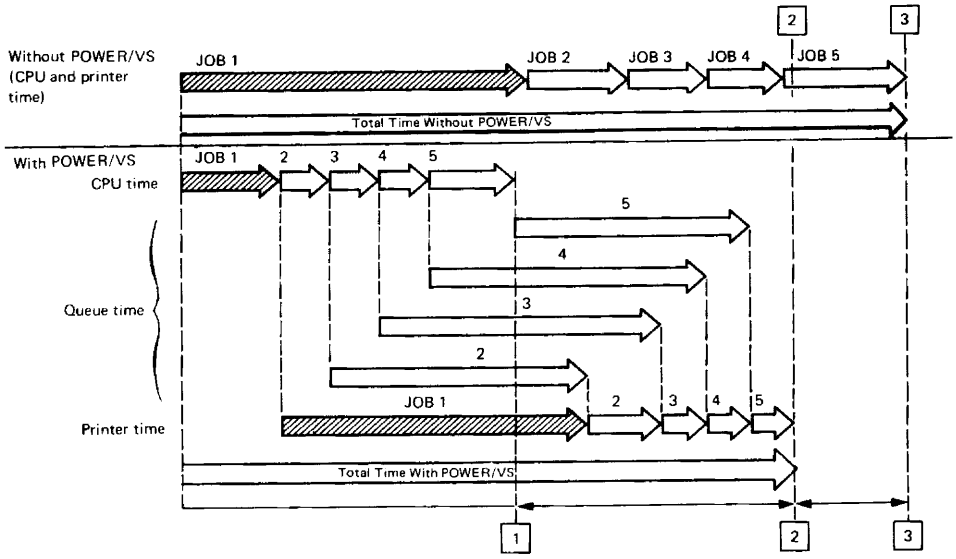


Figure 2.18. Processing Five Jobs with and without POWER/VS

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

Practical Considerations for Using POWER/VS

POWER/VS is distributed by IBM in the core image library as executable phases. This version of POWER/VS will suit the needs of many users; however, the user who has special requirements can tailor POWER/VS more to his own installation by assembling the POWER/VS generation macros, which are distributed in the source statement library. Subsequently, he should catalog his version(s) of POWER/VS into a core image library.

POWER/VS always runs in virtual mode. Its partition must have higher priority than the partitions it services. If VTAM is being used in a POWER/VS-controlled system, the VTAM partition must have a higher priority than the POWER/VS partition.

The user has the option to store POWER/VS output data on tape instead of on disk.

Remote Job Entry

POWER/VS also offers a teleprocessing facility, the Remote Job Entry (RJE). With POWER/VS RJE, jobs may be submitted from remote terminals. Once a job has been entered into the input job queue, the 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. RJE job output may be directed to the terminal from which the job was entered, to other terminals, or to the local output unit of the system.

RJE jobs may be submitted from terminals as follows:

- 2770, 2780, 3741, or 3780, all using Binary Synchronous Control (BSC).
- 377x in 2770/3780 compatibility mode, using Binary Synchronous Control (BSC).
- Certain 377x terminals and the 3790 Communication System with the RJE Facility (SNA), using Synchronous Data Link Control (SDLC).

POWER/VS RJE,BSC supports up to 25 BSC terminals on the same number of leased or dial-up lines concurrently; in non-concurrent (switched) operation, any number of BSC-controlled terminals may be attached. POWER/VS RJE,SNA allows up to 200 SNA work stations to be active at the same time. Mixed configurations of BSC and SNA terminals are possible.

Additional Major POWER/VS Facilities

Easy POWER/VS generation and start-up procedures include automatic start-up of POWER/VS via control cards or a diskette file.

Jobs entered into the system are grouped into user-assigned input classes. Within each class, jobs may be assigned different priorities for execution. The operator may call for execution of jobs of an individual class or of a group of up to four classes.

Job input may be retained after job execution to allow for repeated execution of the same job.

List and punch output is grouped into output classes. The list or punch output class of a job may be different from the input class assigned to this job. The operator may call for list or punch output of an individual output class or of a group of up to four classes.

Job output may be retained after printing or punching has been performed to allow production of further output copies at a later time.

Two or more copies of printed or punched output may be be requested during output scheduling.

List or punch output from successive jobs may be separated by internally generated list separator pages or punch separation cards.

Up to eight logical printers, eight logical punches, and one logical reader may be associated with each partition controlled by POWER/VS when the

partition is started. Hence, concurrent multiple printer and punch output can be handled.

Remote job entry (RJE) using BSC and/or SNA terminals also includes support for ASCII as transmission code.

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.

job accounting under POWER/VS

POWER/VS also has accounting facilities, which are a POWER/VS generation option. If this option is selected and POWER/VS is active, DOS/VS job accounting interface information and POWER/VS job accounting information are combined in the POWER/VS account file for each partition running under POWER/VS. The user does not need to write his own data collection routine. The user does, however, need to process the account file, sorting and summarizing the records to suit his particular accounting requirements. Details on the POWER/VS accounting facility are contained in the *POWER/VS Installation Guide and Reference*, GC33-6048.

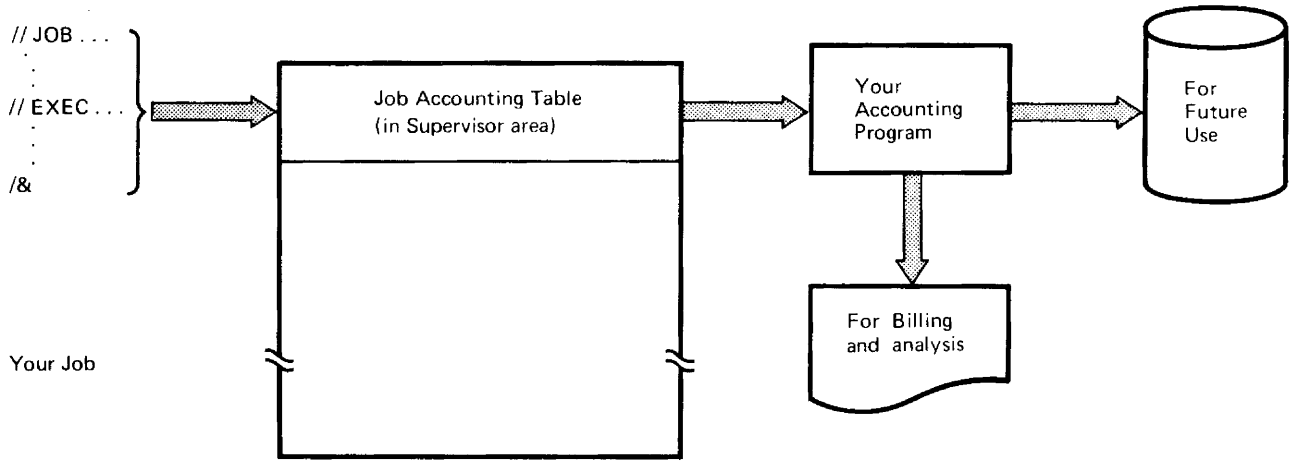


Figure 2.19. Job Accounting Procedure

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

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.

The DOS/VS librarian program provides service, organization, 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.

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 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.

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

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. It may be punched on SYSPCH and then cataloged into the relocatable library, and/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.

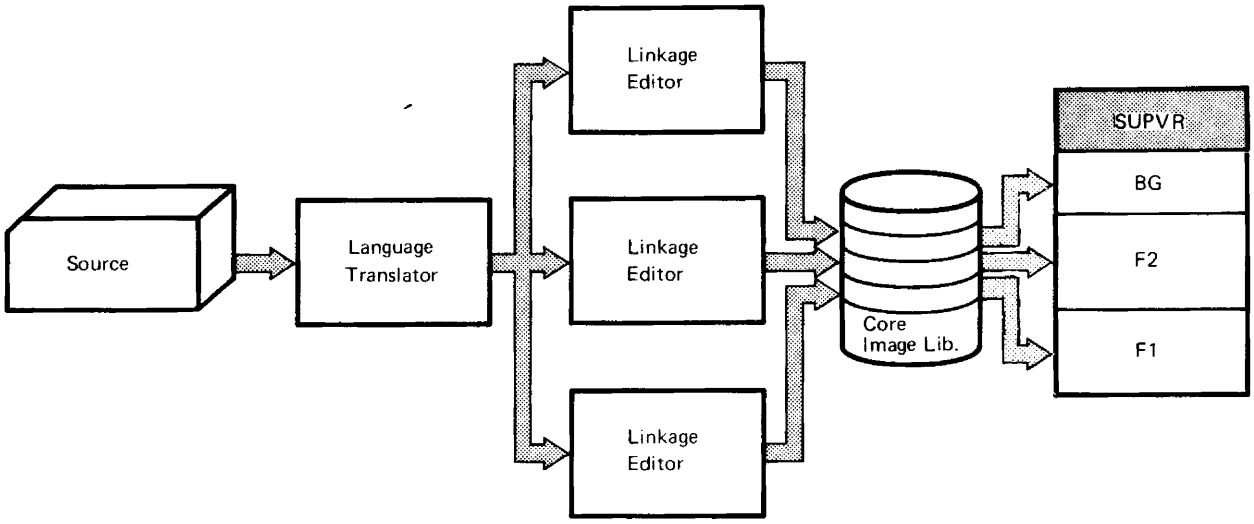
The relocating loader is a standard feature of DOS/VS. The feature can be excluded from the supervisor by specifying RELLDR=NO in the FOPT macro. If the relocating support is excluded and it is desired to use a non-relocatable phase in more than one partition, a separate copy of the phase must be available for each partition in which it is used.

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.

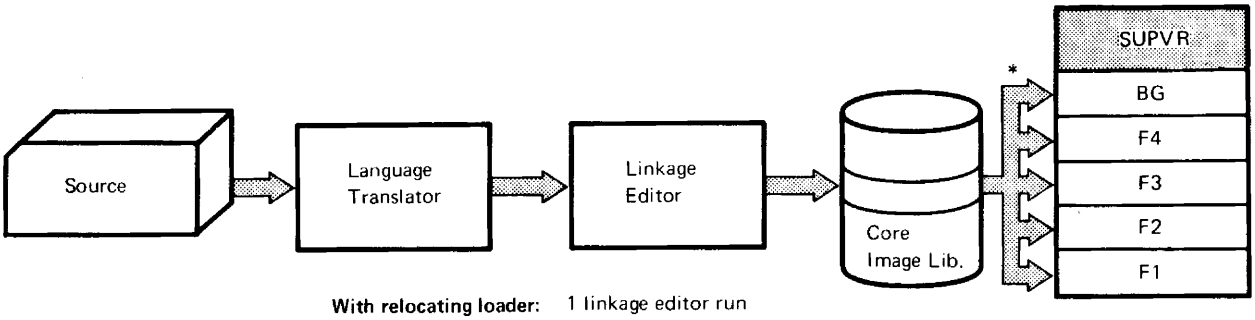
Librarian Programs

DOS/VS contains librarian programs that perform maintenance and service functions for all libraries. These functions include:

- Cataloging, renaming, and deleting of any element in a library.
- Printing of any element or directory.
- Punching of any element.
- Copying all elements from one library to another library of the same type, or copying only those elements that do not yet exist in the destination library.
- Condensing and changing the size and location of the libraries.
- Creating a new system disk pack and creating private core image, relocatable, and source statement libraries.



Without relocating loader: 3 linkage editor runs
3 copies in core image library



With relocating loader: 1 linkage editor run
1 copy in core image library

Figure 2.20. Linkage Editing with and without Relocating Loader

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 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.

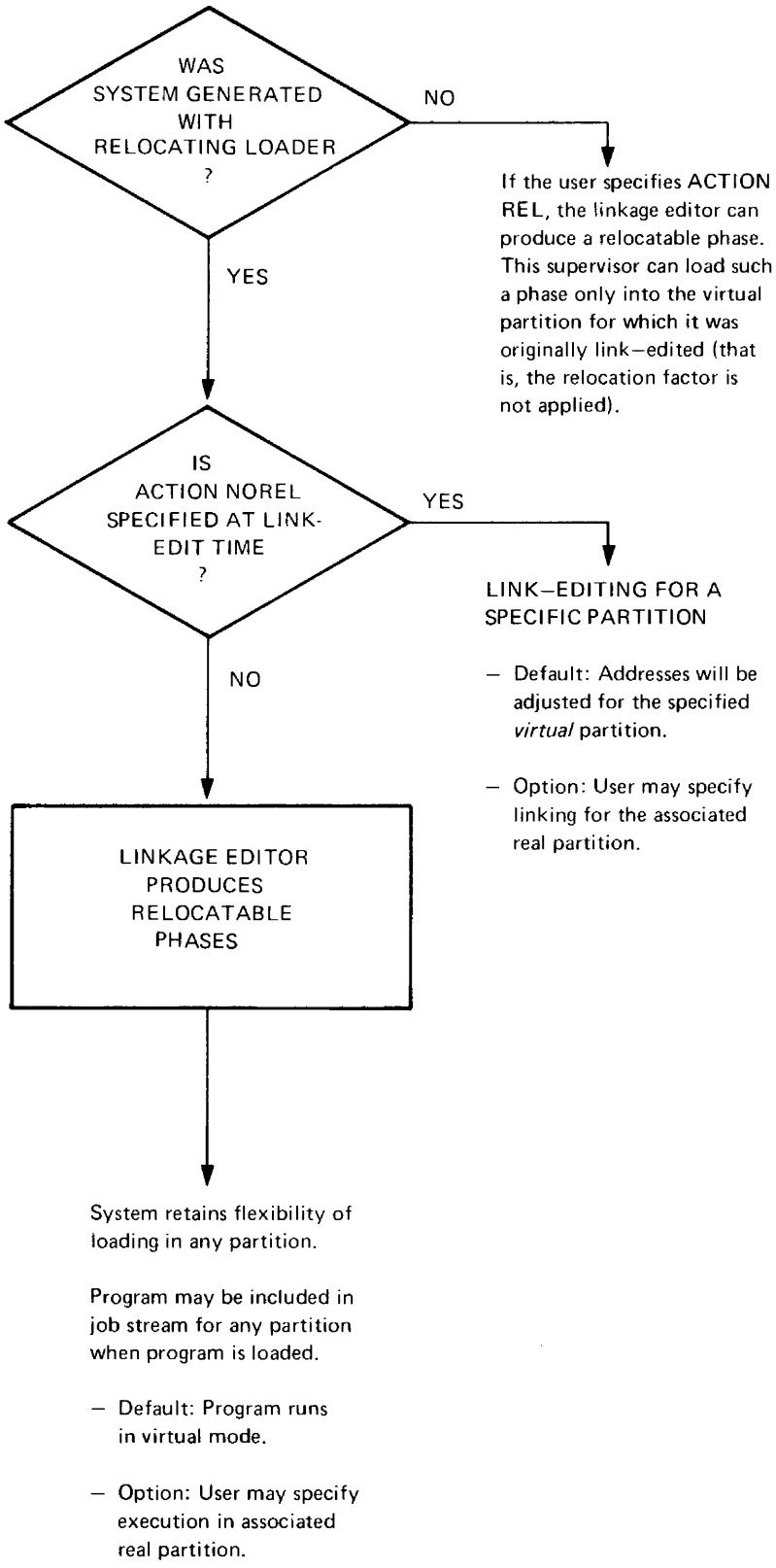


Figure 2.21. Options Available during Link-Editing

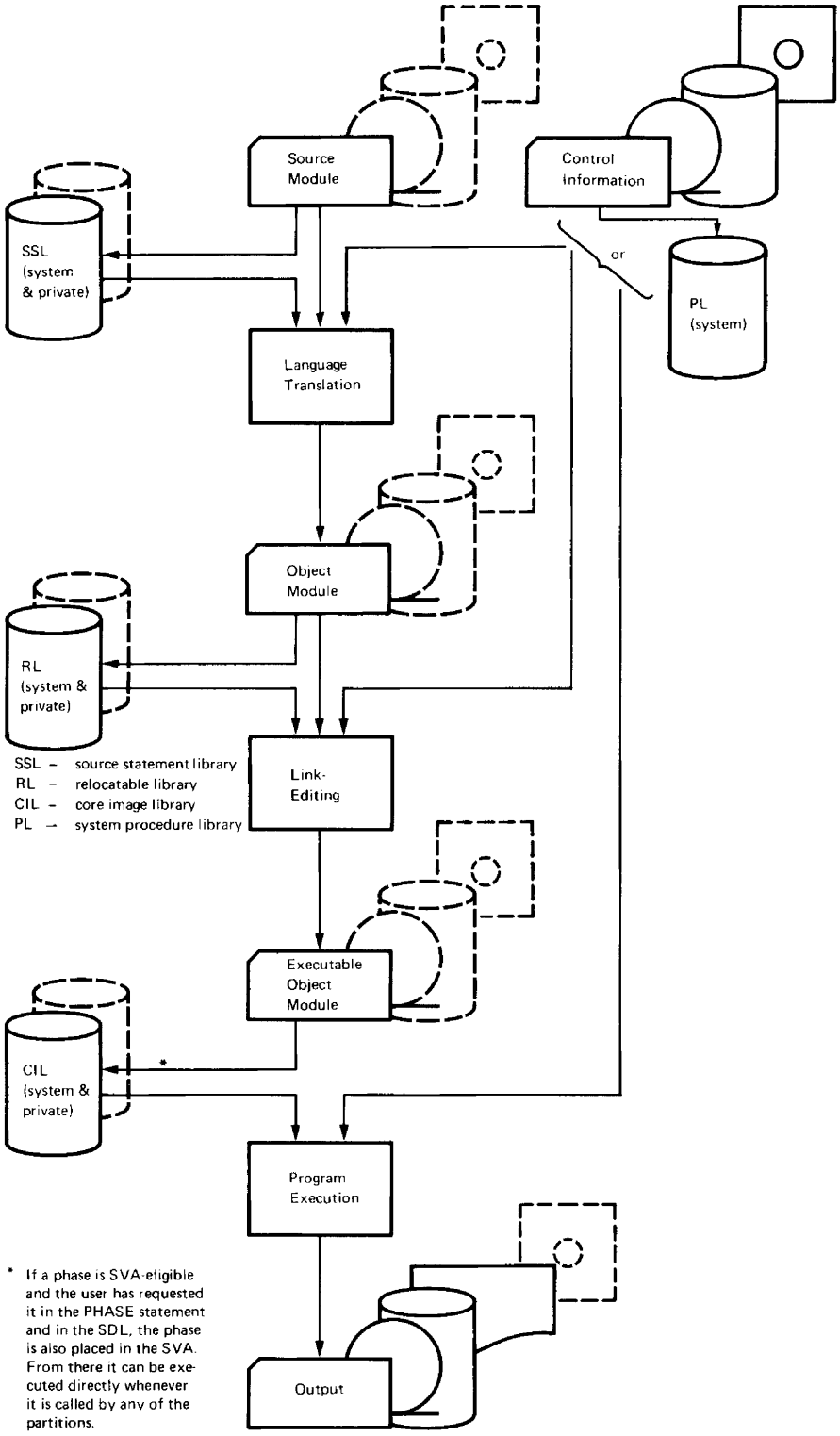


Figure 2.22. Interrelationship of Language Translators, Linkage Editor, and Libraries

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.

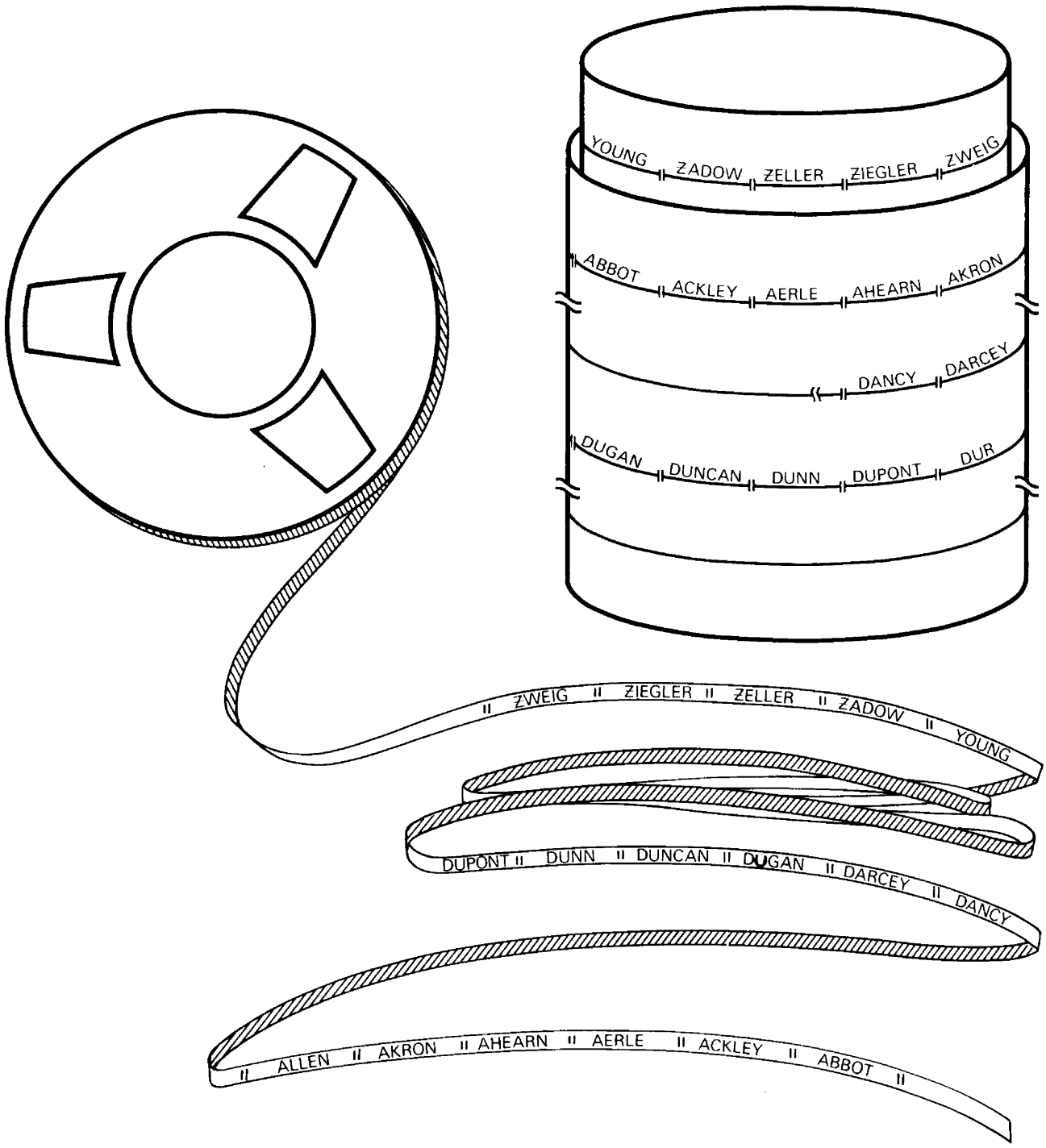


Figure 2.23. Sequential Data Organization

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

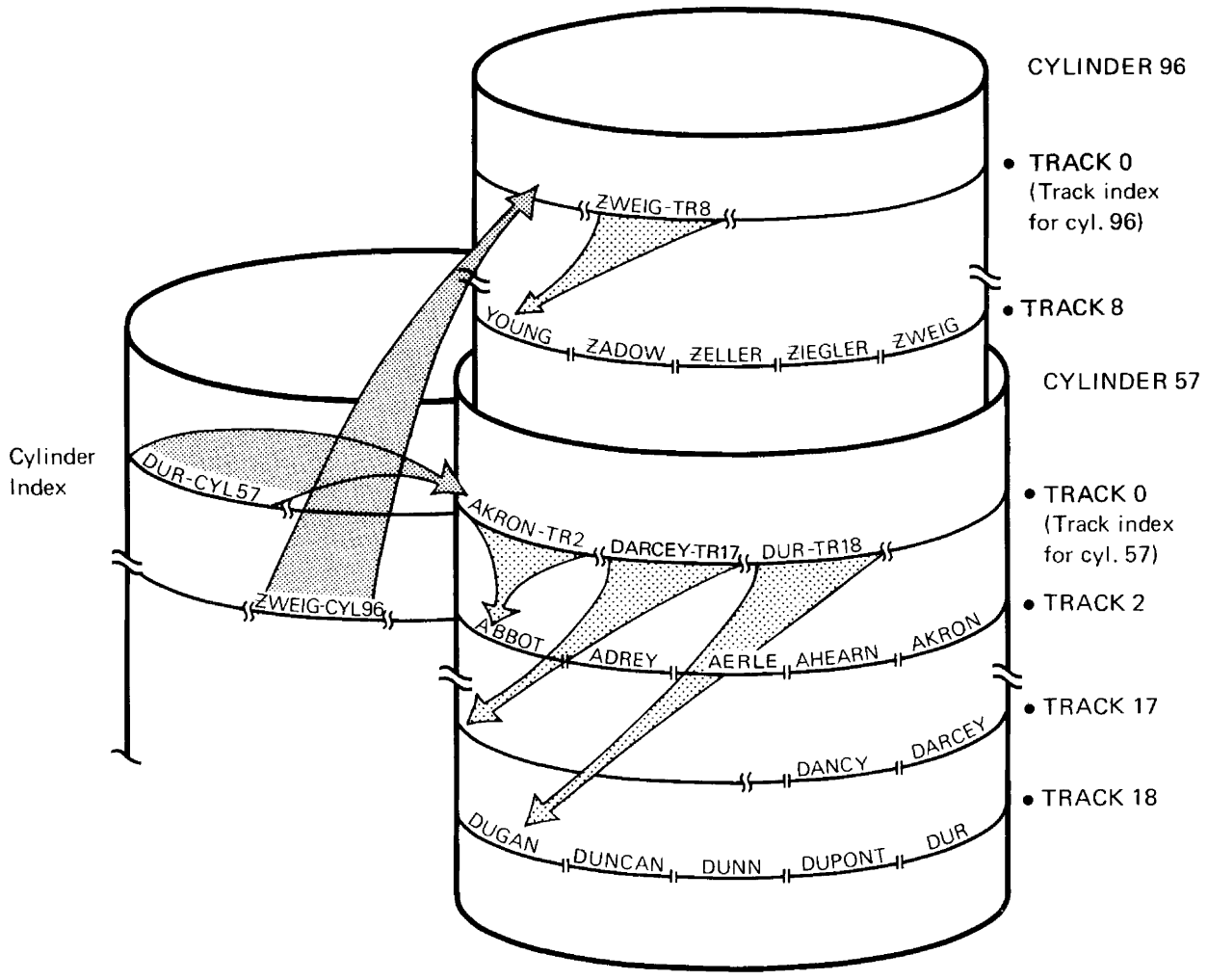


Figure 2.24. Indexed Sequential Data Organization

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.

record from the entire file, if this approach is better adapted to the 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

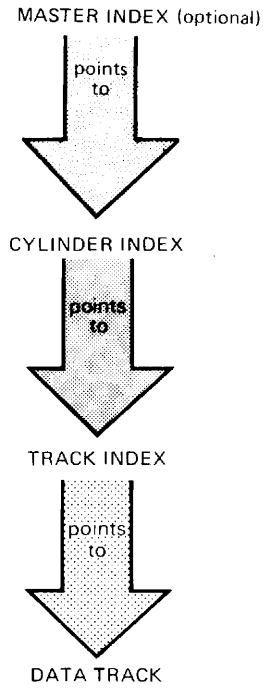
ISAM also provides the routines for creating a file from sorted input, building the index, and adding records to an existing file. For the insertion of records into an existing file, additional disk space, called overflow area, is reserved. On sequential retrieval of a file that has data in the overflow 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 Figure 2.23 may be represented in indexed sequential format as shown in Figure 2.24.

As new records are added to an indexed sequential file, 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 an access method covering a maximum of possibilities for direct and sequential processing of fixed and variable-length (including spanned) records on direct-access devices. VSAM has more functions, generally better performance, better data integrity and security, improved data organization, and is easier to use and control than other DOS/VS access methods.

VSAM storage structure

The storage structure 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. A control area is a group of control intervals (see Figure 2.25).

file organization

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

Key-Sequenced Files. Key-sequenced files are ordered according to a user-defined key field in each record. When a key-sequenced file is created, certain portions can be left empty, that is, free space can be distributed throughout the file for subsequent use when records are added. Also, free space is made available when records are deleted. 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 minimizes data movement. Data overflow is handled by splitting the contents of the overflow unit in two equal parts.

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 reorganized as often as ISAM files.

Key-sequenced files with an index are processed either sequentially or directly, like ISAM files. They can be defined as reusable (work) files.

Entry-Sequenced Files. Records are stored in entry-sequenced files in the order in which they are written on the device. 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. An entry-sequenced file is usually accessed sequentially but it can also be accessed via an alternate index or directly.

Entry-sequenced files can be processed like sequential (SAM) or direct (DAM) files. They can be defined as reusable (work) files.

Alternate Indexes for Key-Sequenced and Entry-Sequenced Files. Instead of only one index, the prime index, you may build several indexes, called alternate indexes, for a single data file, each of which could access the file

by a different key field. This allows you to access a file in different ways so that you need not keep multiple copies of the same information organized differently for different applications. For example, a payroll file originally indexed on man number can now be indexed on other fields, such as employee name or department number. An entry-sequenced file, which does not have a prime index, can be indexed on field values in its records, such as name and location.

Relative-Record Files. The records in a relative-record file are stored according to their relative position in the file. A relative-record file can be regarded as a string of fixed-length slots or record areas, each of which is assigned a relative record number. A record area may be empty or occupied by a record, which is then identified by the number of the slot.

A relative-record file has no index because it is organized in a way that allows VSAM to calculate the address of a control interval that contains the requested record and its position.

You may update records in place, delete records, or insert new records into empty record areas. A relative-record file is processed by key, where the relative record number is treated as a key.

Relative-record files can be defined as reusable (work) files.

data organization

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, or it can contain the segment of a record that spans two or more control intervals. 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

VSAM allows retrieval, storage, update, and deletion of records. VSAM files may contain variable-length as well as fixed-length records. The access can be either sequential or direct. It can be by record key, record address, or relative record number. The key can be that of an individual record or it can be a generic key specifying a group of individual records. Blocking and deblocking of records is done by the access method which optimizes block length to suit the device on which the file is written.

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. (This is called skip-sequential access.)

For keyed or addressed sequential processing there is also an option to process records backwards. The backward option cannot be used with skip-sequential access.

Normally, each file is to be opened and closed by the user. In the case of a VSAM file, however, an attempt is made to close the file automatically if the user forgets to do so before the end of a job. This assists in preventing data that is still present in the buffer from being lost due to program errors or neglect.

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

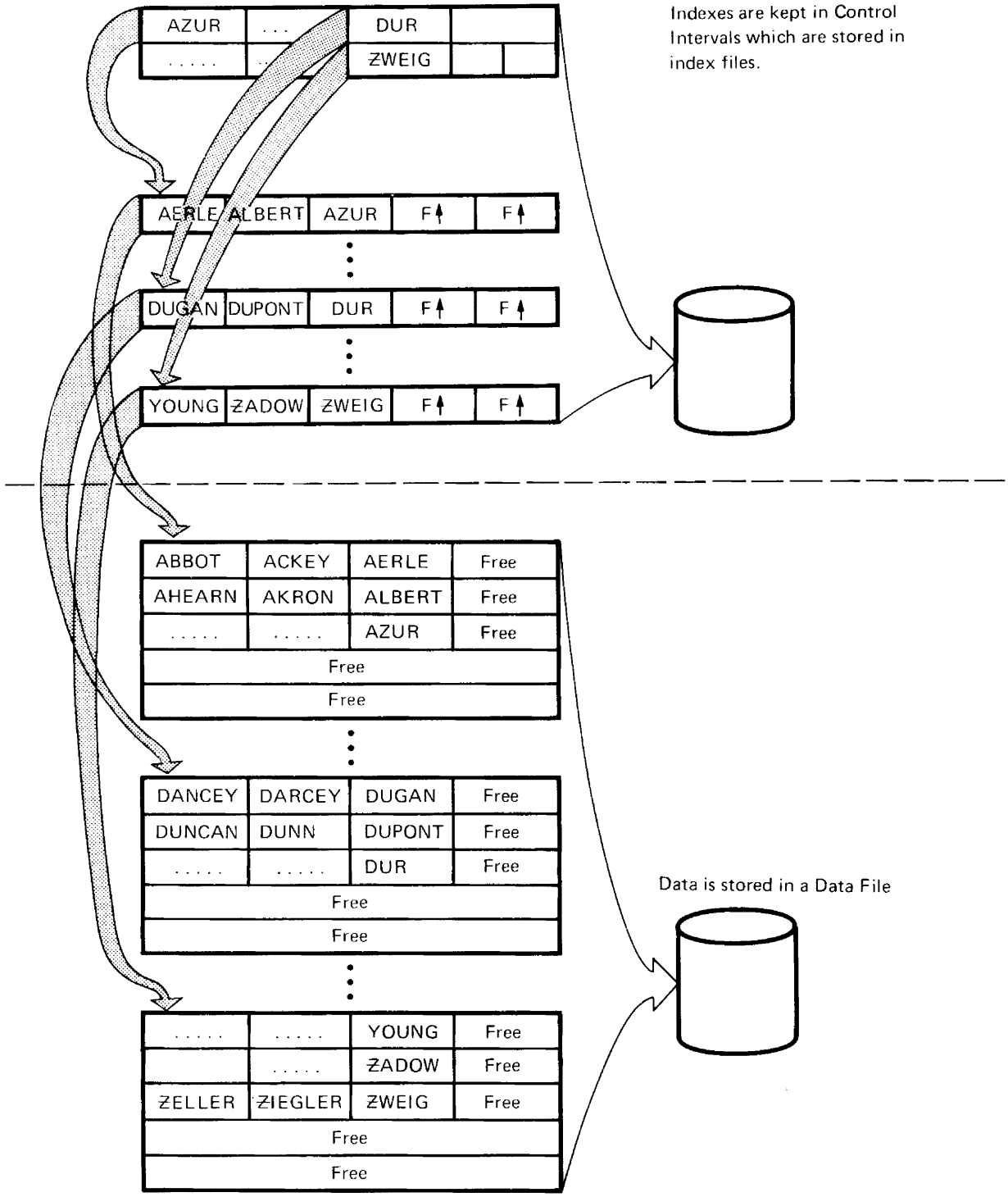
VSAM keeps control over the creation, access, and deletion of files and over the direct-access storage space allocated to those files. This is done by keeping information on file and space characteristics in a VSAM catalog. The information stored in the catalog is used by OPEN and CLOSE and by a set of utilities for handling VSAM files. A catalog also allows VSAM files to be moved 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 can be sub-allocated space on that volume by VSAM. It is not necessary to mount a volume to determine whether or not it has space available for a VSAM file.

master and user catalogs

There are two kinds of VSAM catalogs, master and user catalogs. One master catalog is required with VSAM; any number of user catalogs are optional. User catalogs are pointed to by the master catalog and have the same structure and function as the master catalog. Their main purpose is to increase data integrity and to allow volume portability.

VSAM offers improved data security and integrity. 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, (3) special formatting to indicate the end of a file as it is being created or extended, and (4) a special provision for recovery from damage to the catalog.

All VSAM catalogs may be defined with a recovery attribute, which makes it possible to restore a catalog should it ever be destroyed. Recovery is achieved by recording catalog information about a given volume on that volume, as well as in the catalog itself.



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.

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 and implementation

Support of VSAM is provided through macros in the assembler language, DOS/VS COBOL, and PL/I Optimizer.

Existing ISAM assembler language programs can access VSAM files through the ISAM interface program (IIP) which translates ISAM requests into VSAM requests. The program does not have to be compiled or link-edited again, because the interface routines are incorporated as a VSAM file is opened. However, the full scope of VSAM cannot be obtained through IIP, since only the ISAM statements of a program are mapped into comparable VSAM statements. IIP also provides the interface between ISAM programs written in ANS COBOL, PL/I, and RPG II, and the VSAM files to be processed.

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

service programs

VSAM has an extensive service program package, called DOS/VS Access Method Services, which can be used to:

- Define (create), print, copy, or reorganize VSAM files
- Allocate space to a file
- Load records into a file
- Create a backup copy of a file or catalog
- Create VSAM catalogs
- Alter, delete, or print catalog entries
- Convert a sequential or indexed-sequential file to the VSAM format
- Build an alternate index for a file
- Recover a destroyed catalog
- List the catalog recovery area and compare it with its catalog
- Recover from certain types of damage to a file.

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.

file sharing

VSAM uses the DOS/VS track hold feature to allow files to be shared across partitions. When a record is updated, the control area 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.

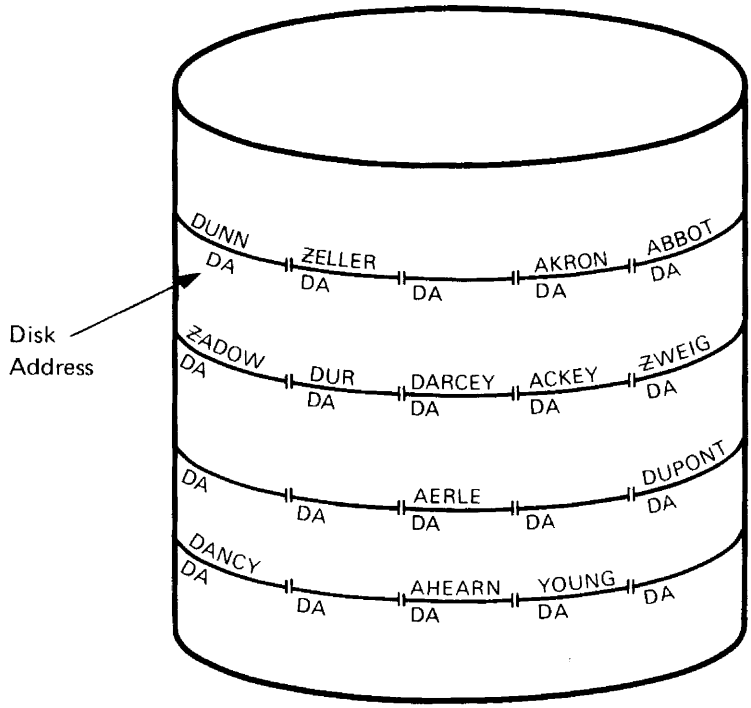


Figure 2.26. Direct Access Method Data Organization

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.

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

Telecommunication Access Methods

In telecommunication, 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 communication lines.

The routines provided by DOS/VS to support telecommunications are contained in BTAM (Basic Telecommunication Access Method), VTAM (Virtual Telecommunication Access Method), and QTAM (Queued Telecommunication Access Method). Use is made of these facilities through assembler language macros.

The BTAM macros provide broad device support and wide functional flexibility. They allow the programmer to conveniently control the telecommunication features of his DOS/VS controlled installation but, on the other hand, require him to be aware of the operational characteristics of his installation's network configuration. The BTAM routines are executed in the same partition as the problem program issuing BTAM macros.

For more information about DOS/VS BTAM and its usefulness at a DOS/VS controlled installation, refer to *DOS/VS BTAM*, GC27-6989.

VTAM, which must run in a separate partition, allows the programmer to code application programs without knowing how the installations network is configured. The flow of data between the application program and the terminal can be controlled outside the CPU by a 3704 or 3705 Communications Controller. VTAM is the primary access method for large or complex telecommunication networks.

Available generation options allow for tailoring an installation's telecommunication support to the installation's requirements.

Basic services available through VTAM macros include:

- Establishing, controlling, and terminating lines of communication between an application program and one or more terminal devices.
- Transmitting data between the application program and terminal devices as needed.
- Permitting two or more application programs to share communication lines, communication controllers, and terminal devices.
- Permitting the installation's network to be monitored and its configuration to be altered during operation.
- Assisting in network service operation through RAS (reliability, availability, serviceability) aids.

For more information about the available VTAM support, refer to *VTAM Concepts and Planning*, GC27-6998.

QTAM, which must run in real mode, is also available for message control and message processing.

There are five ways in which telecommunication 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 telecommunication 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 telecommunication 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 telecommunication 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 VTAM
				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	Through ISAM Interface Program Only	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	YES	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.

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.*

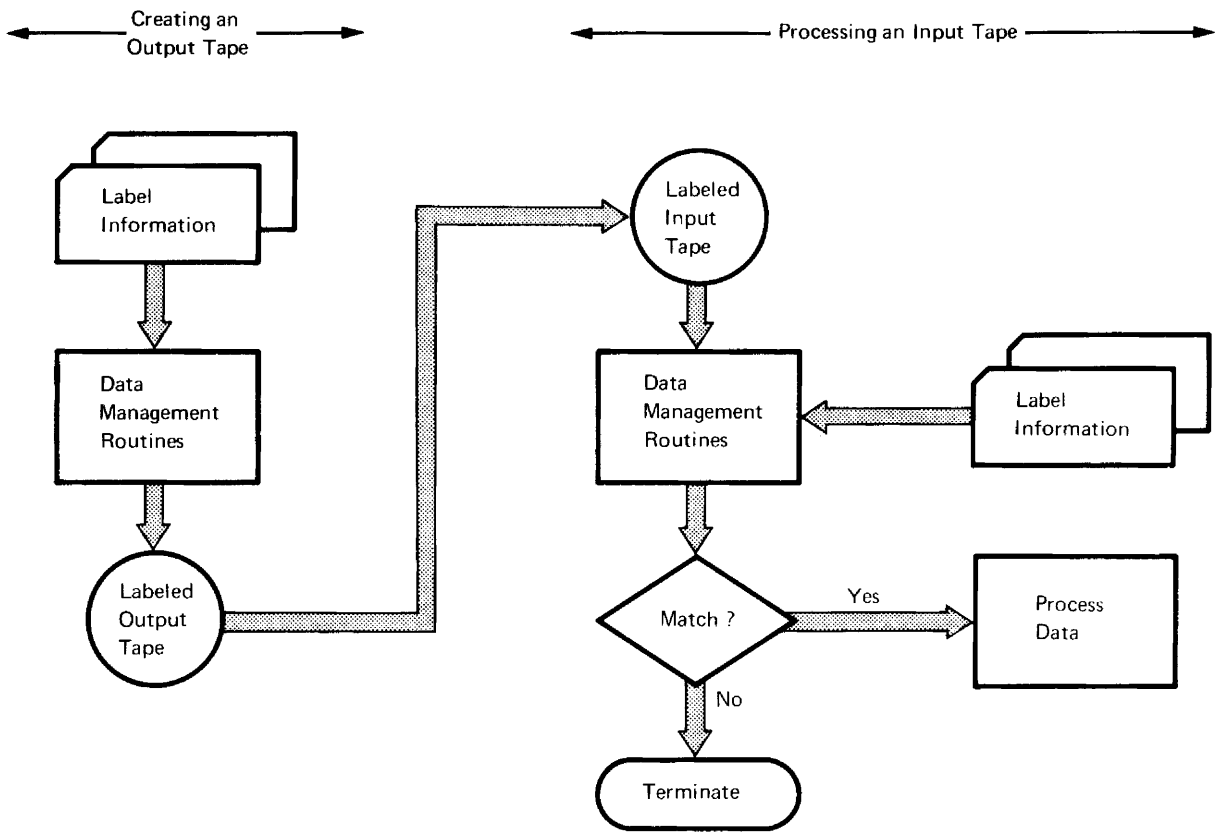


Figure 2.29. Label Processing

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.

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, except when POWER/VS is operational and access of the devices is under its control.
- 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.

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 updating 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.

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 task 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/VS 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 communication device, which may be a printer keyboard, console, or the screen of a display operator console (3277 or 125 DOC). 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.

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 keep 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 an 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.
- **Teleprocessing Online Test Executive Program** (TOLTEP), together with a set of test programs, constitutes an online test system for certain resources in a VTAM telecommunications network. TOLTEP is part of VTAM; it is included in the system when VTAM is generated. TOLTEP conducts the testing of start-stop BSC (Binary Synchronous Communication) and SDLC (Synchronous Data Link Control) lines, devices attached via start-stop or BSC lines, and locally attached 3270 control units and displays. Testing of SDLC subsystems is not conducted by VTAM but is a function of the subsystems themselves.
- **Reliability Data Extractor** (RDE) 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.

Miscellaneous System Functions

DOS/VS includes a number of system service and utility programs that are essential to the proper functioning of the operating system. The functions these programs include are:

- Initializing disk and tape volumes.
- Assigning alternate tracks for disks.
- Copying libraries and data from one storage media to another (also known as the dump/restore function) for back-up and installation purposes.
- Listing of display messages from a hardcopy file.
- Displaying a disk volume table of contents.
- Maintaining object modules, applying temporary program fixes, maintaining a history of those fixes, and producing a printout of the system history or a cross reference list for the system history.
- Loading printer control buffers.

Subsystem Support Services (SSS)

SSS is an installation and maintenance service for the following industry subsystems:

- IBM 3600 Finance Communication System
- IBM 3650 Retail Store System.
- IBM 3660 Supermarket System.
- IBM 3790 Communication System.

Each of these subsystems has a control and data collection unit, called a subsystem controller, that contains control information necessary to the operation of all terminals and components attached to it.

SSS, which is available as a separately shipped DOS/VS component, can be used to place this control information on the disk file that resides in the subsystem controller and to provide a means for maintaining that data. SSS accomplishes this by maintaining, at the System/370 host processor, a library of control information required to operate the industry subsystem. This library contains user-coded application programs and user-defined control records that define the operational environment for each subsystem controller.

An operational environment is based on the configuration of all devices within the subsystem, and on the various industry-related or application-related options that are available with each industry system.

By maintaining a central library at the host processor, SSS provides the capability for installing and updating several subsystems through a single control facility. This capability is in the form of control statements, which may be used to create the subsystem library, make modifications to it, and transmit selected portions of its contents to the subsystem controllers. In addition, SSS control statements may be used to obtain printouts of selected portions of the subsystem library, for use in monitoring subsystem activity.

The facilities of SSS and instructions how to install the component are documented in *IBM System/370 Subsystem Support Services User's Guide*, GC30-3022.

Emulation Under DOS/VS

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.

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 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 and 148	yes	yes	no
Model 135 and 138	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 DOS/VS system distributed by IBM. System generation procedures include:

- Generating a supervisor adapted to the installation and its application needs. If one of the IBM-supplied supervisors meets these needs wholly or partially, this generation step is simplified considerably.
- Assembling or compiling and link-editing user and IBM programs as necessary; cataloging these programs 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. 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's configuration and functional options. These macros are assembled, resulting in a new supervisor, which is 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:

- Selecting the most suitable version from the IBM-supplied supervisors.
- Planning the options and estimating the size of the supervisor. This entails selecting the options that are to be included in the supervisor, and estimating the cost of these options in terms of bytes of real storage. The supervisor is composed of assembled supervisor generation macros. These macros describe the machine configuration, and the installation's standard I/O assignments and 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 available DASD space 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 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 other libraries on the same disk pack, and the plans for future space requirements.

More detailed information for planning system generation is given in *DOS/VS System Management Guide*. Implementation procedures are 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 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.
- 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 assemble the macros that he requires. He can generate a new supervisor from the supervisor generation definitions, and he can assemble IOCS modules from the IOCS macro definitions. For the user's convience, the source statement library also contains sample problems and system generation job streams that can be retrieved as needed.
- IBM supplies procedures in the procedure library to help improve system performance when loading IBM-supplied phases into the SVA. The procedures are SDL, RPS, VSAMSVA, and VSAMRPS.

  Procedure SDL loads a set of selected system phases, but no RPS or VSAM phases. Procedure RPS loads the same phases as procedure SDL plus RPS phases. Procedure VSAMSVA loads the same phases as

procedure SDL plus VSAM phases. Procedure VSAMRPS loads the same phases as procedure SDL plus RPS and VSAM phases.

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 Different about DOS/VS?

This part is divided into two sections. The first section *Advantages over DOS* covers items in DOS/VS that are different from those in DOS and a summary about the compatibility between the two systems. The second section *New in Recent Releases* covers newer items that belong to Releases 32, 33, and 34.

Advantages over DOS

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 (refer to *New Devices Supported*). DOS/VS features also expand the programming facilities available to the user. Among the highlights:

virtual storage

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 of 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.

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.

variable partition priority

The user can now modify the sequence of partition priorities by using an operator command.

DOS/VS 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 extensible 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. More about VSAM is contained in the sections *Data Management* and *New in Recent Releases.*

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.

rotational position sensing (RPS)

RPS is a feature of IBM disk storage devices and provides for a potential increase in channel utilization and thus can increase I/O and system throughput.

new core image library organization

Because of the new layout of the core image library, the system residence file no longer requires subdirectories. The new layout provides for fast retrieval of all phases. The restriction that private core image libraries be of the same device type as SYSRES has been removed. Private core image libraries can reside on any disk device supported by DOS/VS. For more details on this, refer to the *DOS/VS System Management Guide.*

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.

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 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.

Virtual Storage

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.

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 standard feature. The default value in the RELLDR parameter of the FOPT system generation macro is YES. 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 default value), 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 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 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. For each specified partition, provision is made for two DASD tracks, one track to contain the partition's standard label information, and one to contain user label information. Additional tracks are available for use by all partitions.

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 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 minimum 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.

The Shared Virtual Area (SVA)

Phases that are stored in the shared virtual area (SVA) can be used concurrently by more than one virtual partition. To be used concurrently, a phase must be relocatable and reenterable. VSAM phases, for example, can be located in the SVA. The phase must also be listed in the system directory list (SDL) with the SVA indicated and must be cataloged in the system core image library. The retrieval of phases from the SVA is faster than the retrieval of the same phases from the core image library.

The SVA is located in the high end of the virtual address area. Besides being able to contain phases, it can contain an SDL and a system GETVIS area. The SDL contains directory information about all phases in the SVA and other frequently-used phases in the system core image library that are not in the SVA. The SDL allows for fast retrieval of the phases listed in it.

Implementation

When generating a multiprogramming system, the user specifies the size of the SVA through the SVA parameter in the VSTAB macro. Otherwise, the system will contain an SVA of 64K bytes and a system GETVIS area of 0K bytes. The size of the SVA can be changed by the SET command immediately following an IPL.

At link-edit time, the user can add the SVA parameter to the PHASE statement. This indicates that the system should not only catalog the phase to the system core image library, but also store it in the SVA if it was listed in the SDL with the SVA indication. It is the user's responsibility to ensure that such phases are relocatable and reenterable. Some IBM-supplied phases that are suitable for the SVA are listed in a procedure that is distributed with the system.

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 and TEMP parameters which indicate whether the assignment should be permanent or temporary.

Rotational Position Sensing

Rotational position sensing (RPS) is a feature of IBM disk storage devices. It is standard on the IBM 3330/3333/3344/3350 and optional on the IBM 3340 devices. It provides the ability to overlap rotational positioning operations on one device with service requests for other devices on the block multiplexer channel, (or its equivalent on Model 3115/3125 CPUs). It thereby provides for a potential increase in channel utilization which can lead to an increase in I/O and system throughput. DOS/VS support for RPS is provided in all access methods which support the devices and in DOS/VS system control and service programs.

Implementation

Implementation of RPS support in DOS/VS utilizes the virtual storage capabilities of the system to enable the use of the feature without the need to recompile or relink-edit any problem programs.

To enable the problem programs and system control programs to use RPS, you must generate RPS support into the supervisor, provide sufficient space in the SVA for loading of the logic modules, and provide sufficient space in each partition GETVIS area to construct RPS channel programs.

More detailed information on partition GETVIS and SVA requirements is available in the *DOS/VS System Management Guide.*

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.

Subsystem Support Services (SSS)

This DOS/VS service function, which is available as a separately shipped component, facilitates the installation and the programming service for the following industry subsystems:

- IBM 3600 Finance Communication System.
- IBM 3650 Retail Store System.
- IBM 3660 Supermarket System.
- IBM 3790 Communication System.

SSS places required control information on the disk file in the subsystem's controller and provides for servicing that data. It's control library at the host processor enables SSS to install and service several subsystems through a single control facility - control statements that can be used to create or modify the subsystem library and to transmit selected portions of the library to the subsystem controllers.

Implementation

The code as shipped is to be included in a private relocatable library and the D and F sublibraries of a private source statement library in accordance with instructions in the *Memo to Users.* Next, the support is to be assembled (using the IBM-provided generation macro INDGEN) and linkedited to a private core image library. SSS is then ready for execution.

Emulation on Models 115 and 125

An 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.

New Devices Supported

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

- The IBM 3330 Model 1 and 2, and IBM 3333 Model 1 disk storage devices for attachment to DOS/VS supported System/370 CPUs except Model 115.

  The IBM 3330 Model 11 and IBM 3333 Model 11 disk storage devices for attachment to DOS/VS supported System/370 CPUs except Model 115 and 125.
- IBM 3350 in 3330 Model 1 and 11 compatibility mode, or in native mode with System/370 CPUs except Model 115 and 125.
- All models of the IBM 3340 and 3344 disk storage drives for attachment to DOS/VS supported System/370 CPUs. All models of the 3348 Data Module used on the 3340 are also supported.
- 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 Console Printer enhance operating flexibility. The display also replaces the manual control switches.
- The multifunction 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 Model 1 and 2 Printer, which attaches to the Model 115 and 125.

- The IBM 3203-4 Printer, which attaches to the Models 138 and 148.
- The IBM 3540 Diskette Input/Output Unit.
- The IBM 3803 Tape Control Model 3 for Models 115 and 125.
- The IBM 3881 Optical Mark Reader.
- The IBM 3886 Optical Character Reader.
- The IBM 3741 Data Station (supported by POWER/VS as a RJE terminal).
- The IBM 3600 Finance Communication System.
- The IBM 3650 Retail Store System.
- The IBM 3660 Supermarket System.
- The IBM 3770 Data Communication System.
- The IBM 3790 Communication System.
- The IBM 3277 Display Station - now also supported as operator console.

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 to process data for a 2311 can be executed to process the same type of data for a 3333, 3330, or 3340 attached to a System/370 Model 125, or on a 3340 attached to a System/370 Model 115 or 135, through the use of the available 2311 Compatilibity Feature. 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 Models 115 and 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.

New in Recent Releases

This section highlights facilities and feature support in Releases 32, 33, and 34 of DOS/VS.

New in Release 32

The subsequent sections cover the changes and additions introduced with DOS/VS Release 32.

- VTAM Enhancements
- POWER/VS Enhancements
- Fast CCW Translation Option
- Partition Dump Option
- High-Speed Dump Program (DOSVSDMP)
- CIL Patch Program (PDZAP)
- Cross-Partition Event Control
- New Devices Supported

VTAM Enhancements

VTAM now supports unformatted LOGON and LOGOFF commands entered by the terminal operator for the following SNA terminals:

- IBM 3270 Information Display System
- IBM 3767 Communication Terminal
- IBM 3770 Data Communication System

The IBM 3650 Retail Store System and the 3790 Communication System are now supported both on switched and nonswitched lines; the 3790 Communication System, moreover, is supported via local channel attachment.

POWER/VS Enhancements

- POWER/VS now supports two buffers for input from card readers, thus providing for increased throughput of input data. This feature is not supported for the 3540 diskette. Double buffers are specified in the PSTART command.

- The POWER/VS remote job entry (RJE) compatibility now supports the IBM 3741 Data Station as a 2780 Data Transmission Terminal without the multiple record transmission feature.
- A default class for output may now be specified by partition when each partition is activated with the PSTART command.

Fast CCW Translation Option

As the hardware support for virtual storage does not include the translation of virtual addresses contained in channel programs, these addresses are translated by DOS/VS. This operation is normally carried out each time a channel program is executed in a virtual partition.

The fast CCW translation option applies only to programs running in virtual mode. If the option is active, DOS/VS attempts to save and re-use any channel programs which have already been translated, provided that there is sufficient storage available.

The option is activated by specifying FASTTR=YES in the supervisor FOPT macro, and further details of its use can be found in *DOS/VS System Generation,* GC33-5377.

Partition Dump Option

The common DOS/VS printout of a system storage dump may contain information that is irrelevant to the debugging of an abnormally terminated program.

A new option, Partition Dump, allows the user to limit the contents of the dump printout to pertinent information. This can result in advantages such as reducing printed output, improving turnaround time, and making the debugging of a program more convenient.

The Partition Dump option may be invoked in either of two ways:

- As a standard option during system generation by specifying in the STDJC supervisor macro: DUMP=PART.
- Temporarily, using the job control statement: // OPTION PARTDUMP.

Except for the supervisor, the output of the partition dump has the same format as that of the common dump.

Support of the already existing dump options and dump macros is not affected. For more details on this new option, refer to *DOS/VS Serviceability Aids and Debugging Procedures,* GC33-5380.

High-Speed Dump Program (DOSVSDMP)

The IBM DOSVSDMP creates a standalone dump program and provides the means to print a copy of the storage, dumped by the standalone program.

The advantage of the DOSVSDMP over the already existing DUMPGEN facility lies in the reduced tie up of the system because the high-speed dump program dumps all of storage immediately onto tape or disk; the system can then be started again for normal operation. Also, at this stage, a new or additional partition may be started to print the dump file.

DOSVSDMP is provided as a self-relocating program in the relocatable library and can be executed in any partition through the command // EXEC DOSVSDMP. The print facility of DOSVSDMP provides for formatted or unformatted printout. The formatted type has each field of certain system control blocks identified, the unformatted type of printout is only a conventional translated copy of storage.

Note that 7-track tape drives without data converter are not supported by DOSVSDMP.

DMPROG, the standalone dump program (invoked by DOSVSDMP), may reside on either tape or disk and is executed as the result of an IPL procedure. In either case, the standalone dump writes the storage dump onto the devices on which it resides. DMPROG produces a copy of the virtual storage in virtual page order; included in the virtual dump are those pages that are allocated but are paged out to SYSVIS. When all of virtual storage is written out, or when an error occurs that prevents this, real storage is written out in real page order.

Details on the use of DOSVSDMP are given in *DOS/VS Serviceability Aids and Debugging Procedures,* GC33-5380.

CIL Patch Program (PDZAP)

The IBM program PDZAP allows the user to display the contents of core image library phases and to make modifications to such phases. The phases can be cataloged in either the system or private core image library (SCIL or PCIL).

PDZAP acts on control statements which can be entered in a number of formats either through a console or via a card reader. Up to 16 bytes of object code values can be visually verified or changed at one time.

PDZAP allows the user to make temporary fixes until a permanent fix can be made by modifying and recompiling source statements. File protection is bypassed when modifications are made.

Cross-Partition Event Control

Cross-partition event control enables communication between partitions; it is used primarily in subsystem applications.

For detailed information on cross-partition event control, see *DOS/VS System Management Guide,* GC33-5371.

New Devices Supported

The following new direct access storage devices are supported:

- The IBM 3350 in 3330 compatibility mode, which is equivalent to two 3330-1 volumes.
- The IBM 3344 Disk Storage, which is equivalent to four IBM 3340s with 3348 Model 70 data modules mounted.

The DASD concept of *compatibility modes* is applied to the operation of, for example, the IBM 3350. This mode then allows that one physical spindle of the IBM 3350 appears logically as another type of DASD. Thus one IBM 3350 physical spindle is used as two 3330 Model 1 logical spindles.

For more details, refer to *Introduction to IBM 3350 Storage,* GA26-1638, which outlines hardware features, and programming and functional characteristics.

New Models for the IBM 3115 and the IBM 3125

New models have been added to the existing groups of IBM 3115 and IBM 3125 Processing Units. They are the 3115-2 and 3125-2 and offer the following enhancements over the comparable existing models, the 3115-0 and 3125-0:

- Increased execution rate for both, the 3115-2 and 3125-2.
- Increased data rate due to an extended byte multiplexer channel for the 3115-2.
- Increased number of 3340 spindles. Up to eight spindles of IBM Disk Storage can be attached to any 3115-2 and up to sixteen spindles to the 3125-2. This doubles the potential online DASD capacity compared to the 3115-0 and 3125-0.
- The string-switch feature for the IBM 3340.
- Larger storage size (256K) for the 3115-2.
- Improved CPU logic for the 3115-2.

The string-switch feature allows sharing of 3340s between a host and one other system. The switching can be either manual or program controlled on the EXCP level (that is, by coding the channel program). An IBM 3115-2 can share one 3340 Model A2 Disk Storage and its attached string of 3340 Model B1 and/or B2 with any System/370 other than the 3115-0 or 3125-0. An IBM 3125-2 can share two 3340 Model A2 Disk Storage units and their attached strings of 3340 Model B1 and/or B2 with any System/370 except the 3115-0 or 3125-0.

DOS/VS, which currently supports the 3115-0 and 3125-0, also supports the 3115-2 and 3125-2. All I/O units attachable to the 3115-0 and 3125-0 can also be attached to the 3115-2 and 3125-2.

New in Release 33

The subsequent sections cover the changes and additions introduced with DOS/VS Release 33. The sections cover:

- Cardless System Support
- The IBM 3803-3/3420 Subsystem Support on the 115 and 125
- IBM 2501 Card Reader Performance Improvement
- BSC Support for the IBM 3660
- Larger Storage Size Models for the IBM 3115-2 and the IBM 3125-2
- VTAM Enhancements
- POWER/VS RJE Support for the IBM 3770
- POWER/VS Cross-Partition Communication
- POWER/VS Enhancements
- VS Personal Computing (VSPC)
- Cross-Partition Communication Extensions
- Subtask Priority Modification
- Task Timer
- Interval Timer Extension
- The IBM Analysis Program-1 (AP-1)
- The IBM Copy File and Maintain Object Module (OBJMAINT) Utility Program
- PDZAP with Logging Feature
- Installation Improvements
- Operation Improvements
- Supervisor Patch Area Improvements
- Optional Escape from Abnormal Termination
- System Improvements
- IBM 3340 Label Cylinder Area Extension
- VSAM Enhancements
- Access Method Services Enhancements
- New CPU Models 135-3, 138, 145-3, and 148.

Cardless System Support

Installations that are based on IBM System/370 CPU Models 115 or 125 may now be configured without card reader/card punch devices, provided that an IBM 3540 Diskette I/O Unit is included. This minimum configuration offers a reduction in device costs and an improvement in the rate of data entry. The IBM 3540 is supported as a system input/output device. Note, however, that the IPL control statements cannot be entered from the 3540, but have to be entered via the Display Operator Console (DOC). The only exceptions are the stand-alone dump program (as generated by DUMPGEN) and the stand-alone version of the Fast Copy Disk Volume system utility, which can now be stored on IBM 3540 Diskette I/O Units and loaded from there into the system. Also, the stand-alone version of the Initialize Disk utility (distributed on the PID tape) has been altered to accept control information from the Display Operator Console (DOC).

The IBM 3803-3/3420 Subsystem Support on the 115 and 125

A high-speed tape subsystem is now provided for users of System/370 CPU Models 115 and 125. The Magnetic Tape Adapter feature of the System/370 Models 115 and 125, used in conjunction with the new 3803 Tape Control Model 3, makes available the tape processing capability of the 3420 Tape Unit Models 3 and 5.

The 3420 Models 3 and 5 have data rates of 120K and 200K per second respectively. Special features provide dual density (800/1600 bpi) and the seven track option.

IBM 2501 Card Reader Performance Improvement

Modifications have been made to the double buffering concept of the DOS/VS Sequential Access Method (SAM) in order to improve the performance of the IBM 2501 Card Reader when it is attached to System/370 CPU Models 3115-2, 3125-0 or 3125-2.

BSC Support for the IBM 3660

Subsystem Support Services (SSS) now provides binary synchronous communications (BSC) support for the IBM 3660 Supermarket System. An overview of SSS is given in Part 2 of this manual. For detailed documentation of the SSS facilities, please refer to *IBM System/370 Subsystem Support Services User's Guide,* GC30-3022.

Larger Storage Size Models for the IBM 3115-2 and IBM 3125-2

Models of larger storage size have been added to the existing groups of IBM 3115-2 and IBM 3125-2 Processing Units. For the 3115-2, the Model HG2 provides 384K. For the 3125-2, the two Models HG2 and I2 provide a storage size of 384K and 512K respectively.

VTAM Enhancements

VTAM Level 2 supports facilities that:

- Allow an installation to authorize VTAM application programs to issue VTAM network operator commands (except START VTAM and HALT NET) and receive responses to commands.
- Provide recovery of the VTAM network configuration to either its initial status or its status prior to the deactivation or failure of an NCP, host operating system, host processor, or VTAM. This feature allows the installation to define VSAM configuration restart data sets to record changes in the network configuration.
- Allow logical units to be associated with a predefined set of SNA-oriented application program protocols. Procedures are available that enable VTAM, the application program, and the logical unit to inform each other of the protocols that are to be used.

POWER/VS RJE Support for the IBM 3770

The POWER/VS remote job entry (RJE) teleprocessing facility now supports the IBM 3771, 3773, 3774, 3775, and 3776 Communication Terminals using Synchronous Data Link Control (SDLC).
For further information on RJE support, refer to the manual *DOS/VS POWER/VS with RJE,SNA Guide,* GC33-5405.

POWER/VS Cross-Partition Communication

POWER/VS functions may now be accessed by other DOS/VS partitions when using the new macros PUTSPOOL, GETSPOOL, and CTLSPOOL. Commands, such as PSTART or PSTOP, are not required with this new cross-partition facility. Spool management tasks are created and deleted dynamically by POWER/VS.

POWER/VS Enhancements

The following changes have been made to POWER/VS to enhance operations:

- Forms Control Buffer Handling
  - A default FCB is loaded for all LST output that does not have an FCB specification in the LST statement.
  - FCB loading is provided for all segments of segmented output regardless of the order in which the output is printed.
  - FCB is loaded before separator pages are printed.
  - Redundant loading of forms control buffers has been eliminated.
  - Ending separator pages are now printed only at the end of the last copy of output.
- Multitasking Extensions
  - With the new macro SEGMENT it is now possible to segment LST and PUN output.
  - The new parameter MT in the PSTART command identifies a partition as a multitasking partition.
  - Unique job numbers are now provided for LST and PUN output.
- Execution Account Record for POWER/VS Partition
  For POWER/VS with job accounting, an execution account record is now written at normal shutdown time for the POWER/VS partition.
- Partition Independent SLI Books
  POWER/VS now supports source library inclusion (SLI) books with partition independent names. For these names, the same naming conventions are used as for the procedure library.
- LST and PUN Statements in SLI Books
  LST and PUN statements may now be included in source library inclusion (SLI) books.
  Restriction: Continuation of LST and PUN statements are not allowed when cataloged in SLI books.
- New Parameter (PRI=) in LST and PUN Statements
  Priority of job output may now be specified using the new parameter PRI= in the LST and PUN statements.
- New Parameter (T) in the PDISPLAY Command
  The time and date may now be displayed using the new parameter T in the PDISPLAY command.
- Short Form (U) of PSETUP Command
  The PSETUP command has now also a short form, namely U.
- Unit Record Error Recovery Options
  Whenever an irrecoverable I/O error occurs on a printer or punch device, the operator now has the option to cancel the job, restart the operation, or ignore the error.

- Extended POWER/VS Job Names and Forms Names
  Slash (/), dash (-), and period (.) are now also allowed in POWER/VS job names and forms names.

- Improved Separator Pages
  Job name, class, priority, and job number are now printed in large block letters. In addition, forms lengths of 44 through 99 are supported with printing across the perforation.

- Display of Card Counts for Reader Queue
  A PDISPLAY request of the reader queue now also displays the number of cards in the queue entry.

- Performance Improvements
  - The message CP READY has been shortened to OK to improve performance.
  - The display time for queues on non-CRT console systems without RJE has been shortened by omitting the '*****' printed to the right of each displayed queue line.
  - Execution processor, printer, and READER exit performance has been improved.

- RJE,BSC Improvements
  - The new parameter SWITCH=NO/YES added to the PLINE macro, allows to describe to POWER/VS the BSC line configuration.
  - The short form of all POWER/VS RJE terminal commands is now accepted.
  - All non-printable characters below X'40' are translated to X'40' to allow space compression.
  - POWER/VS RJE commands may now be entered in lower case via terminals with keyboards (2770 and 3770 in BSC mode).
  - The trace facility of POWER/VS BSC,RJE has been improved to provide more meaningful information for problem diagnosis.

VS Personal Computing (VSPC)

The new IBM Program Product, VS Personal Computing (VSPC), enables a user to create and execute programs at a terminal. A description of VSPC is given in Part 4 of this manual under *VS Personal Computing.*

Cross-Partition Communication Extensions

Cross-partition communication extensions allow for RESET of the XECB, and additional conditions under which a task may be removed from the wait state.

Subtask Priority Modification

The CHAP macro allows a subtask to request that its own priority be set to the lowest priority of all subtasks within the partition.

Task Timer

The task timer allows the main task of the partition owning the task timer to specify that it wants to receive control in an exit routine after the latter has executed for a specified period of time. The task timer requires, in the case of System/370 Models 135 and 145, the installation of the clock comparator (an optional hardware feature) and the CPU timer.

Interval Timer Extension

A new parameter on the SETTIME macro allows for the specification of a smaller (1/300 of a second) interval for the interval timer. The current minimum interval for this timer is one second.

The IBM Analysis Program-1 (AP-1)

This program determines whether a data error on an IBM 3344 or 3350 Direct Access Storage device is due to a faulty recording surface or due to a hardware error. The output of AP-1 enables the system operator or programmer to decide if the data recovery procedure or the installation's procedures for hardware problems should be initiated.
For detailed information on AP-1, please refer to *OS/VS and DOS/VS Analysis Program-1 (AP-1) User's Guide,* GC26-3855.

The IBM Copy File and Maintain Object Module (OBJMAINT) Utility Program

This IBM multi-purpose utility program is of particular interest in conjunction with System/370 cardless systems. It allows:

- File-to-file copying, including blocking and deblocking, of card image files.
- Modification of object modules, phases, and PTFs via card, tape, disk or diskette devices.
- Comprehensive listing of card image files including object programs.

For detailed information on this program, refer to *DOS/VS System Utilities,* GC33-5381.

PDZAP with Logging Feature

The CIL Patch Program (PDZAP) now includes a logging feature. The log produced by PDZAP contains information relating to changes made to core image library phases. The information written to SYSLST consists of:

- Date and time of change, and identification of initiator.
- Name and load address of the phase changed.
- Old and new data in hexadecimal notation.

For further information on this feature, please refer to *DOS/VS Serviceability Aids and Debugging Procedures,* GC33-5380.

Installation Improvements

A number of improvements have been made for greater convenience and efficiency when installing a new release of DOS/VS. This section highlights the following improvements:

- Prelinked IBM System components
- Pre-assembled I/O modules for RPGII and PL/I Optimizing Compiler
- Pre-assembled modules for the IBM 1403-N1 Printer
- A choice of seven pre-compiled supervisors
- Identification for supervisors
- DOS/VS supervisors now generated with RELLDR=YES
- Coded samples in the procedure library
- The IBM Backup and Restore Utility Programs
- The IBM Copy Service Program (COPYSERV)
- The IBM Maintain System History (PTFHIST) Utility Program.

- IBM system components are prelinked into the core image library to eliminate a time consuming step when installing a new release of DOS/VS. For the deletion of unwanted components, new delete procedures are provided.
- In addition to the standard IBM-supplied IOCS, pre-assembled versions of all I/O modules required for RPGII and the PL/I Optimizing Compiler are provided in the relocatable library.
- Buffer Load modules for the IBM 1403-N1 Printer are pre-assembled.
- DOS/VS now includes a range of seven, different, precompiled supervisors. By selecting one (or more) from this range, system generation efforts can be reduced considerably.
- DOS/VS supervisors include an identification to show which release level of DOS/VS is applicable for a given supervisor. Furthermore, a new parameter (USERID) in the FOPT macro allows the user to distinguish between different supervisors in the system. These identifiers

eliminate ambiguities.They are listed on the console at IPL time or can be derived from a supervisor dump.

- All DOS/VS supervisors are generated with the relocating feature as a standard feature. To override this standard feature, the RELLDR parameter in the FOPT macro can be specified with NO.

- The procedure library now includes coded samples for:

  - Procedures to delete and link system components
  - Standard labels
  - Creation of private libraries
  - VSAM file definition.

- The IBM Backup and Restore Utility Programs

  The two utility programs Backup and Restore are of particular interest during system generation and for library maintenance purposes. The scope of the programs is given below; for a detailed description, refer to *DOS/VS System Utilities,* GC33-5381.

  The Backup program can be used to create a backup copy on tape of DOS/VS and private libraries.

  The Restore program can be used to restore the backup copy from tape to disk and can also be used to restore the DOS/VS distribution tape to disk prior to system generation.

  The Backup and Restore programs can be used together to efficiently condense libraries. There are no condense restrictions for multiprogramming environments. Backup and Restore used together also facilitates the migration of libraries from one type of IBM disk device to another. A change in the DASD type used for SYSRES does not require a re-generation of the system from the IBM-supplied distribution tape, because the backup copy of the current SYSRES can be restored to the changed DASD type.

- The IBM Copy Service Program (COPYSERV)

  This program is part of the librarian of DOS/VS. COPYSERV automates the process of comparing the library directories of current and/or new system packs. Directories of both types of libraries, system and private, can be compared.

  The output of the program consists primarily of COPY statements that correspond to the library elements not contained on the new system pack. These statements can be used with the copy system program (CORGZ) to copy or merge the missing elements to the new system pack. COPYSERV also produces a listing of the results for control purposes.

  Because the manual comparison of directory listings is eliminated, the use of COPYSERV can significantly reduce the preparation effort when installing a new release of DOS/VS.

  For details of this program, please refer to *DOS/VS System Management Guide,* GC33-5371. See also *COPYSERV Functions now in CORGZ* in the subsequent section *New in Release 34.*

- The IBM Maintain System History (PTFHIST) Utility Program

  This new IBM utility program is a general installation and maintenance tool for DOS/VS system control programs and IBM licensed programs. It simplifies the selection of PTFs (Program Temporary Fixes) from a master PTF tape and their installation in a current DOS/VS.
  For detailed information of this program, please refer to *DOS/VS System Utilities,* GC33-5381.

Operation Improvements

The following modifications help to improve efficiency and ease of use of an installation.

- The printing of relocatable dictionary listings is suppressed through the newly introduced option RLD with a standard setting of NO. Thus, the often unwanted printout of the dictionary is eliminated. A printout can be obtained by specifying RLD=YES in the standard job control (STDJC) macro instruction, or by specifying the RLD option in the OPTION statement in the job control language (JCL).
- The $JOBACCT dummy phase can now be loaded into the shared virtual area (SVA). This improves the performance of installations that use the POWER/VS job accounting option.
- The message 'AR READY FOR COMMUNICATION', formerly issued after the pressing of the request key, is suppressed. The system enters read mode immediately after 'AR' is printed.
- The operator response 'IGNORE' to the message 'INTERVENTION REQUIRED' is no longer necessary.
- The foreground-1 partition may now have up to 241 entries of symbolic programmer logical units (LUBs).
- The whole virtual storage of a partition is now available via the GETVIS/FREEVIS macro instruction.

Supervisor Patch Area Improvements

A low core patch area of 64 bytes is now available. This makes supervisor patches with absolute addresses possible, a base register is not neeeded. The label of the low core patch area is IJBPATCH, the address is contained in the first four bytes of the area. The high core patch area is now reduced to 300 bytes.

Optional Escape from Abnormal Termination

The new supervisor macro function EXIT AB provides an alternative to the function of the existing EXIT macro. The EXIT AB macro, if included in the Abnormal Termination User Exit Routine, returns control to the supervisor to allow continued processing of the problem program (main task) rather than its termination.

For further information, refer to *DOS/VS Supervisor and I/O Macros,* GC33-5373.

System Improvements

- MAP Command

  The upper limit addresses of the storage areas displayed by the MAP command are shown in hexadecimal instead of decimal notation. These addresses can therefore be used in commands, for instance the DUMP command, that require hexadecimal notation.

- Printout of the System Directory List (SDL)

  The new DSERV parameter DSPLY SDL can be used to obtain a printout of the SDL contents. Also, the parameter DSPLY(S) ALL is extended to include the SDL as well.

- Suppression of UCB/FCB Load for Dummy Printers

  The forms control buffer load for POWER/VS dummy printers is suppressed. The message 'DEVICE NOT OPERATIONAL' will no longer appear for the dummy printers at IPL time.

IBM 3340 Label Cylinder Area Extension

IBM 3340 Disk Storage, if used as a system residence (SYSRES) device, now provides a second cylinder on which to store standard label information for all the partitions assigned. This cylinder is an extension of the system standard label area and requires no specification for its allocation.

The additional cylinder allows an increase in the number of labels that can be used and is of particular interest when operating a five partition system.

The above change does not affect operations that involve IBM 2314, 2319, 3330, or 3333 devices. The total space available on these devices for label information remains unchanged.

For more information on this subject, refer to *DOS/VS DASD Labels,* GC33-5375.

VSAM Enhancements

The VSAM shared resources facility allows application programs to share buffers, I/O control blocks, and channel programs among several VSAM files within a partition. Sharing of these resources optimizes their use and reduces the amount of virtual storage required per partition. The facility is especially useful:

- When many VSAM files are open and it therefore becomes difficult to predict the amount of activity for a given period.
- When a transaction refers to several files.

The buffers and control blocks are allocated from a common resource pool when an I/O request is issued for a file; on completion of the request, the same buffers and control blocks can be assigned again to another file. Provision is made to display certain file information for the purpose of defining the resource pool.

In addition, facilities are provided for managing I/O buffers. VSAM buffer management can be used to increase the processing speed of VSAM files that have unpredictable activity. The facilities include:

- Deferring write operations for direct PUT requests. This reduces the number of I/O operations.
- Relating deferrred requests by a translation ID.
- Writing out buffers whose writing has been deferred.

Resource sharing and buffer management are provided through macros (and macro extensions) in the assembler language. For information, please refer to *DOS/VS Supervisor and I/O Macros,* GC33-5373.

DOS/VS Access Method Services Enhancements

- VSAM Catalog Recovery
  Access Method Services now include a new tool for VSAM catalog recovery; it is implemented by the new command RESETCAT.

  This command provides for synchronization of the entries of a VSAM catalog and the associated recovery information recorded in the catalog recovery area (CRA). Use of RESETCAT is an efficient way to reset a recoverable catalog to a level consistent with the CRA volume, particularly when extensive discrepancies exist. For further information, refer to *DOS/VS Data Management Guide,* GC33-5372.

- LISTCAT Output Enhancement
  The output from the LISTCAT command is now in tabular format. This improves the readability of the listed VSAM catalog entries.

- ALTER Error Checking
  The ALTER command now provides for additional error checking. Incompatiblities between the object to be altered and the attributes specified in the command are detected.

New CPU Models 135-3, 138, 145-3, and 148

System/370 CPUs with improved performance have been added to the existing groups of IBM 3135 and 3145. They are the IBM 3135-3 which provides storage from 256K to 512K, and the IBM 3145-3 which provides storage from 192K to 1984K.

Two new groups of central processing units are now available: The IBM 3138 and the IBM 3148. These CPUs are equipped with an IBM 3277 Display Station as operator console which can be used in one of two modes:

- Display Operator Console (DOC) as installed in Models 115 and 125.
- IBM 3210 or IBM 3215 Printer-Keyboard as installed in Models 135, 145, and 155. This mode requires the 3286-2 console printer.

New in Release 34

The subsequent sections cover the changes and additions introduced with DOS/VS Release 34. The sections cover:

- New Devices Supported
- IPL Communication Device List
- IBM 3540 as IPL Communication Device
- New Procedures to Load the SVA
- New Parameter in the DLBL Statement
- Access Method Services Enhancements
- EREP Enhancements
- The List System History Program (HISTLIST)
- COPYSERV Functions now in CORGZ
- Job Accounting Improvements
- POWER/VS Enhancements

New Devices Supported

- The IBM 3277 Model 2 Display Station can be used as display operator console (DOC) on all DOS/VS supported CPUs.

  The screen of the 3277 Model 2 has 24 lines, 20 of which are used as message area. The remaining four lines are used as instruction line, entry area (2 lines), and warning line. The full length of 80 characters is supported.

  The supervisor must be generated with DOC=3277 in the FOPT macro. When used as DOC, the 3277 must not be attached to a selector channel.

- Full support is provided for the new IBM 3330-11 (3333-11) Disk Storage and the IBM 3350 Direct Access Storage. ISAM is not supported on IBM 3330-11 and IBM 3350. The 3330-11 is not attachable to CPU Model 115; the 3350 in native mode or in 3330 compatibility mode is not attachable to CPU Models 115 and 125.

  Existing programs can access files on the 3330-11 and 3350 without modification, providing all of the following conditions are met:

  - The supervisor supports RPS,
  - Sufficient SVA space is available to load RPS logic modules,
  - Sufficient space is available for an RPS DTF extension,
  - The program runs in virtual mode.

- The new IBM 3203-4 Printer is compatible with the IBM 3211 Printer and is supported as such. The 3202-4 and 3211 can be specified to the system by the same device class (PRT1). The 3203-4 is attachable only to Model 138 and 148 CPUs.

IPL Communication Device List

It is now possible to build a restrictive pool of communication devices for use during initial program load (IPL). Transmission of IPL commands, then, can only occur from one of these devices. This is of interest for telecommunication installations and for installations with locally attached terminals such as IBM 2260, 2741, and 3277.

Up to eight devices can be specified to make up this pool. The operator's console (SYSLOG device) must be included in the pool along with terminal devices, card readers and diskette I/O units at the user's discretion. The list of specified devices has to be linked to the system's core image library. Use of an IPL communication device list is optional.

If the list is cataloged, DOS/VS IPL checks for its contents and accepts only interrupts from devices contained in the list. This prevents that an interrupt from a terminal outside the computing center causes such terminal to wrongly establish itself as an IPL communication device.

For more information on this new DOS/VS feature, refer to *DOS/VS System Management Guide* and *DOS/VS IPL and Job Control Logic.*

IBM 3540 as IPL Communication Device

The IBM 3540 diskette I/O unit is now supported as an IPL communication device and may be used to submit IPL commands. The IPL commands must be written one command per sector, in card image format onto a single volume diskette file.

New Procedures to Load the SVA

Two new procedures, RPS and VSAMRPS, are now available to further improve system performance when loading IBM-supplied phases into the SVA. Procedure RPS loads RPS phases, procedure VSAMRPS loads VSAM and RPS phases. Each of the two procedures, in addition, loads a set of selected system phases normally loaded by the existing procedure SDL. The procedures are cataloged in the procedure library.

New Parameter in the DLBL Statement

BLKSIZE, a new parameter in the DLBL job control statement, provides support for dynamically changing the blocking factor for a sequential disk

file on an IBM 3350 or IBM 3330-11. When transferring files to such devices, it is not necessary to change the blocksize specifications in the problem program.

)OS/VS Access Method Services Enhancements

- User-Supplied Print Chain/Train Support

  Support for previously non-supported print chain/trains (such as the KATAKANA print chain/train) is now provided. The TABLE subparameter has been added to the GRAPHICS parameter in the PARM command. This subparameter allows the user to specify a translate table that defines the graphic to be printed for a particular bit configuration.

- Page Length Improvements

  Prior to Release 34, Access Method Services always used the SET LINECT default (56) as the number of lines to be printed on SYSLST. The user can now specify any value between 30 and 99 for the number of lines to be printed on each page of SYSLST either at system generation or between jobs. If no other value has been specified, the Access Method Services prints 56 lines on each page of SYSLST.

- Tape Processing Improvements

  Options to process unlabeled tapes or to rewind or rewind and unload tapes for OPEN, CLOSE, and EOV condition have been added to the tape processing commands of Access Method Services.

:REP Enhancements

- Model 158 machine check and inboard channel check records are now supported by the EDIT and SUM options.
- Models 135 and 138 are now supported by the SUM option in a manner similar to Models 145 and 148.

'he List System History Program (HISTLIST)

This utility program provides a formatted printout of the installation's history book(s). Cross-reference lists sorted by PTFs, local fixes, APARs, and library members simplify locating installed changes in the various libraries.

:OPYSERV Function now in CORGZ

The functions of the COPYSERV program - to compare library directories and to identify missing library elements to CORGZ - are now integrated in

the librarian program CORGZ. The added functions in the CORGZ program are invoked by the new parameter NEW.

Job Accounting Improvements

The job accounting support has been improved in two ways:

- The date in the job accounting record of a job step will now always correspond to the date at the end of the job step. (Up to now, the date in the job accounting record for the last or only job step was that of the beginning of the job step.)
- The job accounting support will no longer be canceled if the job accounting routine is canceled.

POWER/VS Enhancements

SNA Enhancements

The extended POWER/VS support for RJE,SNA work stations provides support for the 3790 Communication System with RJE facilities for up to three logical printers, one logical card reader, and one console per work station on up to five sessions. The present support for 3770 SNA is continued without new features. This support includes one logical printer, one logical punch, one logical card reader, and one console with one active session. The functions provided in conjuction with the 3790 Communication System with RJE facility are:

- Concurrent device operation. Support for concurrent operation of one inbound card data stream, one inbound or outbound message, and up to three outbound printers.
- Host-to-3790 compaction for printer data. Compaction, a method of reducing the amount of data to be transmitted, improves communication line utilization beyond the improvements achieved by compression.
- Remote data spooling. Remote spooling is possible if the receiving work station is capable of accepting peripheral data stream records (PDIR). The 3790 has this capability. The PDIR transmitted by POWER/VS provides the information for remote spooling. Even if multiple copies of a job output are required at a work station, this output needs to be transmitted only once.

Non-SNA Quality Improvements

These improvements include:

- Spool tape integrity. A check is made whether the spool tape has a volume label VOL1.
- Improved FLUSH and PFLUSH commands.
- Improved RELEASE and PRELEASE commands.
- Automatic EJECT and FEED at EOF on a 3540.
- Optional logging on SYSREC for RJE,BSC lines.
- Correction of JECL at the time of job execution.
- Multiple printer channel-12 control characters per page.

Part 4. Licensed IBM Programs for DOS/VS

As its name implies, a licensed IBM program is available only through a license agreement between IBM and the user who intends to utilize the program at his installation.

A licensed program may be a self-contained program which can be executed under DOS/VS or, as in the case of Advanced Functions - DOS/VS, it may be a set of routines or modules that enhance one or more component programs of DOS/VS. Self-contained licensed programs may be categorized as follows:

- Service programs (or subsystems)
- Language translators
- Application programs (or systems)

The last category will not be dealt with in this section.

Advanced Functions - DOS/VS

This is a set of routines in source statement format plus a number of relocatable modules ready for installation on a Release 34 DOS/VS. This licensed program offers the user significant functional improvements:

- Support of up to seven partitions instead of only five.

  An enhancement that may be of particular benefit for telecommunication-oriented installations which, nevertheless, have to cope with a considerable batch work load.

- Deferred operator replies to messages.

  The operator need no longer reply to messages in the same sequence as they are displayed on the console. He can defer the reply to one message per active task in the system before DOS/VS locks the console for the transmission of messages from other tasks.

- Dynamic partition balancing.

  Two or more or all partitions may be specified to participate in partition balancing. Out of a group of partitions so specified, DOS/VS privileges those which have a low record of CPU activity during a specific time interval.

- Library device independence.

  The facility allows users of DOS/VS to allocate DASD space for private relocatable and source statement libraries on device types other than those used for system residence.

- Improved performance of the linkage editor.

A linkage editor run under DOS/VS with Advanced Function - DOS/VS installed completes faster than under DOS/VS without this program if modules stored in the relocatable library are to be included in the program that is being link-edited.

- Improved performance of DOS/VS under VM/370.

  This facility, which is known as DOS/VS - VM/370 Linkage Enhancements, improves performance of DOS/VS under VM/370 by:

  - avoiding many of the instructions that are redundant in a VM/370 environment (DOS/VS avoids functions such as seek separation, load leveling, paging as well as page fixing and page freeing; DOS/VS executes fewer privileged instructions).
  - returning control to DOS/VS immediately after VM/370 has handled a DOS/VS detected 'pseudo' page fault, thus enabling DOS/VS to dispatching (give control to) anotyher program or task that is being executed under the virtual DOS/VS.
  - avoiding a program check interrupt that previously was caused by the DOS/VS BTAM routines in order to check for modified CCWs when the BTAM CCW string was being executed.
  - automatically closing the printer and punch files that are spooled by POWER/VS. These files are printed (punched) without a specific request.

  In addition, the facility ensures that VM/370 updates the interval timer just before this timer is accessed by DOS/VS for job accounting purposes.

For more information about these functional enhancements and their usefulness at a DOS/VS controlled installation, refer to *Advanced Functions - DOS/VS, General Information,* GC33-6040.

Service Programs or Subsystems

Advanced Communication Function/VTAM (ACF/VTAM)

This licensed program provides the same functions and support as the VTAM level 2 (unlicensed VTAM) plus a number of significant enhancements. These enhancements are:

- Program-to-program communication. The facility allows the user additional flexibility in designing and implementing more efficient telecommunication applications.
- Message traffic pacing. By pacing the flow of messages between application programs and between an application program and the logical units (3770, 3776, and 3777) with which the program has established a session, ACF/VTAM can minimize congestion and prevent the exhaustion of buffer pools.

- Remote 3705-II support. Together with the ACF/NCP/VS program, this support offers the current remote 3704/3705 NCP/VS support combined with the advantages of the new capabilities of the 3705-II.
- Multiple channel attachment. Support of the multiple attachment capability of ACF/NCP/VS expands the sharability of a 3705 Communications Controller. This sharability allows, for example, one 3705-II to support up to four ACF/VTAM host networks, none of which need to have a dedicated communications controller.
- DISPLAY command enhancements. They allow a network operator (or a program operator application program) to obtain more information about the status of nodes, lines, and terminals; logical units and physical units; application programs and buffers. Such information may be used for monitoring, for performance analysis, or for determining subsequent action.
- Dynamic buffer pool allocation. ACF/VTAM can dynamically acquire storage for buffer pools according to message traffic and availability of real storage. This optimizes the use of real storage and assists a user in determining actual buffer pool requirements.
- Tuning statistics. The facility dynamically accumulates data about the I/O activity of a local communications controller or a local 3790. This data may help in selecting optimum values for start parameters and for network definition statements.
- Concurrent line tracing. Allows tracing of up to eight communications controller lines concurrently through ACF/NCP/VS and thus the determination of line utilization or error situations for several lines at the same time.
- Switched network backup. SNA-SDLC devices associated with a failed or deactivated line can continue to receive communication support via a switched backup line.
- Multisystem communication. Allows an ACF/VTAM application program in one System/370 host system to communicate, via the ACF/NCP/VS program, with the following programs or devices associated with another host system:
  - Another ACF/VTAM application program.
  - SNA-SDLC and 3270 BSC devices, if they are controlled by ACF/VTAM in their own system.
  - Start-stop, BSC, and locally attached devices if they are controlled by ACF/VTAM in their own system and if an ACF/VTAM application program in that system is available to route the data to these devices (this amounts to an extension of program-to-program communication).
- Multisystem routing. Data can flow through a network of communications controllers without having to pass through the host systems attached to the various controllers.

- Resource backup capability. Network operator commands can be issued in one host system to assume control of an ACF/NCP/VS-supported 3705 that is channel-attached to another (failing or deactivated) ACF/VTAM host system.

The enhancements described above have many potential advantages for users of ACF/VTAM under DOS/VS.

- More flexibility in application processing and in accessing information between multiple System/370 networks.
- Expansion of the scope of network facilities (such as application programs and terminals) that are available for communication of data and for processing.
- Continued operation of communication controllers and associated terminals (or devices) independently of a specific System/370 CPU.
- A wide range of multiple System/370 configurations that support orderly applications growth under the Systems Network Architecture (SNA).
- Elimination of redundant network applications.
- Interconnection of System/370 networks with the complete range of low-end to high-end CPUs and the associated operating systems.
- Consolidation of communications management functions (such as terminal ownership control, session establishment, and ACF/NCP/VS loading) in a *communications management* CPU that is connected through the multiple channel attachment capability of ACF/NCP/VS and the 3705 Communications Controller.

For more information about the functional enhancements provided by ACF/VTAM and their usefulness at a DOS/VS installation, consult these publications:

ACF/VTAM General Information Manual, GC38-0254
ACF/VTAM Concepts and Planning, GC38-0255

The *General Information Manual* describes the product in a general manner, names the programs and devices that can be used with it, and lists the steps that must be accomplished to install the product.

Concepts and Planning contains extensive descriptions of the ACF/VTAM facilities and describes the installation process in more detail. Both publications describe the entire ACF/VTAM documentation library.

Data Language I (DL/I)

DL/I is a data management control system developed to assist the user in implementing data base processing applications. The system provides data organization methods for creating, accessing, and maintaining large common data bases; it permits the expansion of data processing applications from a batch-only environment to a teleprocessing environment such as the CICS/VS.

For more details on the DL/I library, consult the publications *Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS) General Information Manual*, GH20-1246.

Customer Information Control System (CICS/VS)

CICS/VS is a transaction-oriented data base/data communication interface to user-written application programs. CICS/VS provides many of the facilities necessary for standard terminal applications. Some of the available service functions are:

- Message switching.
- Inquiry operations.
- Data collection and order entry.
- Batched and conversational data entry.

For more details on this program and the manuals available for it, consult the publication *Customer Information Control System/Virtual Storage (CICS/VS) General Information Manual*, GH20-1280.

VS Personal Computing (VSPC)

VSPC, a licensed program, enables a user to create and execute programs at a terminal.

VSPC interacts conversationally with the terminal user and is designed for use with interactive programming languages. The IBM programs VS APL, VS BASIC, and VSPC FORTRAN can be used as foreground processors with VSPC.

VSPC has:

- Interactive commands for entering, editing, and printing data or text (including source program statements). Facilities are available for authorized users to submit jobs through POWER/VS and retrieve the output for examination at the terminal.
- An online direct-access library with three kinds of user libraries to meet differing security needs, and to make a variety of data and applications available to different people.
- Supervisory commands and a batch service program for administering and controlling the use of VSPC, for converting data, for optimizing performance, and for maintaining the VSPC library.

For more information on this program and the manuals available for it, consult the publication *VS Personal Computing (VSPC) for OS/VS and DOS/VS General Information*, GH20-9070.

DOS/VS Sort/Merge

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

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

- Support of new DASDs that are not supported by DOS Sort/Merge.
- Support of key-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 and the manuals available for it, consult the *DOS/VS Sort/Merge General Information Manual*, GC33-4030.

Language Translators

DOS/VS RPG II Compiler

RPG II, an easy-to-use programming language in which a wide variety of commercial data processing applications may be implemented, is an expanded version of the RPG language provided with DOS.

The DOS/VS RPG II compiler offers performance improvements over the DOS RPG compiler in two areas:

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

The compiler includes also significant enhancements over the previous RPG II version. Some of these enhancements are highlighted below:

- Support of a defined set of VSAM functions.
- Integration of the Auto Report facility.
- System/3 RPG II equivalent functions:

- No need to specify input array decimal positions.
- TIME operation code to access the system time of day.
- PRINT operation code to have the 2560 print the contents of punched fields.
- Output specifications are now optional.
- Added device independence.

For more details on this licensed program and the manuals available for it, consult the publication *DOS/VS RPG II General Information,* GC33-6030.

DOS/VS COBOL Compiler

This compiler, a licensed program, compiles programs written in the ANS COBOL language; it is designed for use under DOS/VS. The compiler contains all the functions of the DOS COBOL compiler, Version 3, and includes additional support as follows:

- Support of VSAM functions available with DOS/VS Release 28.
- Device support also for devices that are supported by DOS/VS but not by DOS.
- The FIPS flagger, which identifies areas of a user's program that do not conform to the Federal Information Processing Standard.

Note: For COBOL D source programs to be in a form suitable for this (or the Full) ANS COBOL compiler, they can be converted by using the *COBOL-to-American National Standard COBOL Language Converion* program. However, some direct programming may still be required to accomplish full conversion. The amount of this programming varies with each application program. The compiler can run in a minimum size virtual partition of 64K.

For more information about this compiler and the manuals available for it, consult the publication *DOS/VS COBOL, General Information*, GC28-6473.

PL/I Optimizing Compiler

PL/I is a general purpose programming language which can be used to program both commercial and scientific applications. It 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.

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.

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

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

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 programs.

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. 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.

For more information about the PL/I Optimizing Compiler and the manuals available for it, consult the *PL/I Optimizer, Resident Library and Transient Library, General Information*, GC33-0004.

FORTRAN IV Library, Option 1

FORTRAN is a programming language designed for the solution of scientific and computational problems. For users of FORTRAN, the DOS FORTRAN IV compiler is available as an IBM-supplied Type I program.

The FORTRAN IV Library, Option 1, is available as a licensed program; together with the Type 1 DOS FORTRAN IV compiler, the library allows the programmer to write and have compiled FORTRAN programs that:

- access files on DASDs that have been designed for attachment to System/370 CPUs
- create and process magnetic tape files which conform to the American National Standard Code for Information Interchange (ASCII)
- use larger block sizes for EBCDIC tape records.

For more information about this FORTRAN library, consult the *FORTRAN IV Library, Option 1, Program Product Specifications*, GC28-6882.

Part 5. Configurations

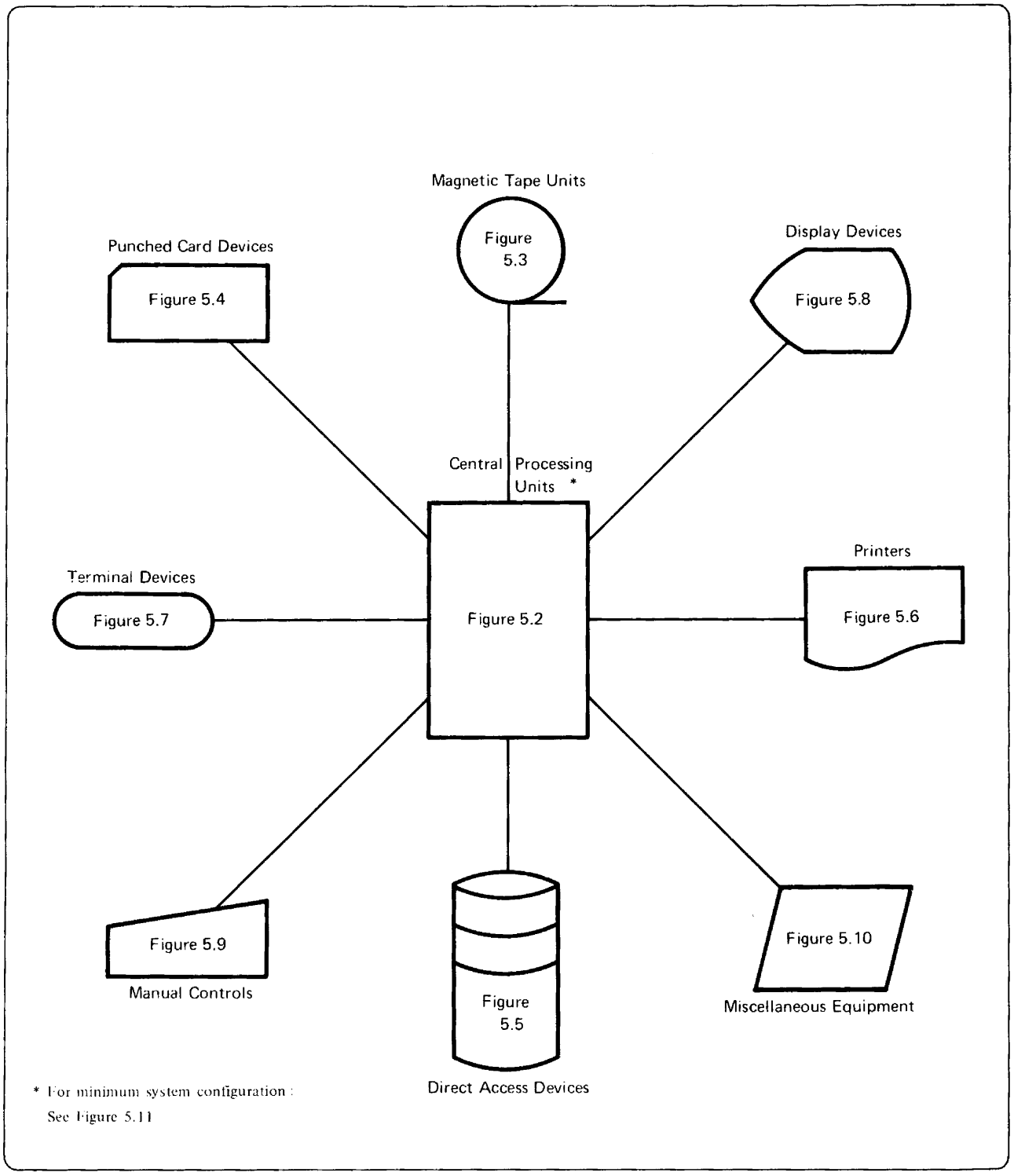


Figure 5.1. IBM System/370 Configurations Supported by DOS/VS
In each of the figures referenced there is a full list of devices supported.

System/370		Central Processing Units	
No.	Model	Storage size in bytes	Remarks
3115-0	F	65,536	
	FE	98,304	
	G	131,072	
	GE	163,840	
	GF	196,608	
3115-2	F2	65,536	
	FE2	98,304	
	G2	131,072	
	GE2	163,840	
	GF2	196,608	
	H2	262,144	
	HG2	393,216	
3125-0	FE	98,304	
	G	131,072	
	GE	163,840	
	GF	196,608	
	H	262,144	
3125-2	FE2	98,304	
	G2	131,072	
	GE2	163,840	
	GF2	196,608	
	H2	262,144	
	HG2	393,216	
	I2	524,288	
3135-0	FE	98,304	
	GD	147,456	
	GF	196,608	
	DH	245,760	
	H	262,144	
	HF	327,680	
	HG	393,216	
	I	524,288	
3135-3	A01	262,144	
	A02	327,680	
	A03	393,216	
	A04	524,288	
3138	J01	524,288	
	J01	1,048,576	
3145-0	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	

Figure 5.2. Central Processing Units (Part 1 of 2)

System/370		Central Processing Units	
No.	Model	Storage size in bytes	Remarks
3145-3	A01	196,608	
	A02	327,680	
	A03	458,752	
	A04	720,896	
	A05	983,040	
	A06	1,507,328	
	A07	2,031,616	
3148	J01	1,048,576	
	K01	2,097,152	
3155-II	H	262,144	
	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 (Part 2 of 2)

System/370		Magnetic Tape Devices				
No.	Mode	Name	Maximum Data Rates		Remarks	Control Unit
			Kilobytes per second	Bytes per inc		
2401	1	Magnetic Tape Unit	30	800		
	2	Magnetic Tape Unit	60	800		
	3	Magnetic Tape Unit	90	800		
	4	Magnetic Tape Unit	60	1600		2803 or 2804
	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	Not attachable to a Model 115/125	
	4-6	Magnetic Tape Unit and Control	30	1600		none
2420	5	Magnetic Tape Unit	160	1600		2803
	7	Magnetic Tape Unit	320	1600		
2495	1	Tape Cartridge Reader	0.9	20	Not supported by POWER/VS	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		Note 2
	4	Magnetic Tape Unit	470	6250	Not attachable to a Model 115/125	Note 3
	5	Magnetic Tape Unit	200	1600		Note 2
	6	Magnetic Tape Unit	780	6250	Not attachable to a Model 115/125	Note 3
	7	Magnetic Tape Unit	320	1600		Note 4
	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, 2, and 3. Model 3 is needed for attachment of the 115 or 125.
Note 3: Attaches to the 3803 Model 2 only.
Note 4: Attaches to the 3803 Models 1 and 2.

Figure 5.3. Magnetic Tape Units

System/370		Punched Card Devices					
No.	Mode	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					
	B2	Card Reader					
			600	-	-	none	
			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	160	-	none	For Models 11 and 125 only. (See Note 2 and 3)
2596	1	Card Read Punch (Note 1)	500		120	none	96-column card machine
3504	A1	Card Reader	800			none	For Model 125 only (See Note 2)
	A2	Card Reader	1200				
3505	B1	Card Reader	800	-	-	none	
	B2	Card Reader	1200	-	-		
3525	P1	Card Punch	100	-	-	3505	See Note 2
	P2	Card Punch	200	-	-	Card	
	P3	Card Punch	300	-	-	Reade	
5425	A1	Multifunction Card Unit	250	-	60	none	For Models 11 and 125 only. (See Note 2 and 3)
	A2	Multifunction Card Unit	500	-	120	none	96-column card machine.

Note 1: DOS/VS does not support SYSIPT, SYSRDR, or 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 or
3) One 3504 and one 3525.

Note 3: Either one 2560 or one 5425 can be attached to a Model 115 or Model 125.

Figure 5.4. Punched Card Devices

System/370		Direct Access Devices			
No.	Model	Name	Million Bytes Capacity (Max.)		
			Drive	Unit	See Note
2311	1	Disk Storage Drive	7	7	1, 2, 4
2312	A1	Disk Storage	29	29	1, 2
2313	A1	Disk Storage	29	117	1, 2
2314	A1, B1	Direct Access Storage Facility	29	233	1, 2
2318	A1	Disk Storage	29	58	1, 2
2319	A1, A2, A3	Disk Storage	29	87	1, 2
2219	B1, B2	Disk Storage	29	87	1, 2
2321	1	Data Cell Drive	400	400	1, 2, 4
3330	1	Disk Storage	100	200	1
3330	2	Disk Storage	100	100	1
3330	11	Disk Storage	200	400	1, 2
3333	1	Disk Storage and Control	100	200	1
3333	11	Disk Storage and Control	200	400	1, 2
3340	A2	Direct Access Storage Facility	70	140	5
3340	B1	Direct Access Storage Facility	70	70	5
3340	B2	Direct Access Storage Facility	70	140	5
3344	B2, B2F	Direct Access Storage	280	560	3, 6
3350	A2, A2F	Direct Access Storage	317	635	1, 2, 7
	B2, B2F	Direct Access Storage	317	635	1, 2, 7
	C2, C2F	Direct Access Storage	317	635	1, 2, 7

Except as noted, these devices attach to all DOS/VS supported IBM System/370 CPU Models.
For specific configuration capabilities and attachment prerequisites, refer to the applicable hardware manual.

Note 1: Not attachable to IBM CPU Model 115.

Note 2: Not attachable to IBM Model 125.

Note 3: Not attachable to IBM CPU Model 115-0 or 125-0.

Note 4: Supported only as a data storage device.

Note 5: The 3340 uses 3348 data modules for storage. Models 35 and 70 operate on all 3340 drives, Model 70F operates only on 3340 drives with the Fixed Head Feature installed.

Note 6: One Head/Disk Assembly (HDA) of the IBM 3344 Direct Access Storage is equivalent to four (4) 3340s with 3348 Model 70 data modules mounted.

Note 7: The IBM 3350 in 3330-1 compatibility mode is equivalent to two (2) 3330-1 volumes on a single non-removable Head/Disk Assembly (HDA); in 3330-11 compatibility mode, it is equivalent to one (1) 3330-11 volume.

Figure 5.5. Direct Access Devices

System/370		Printers			
No.	Model	Name	Control	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	
3203	1 2 4	Printer Printer Printer	none none none	600 lines per minute 1200 lines per minute 1200 lines per minute	For Models 115 and 125 only For Models 138 and 148 only
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 (Note 1)
5213	1	Console Printer	none	85 chars. per second	For Model 125 (Note 2) and Model 115
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 5203 mode.

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

Figure 5.6. Printers

System/370			Terminal Devices	
No.	**Mode**	**Name**	**Control Unit**	**Remarks**
1030		Data Collection System	2701 2702	
1050		Data Communication System	2703 3704	
1060		Data Communication System	3705	
2721		Portable Audio Terminal	7770	
2740	1, 2	Communication Terminal	2701 2702	
2741		Communication Terminal		
2760		Optical Image Unit	2703 3704 3705	
2770		Data Communication System	2701 2703 3704	
2780	1 - 4	Data Transmission Terminal	3705	
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	
2980		General Banking Terminal System	3704 3705	
3650		Retail Store System	3704	
3660		Supermarket Store System	3705	
3270		Information Display System	3271	Remote attachment (Note 2)
3270		Information Display System	3272	Local attachment
3270		Information Display System	3275	Remote attachment (Note 2)
3600		Finance Communication System	3704 3705	
3735		Programmable Buffered Terminal	2701 2703 3704 3705	

Note 1: 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 of the specified devices can be attached via the ICA, except the 7700, 3790, and, when using synchronous data link control the 3650, 3660, 3667, and 3770.

Note 2: For remote attachment, a 2701, 2703, 3704, or 3705 is required.

Note 3: POWER/VS supports the 3741 as a 2780 without the multiple record transmission feature.

Figure 5.7. Terminal Devices (Part 1 of 2)

System/370		Terminal Devices		
No.	Mod	Name	Control Unit	Remarks
3740 3767 3770 3790		Data Entry System Communication Terminal Data Communication System Communication System	3704 3705	
3741	2 4	Data Station Programmable Terminal	3704 3705	Notes 2 and 3
3780		Data Communication Terminal		
		IBM Communicating Magnetic Card Selectric Typewriter	2701 2703 3704 3705	
		IBM World Trade Teletypewriter Terminals (WTTY)		

Note 1: 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 can be attached via the ICA, except the 7700, 3790, and, when using synchronous data link control the 3650, 3660, 3667, and 3770.

Note 2: For remote attachment, a 2701, 2703, 3704, or 3705 is required.

Note 3: POWER/VS supports the 3741 as a 2780 without the multiple record transmission feature.

Figure 5.7. Terminal Devices (Part 2 of 2)

System/370		Display Devices			
No.	Mod	Name	Characters	Control Unit	Remarks
2260	1 2	Display Station Display Station	960 480	2848*	Local or remote attachment
2265		Display Station	960	2845*	For remote attachment only
3277**	1 2	Display Station Display Station	480 1920	3271, 3272	3271 if remotely attached; 3272 if locally attached.

* For remote attachment a 2701 control unit is required.

** When used as DOC, the 3277 must not be attached to a selector channel.

Figure 5.8. Display Devices

System/370		Manual Controls			
No.	Mod	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.

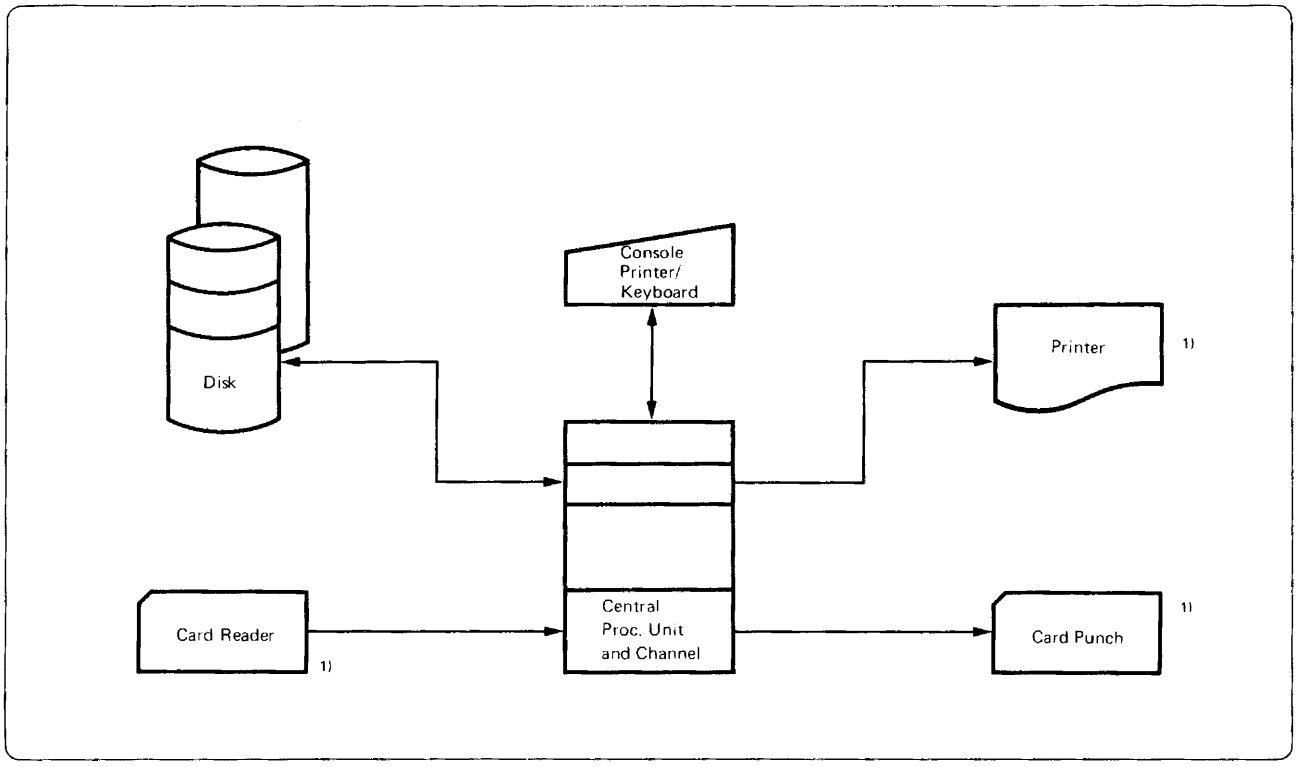
Figure 5.9. Manual Controls

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 Reader Sorter	750 dpm	none	
1275	2,4	Optical Reader Sorter	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	
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 local IBM representative.

** 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 the printer can each be replaced by a magnetic tape unit or by a disk or diskette extent. This does not apply to the card reader during IPL, unless the user has a cardless system.

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. Consult your IBM representative for the offerings available to you from the IBM education center.

The DOS/VS Library

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

Topical Groups in the Library

The library can be divided into the topical groups as follows:

System Generation
Maintenance
System Control and Service
DOS/VS POWER/VS
Data Management
Operation
Teleprocessing
Emulation
Assembler

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

Types of Manuals in the Library

Wherever appropriate in the 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 system. In developing a topic, a descriptive manual or section attempts to address practical implications by means of examples and careful explanation.

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.

3. **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.

Manuals for Licensed IBM Programs

The *IBM System/370 Bibliography,* GC20-0001, lists the specific manuals available with each licensed IBM program.

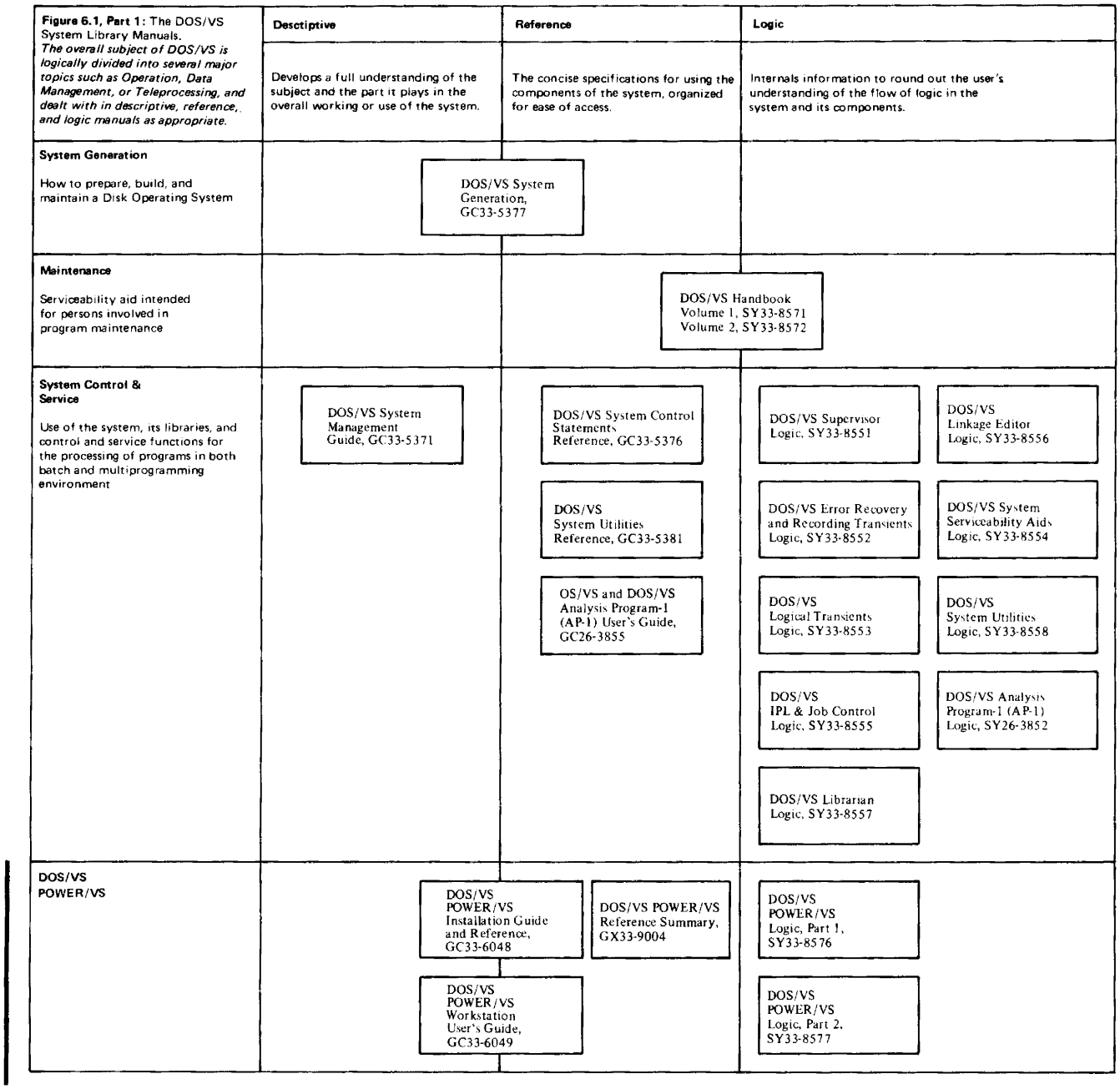


Figure 6.1. DOS/VS Documentation (Part 1 of 3)

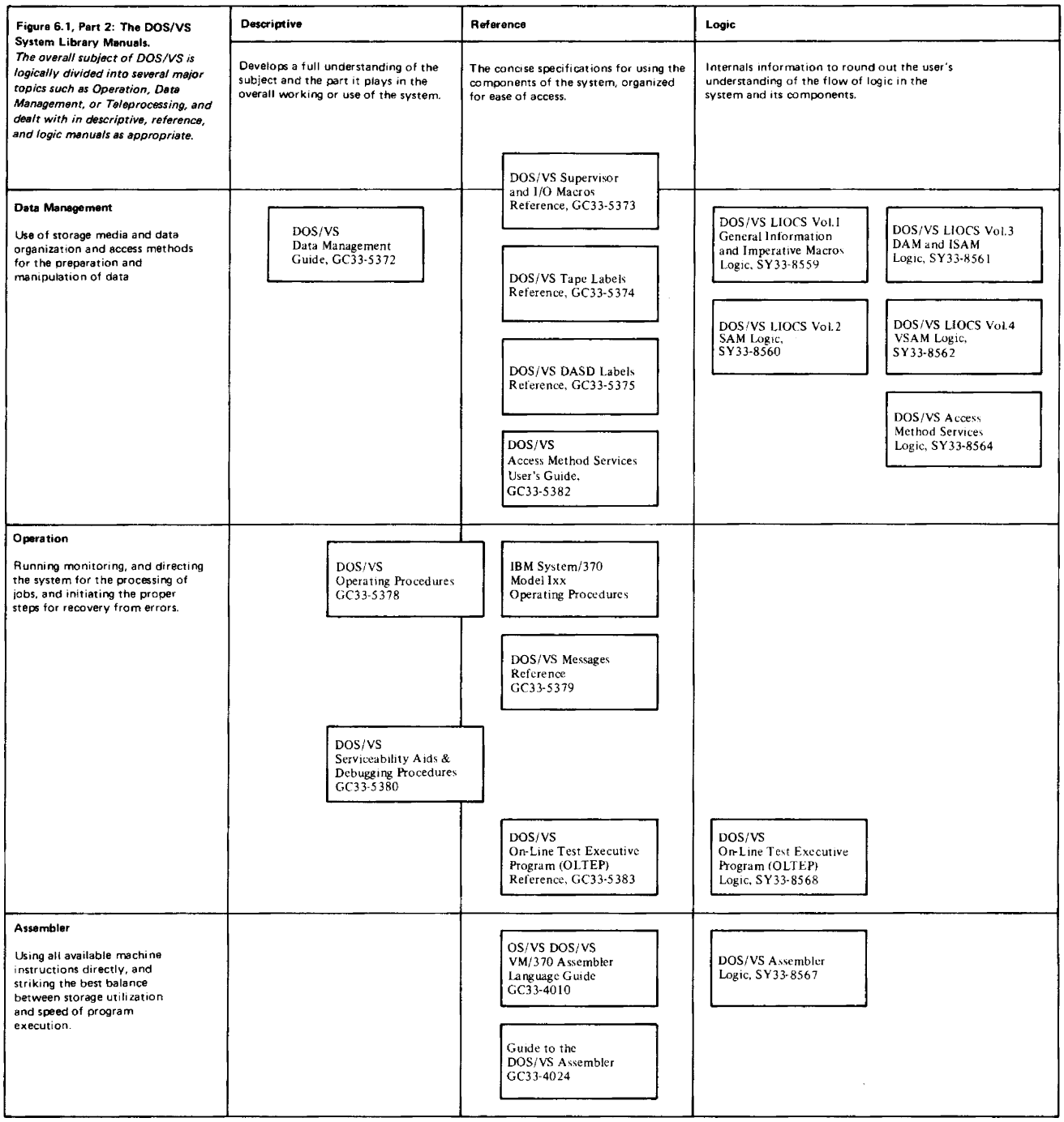


Figure 6.1. DOS/VS Documentation (Part 2 of 3)

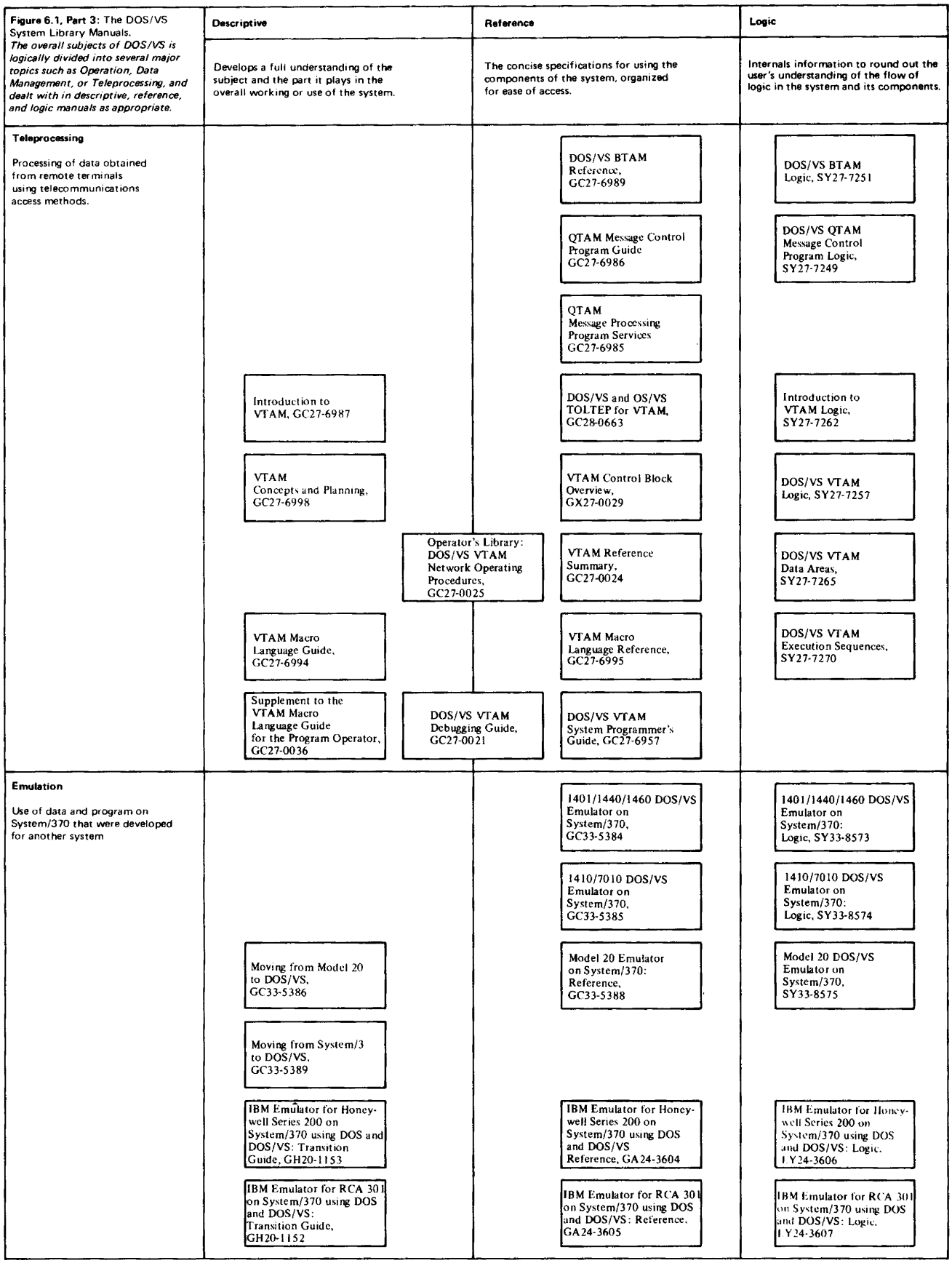


Figure 6.1. DOS/VS Documentation (Part 3 of 3)

Glossary

This glossary defines the terms proper to an introductory manual on DOS/VS. If you do not find the term you are looking for, refer to the index or to the *IBM Data Processing Glossary,* GC20-1699.

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

ACF/NCP/VS: An enhanced program version of the Network Control Program/Virtual Storage (NCP/VS). ACF/NCP/VS is a control program for the 3705 Communications Controller.

ACF/VTAM: An enhanced program version of the Virtual Telecommunications Access Method (VTAM). An optional feature of ACF/VTAM offers communication between application programs in one System/370 CPU with SNA-SDLC devices associated with another System/370 CPU.

***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.

American National Standard: ANS.

ANS: American National Standard.

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.

asynchronous operator control: A DOS/VS facility that allows the operator to defer the reply to a system message which requires a response.

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.

batch partition: Partition in which batch processing takes place.

batch processing: Sequential processing of computer programs, submitted to the computer as a collection (batch) of jobs that are separated from one another by job control statements.

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.

cardless system A System/370 Model 115/125 configured without a card reader or card punch, but with an IBM 3540 Diskette Input/Output Unit.

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.

Communications Controller: A device that controls (in conjunction with its ACF/NCP/VS or NCP/VS program) the transmission of data over lines in a telecommunication network. The 3705 Communications Controller, for example, handles the routine aspects of communication line protocol between itself and the terminals that are attached to it, thus reducing the number of trivial interruptions that must be serviced by the CPU. The 3705 Communications Controller, in conjunction with an ACF/NCP/VS program can route data to other 3705 Communications Controllers attached to it by communication lines.

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 A flexible magnetic oxide coated disk suitable for data storage and retrieval.

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.

dynamic parition balancing: A DOS/VS facility which allows the user to specify two, more, or all partitions of the system to have their processing priority changed dynamically such that each of these partitions receives approximately the same amount of CPU processing time.

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.

pacing: A procedure by which the telecommunications access method controls the rate at which data is received by application programs in the CPU or by the logical units associated with SNA-SDLC terminals. Pacing is intended to protect the application program or logical unit that is receiving data from being overrun with too much input.

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.

private second level directory: A table located in the supervisor and containing the highest phase names found on the corresponding directory tracks of the private core image 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. Commonly referred to as processor, main, CPU, or internal storage.

reenterable: The attribute of a load module that allows the same copy of the load module to be used concurrently by two or more tasks.

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.

rotational position sensing: A feature which permits certain DASD devices to disconnect from a block multiplexer channel during rotational positioning operations, thereby allowing the channel to service the other devices on the channel during the positioning delay.

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

secondary storage: Same as auxiliary storage.

second level directory: A table located in the supervisor and containing the highest phase names found on the corresponding directory tracks of the system core image library.

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.

SNA: See **System Network Architecture.**

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.

System Network Architecture (SNA): The total description of the logical structure, formats, protocols, and operational sequences for transmitting information units through the communication system. The structure of SNA allows the ultimate origins and destinations of information - that is, the end users - to be independent of and unaffected by the specific communciation-system services and facilities used for information exchange.

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.

Virtual Telecommunications Access Method (VTAM): A set of IBM programs that control communications between terminals and application programs.

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.

VTAM: Virtual Telecommunications Access Method.

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

3705 Communications Controller: See communications controller.

Index

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Introduction to DOS/VS Printed in U.S.A GT00-0474-0 (formarly GC33-5370-5)

IBM

Introduction to DOS/VS

GT00-0474-0

READER'S COMMENT FORM

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address:

_____

_____

_____

Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

GT00-0474-0

leader's Comment Form

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 824
1133 Westchester Avenue
White Plains, New York 10604

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

Cut or Fold Along Line

Introduction to DOS/VS (File No. S370-20) Printed in U.S.A. GT00-0474-0

Introduction to DOS/VS

GT00-0474-0

READER'S COMMENT FORM

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address:

_____

_____

_____

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

GT00-0474-0

eader's Comment Form

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 824
1133 Westchester Avenue
White Plains, New York 10604

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

Cut or Fold Along Line

Introduction to DOS/VS (File No. S370-20) Printed in U.S.A. GT00-0474-0