GC28-0661-1
File No. S370-34

Systems

Introduction to OS/VS2 Release 2

IBM

First Edition (March, 1973)

This edition is a reprint of GC28-0661-0 incorporating some editorial changes.
It does not obsolete GC28-0661-0.

This edition applies to Release 2 of OS/VS2 and to all subsequent 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/360 and System/370 Bibliography**, Order No. GA22-6822, and the current **SRL Newsletter**, Order No. GN20-0360, for the editions that are applicable and current.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Publications Development, Department D58, Building 706-2, PO Box 390, Poughkeepsie, N.Y. 12602. Comments and suggestions become the property of IBM.

Preface

This publication contains introductory information about OS/VS2 Release 2, a system control program (SCP) that features virtual storage, multiprogramming, multiprocessing, time sharing, and job entry subsystems. It is assumed that readers have a basic knowledge of programming systems such as OS/MVT or OS/VS2 Release 1.

This publication is for planning purposes only. The functions and capabilities that it describes reflect the information that is currently available. This information is subject to change before the availability of OS/VS2 Release 2. Further information will be published when OS/VS2 Release 2 becomes available.

Although this manual is designed to be read from cover to cover, the chapters are generally independent and can be referred to in any order. The chapters contain the following information:

Introduction -- presents some of the system highlights.

Features and Facilities -- discusses the system functions from a user's point of view.

Design Concepts -- shows sequence of operation and other highlights of system design.

System Requirements -- lists the basic hardware requirements.

Compatibility -- points out the major differences from MVT and from VS2 Release 1 and discusses general restrictions related to upward compatibility.

Glossary -- defines VS2 terms used in this manual.

Also available, the following publications introduce specific features of the system:

Introduction to Virtual Storage in System/370, GR20-4260

Introduction to VTAM, GC27-6987

Information in this publication related to the JES3 job entry subsystem, VTAM (Virtual Telecommunications Access Method), and TOLTEP (Telecommunications Online Test Executive Program) is for advance planning purposes. These items will not be included in the initial version of VS2 Release 2.

Contents

Figures

Introduction

OS/VS2 Release 2 is a virtual storage operating system with multiprogramming, multiprocessing, time sharing (TSO), and job entry subsystems. It includes significant new features as well as many enhancements to the facilities already provided in OS/MVT and OS/VS2 Release 1, and it is a generally compatible extension to those systems.

VS means *virtual storage*, the basic concept of the system, where each job can use storage space that far exceeds the amount of real (main) storage. The term *real storage* is used to designate the main storage of System/370, whereas *virtual storage* refers to its complete addressing range.

A Growth System...

VS2 Release 2 supports System/370 Models 145, 155II, 158, 165II, and 168. Model 158 (Figure 1) and Model 168 (Figure 2) include multiprocessing models.

VS2 for System/370 evolves from and extends the capabilities of the operating system used for System/360. It uses the System/370 dynamic address translation hardware to provide up to 16,777,216 bytes of virtual storage, which the user shares with the system control program. VS2 Release 2 is generally upward compatible with MFT, MVT, VS1, and VS2 Release 1; that is, object programs and load modules that operate in those systems and that follow IBM programming conventions will usually operate without change in VS2 Release 2.

A minimal VS2 Release 2 batch and time sharing system, with the JES2 job entry subsystem, is designed to operate concurrently in 768K bytes of real storage.

A Growth System...With Flexibility

The VS2 Release 2 system is designed to handle a variety of work in different environments, make use of available resources, bring enhanced multiprocessing forward from MVT, enhance time sharing (TSO) and multiprogramming, and add the capabilities of two job entry subsystems.

Multiprocessing with two tightly coupled Model 158s or Model 168s is designed to provide improved availability as well as flexibility (over two uniprocessors) by pooling resources such as real storage, auxiliary storage, and central processing units. Inter-processor communication and alternate path I/O control are designed to add to multiprocessing availability. The two processors can also operate separately as independent systems.

The choice of a backup system is also more flexible. With virtual storage, programs designed for large systems can be run on systems with less real storage, so storage size of the backup system is less of a consideration.

As in MVT, the significance of multiprogramming is that it provides concurrent execution of jobs and an opportunity for better use of resources. In a virtual storage system, these advantages are increased by dynamically moving pages of a job in and out of real storage, rather than dedicating enough real storage for the whole job.

A number of other features of VS2 Release 2 are designed to improve the use of resources. These include job entry subsystems generally compatible with HASP II and ASP Version 3, new resources management facilities, and extension of the dynamic data set allocation capability to all jobs.

In addition, an access method called VSAM (Virtual Storage Access Method) is designed to offer more function and flexibility to online and data base environments. In VS2 Release 2, the system catalog structure has been converted to a key-sequenced VSAM data set.

VTAM (Virtual Telecommunications Access Method) is a new direct-control teleprocessing access method whose facilities are available to applications programs, including those using TCAM.

VIO (Virtual I/O), a new way of processing temporary data sets, is accomplished on a page basis in virtual storage and allows a page size blocking factor.

A Growth System...With Productivity

In VS2 Release 2, the virtual storage concept is extended to multiple virtual storage, giving each user a private address space. Generally, this is more address space for jobs than was provided in the fixed real storage regions of MVT or the virtual regions of VS2 Release 1. The programmer can concentrate on the application itself with less concern for programming practices such as overlays. Furthermore, with virtual storage available during program design, applications intended for large machines can be tested on smaller machines.

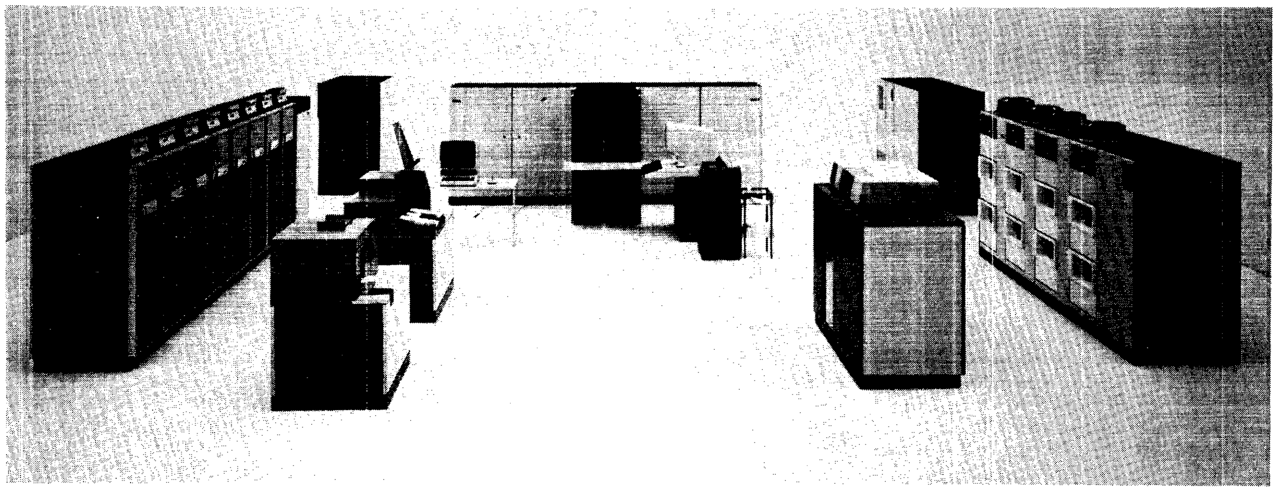


Figure 1. System/370 Model 158 (Design Model)

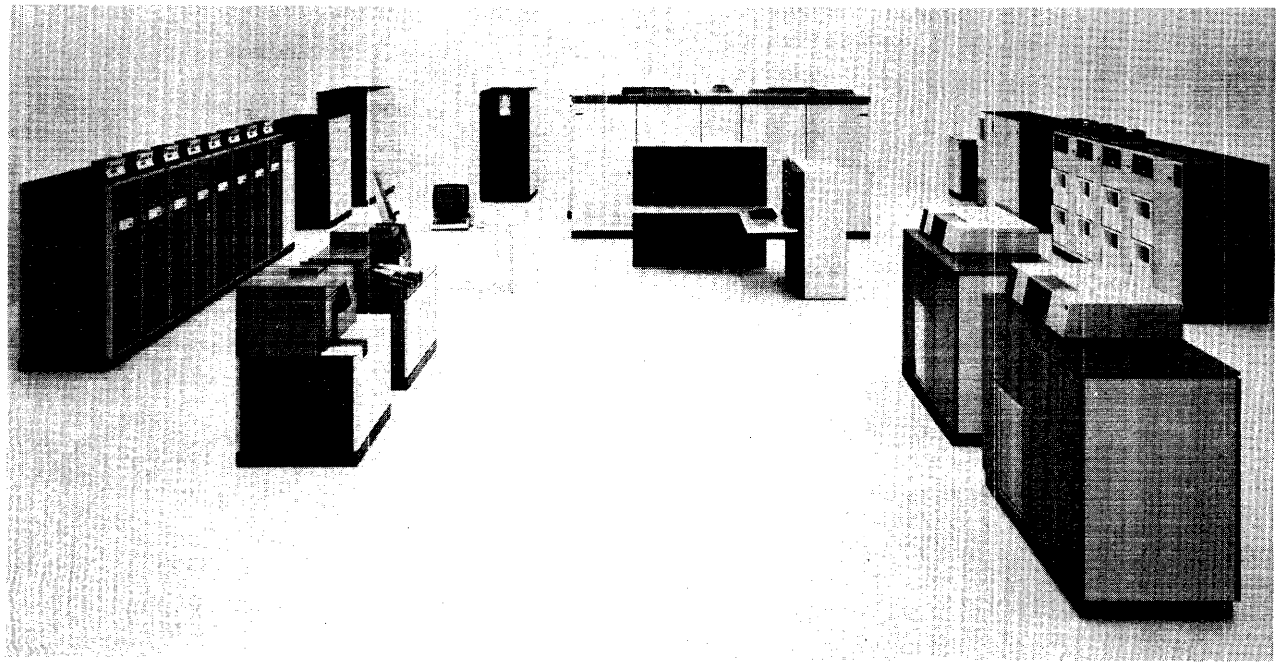


Figure 2. System/370 Model 168 (Design Model)

In multiprogramming systems without virtual storage, it was necessary to plan carefully to avoid fragmentation of real storage. Virtual storage eliminated most of this fragmentation; inter-region virtual storage fragmentation is eliminated in VS2 Release 2. Fragmentation, therefore, is no longer a consideration when selecting jobs to be multiprogrammed.

For the operator, VS2 Release 2 includes facilities aimed at increasing the speed and flexibility of system initialization. Most of the changes are designed to decrease the amount of operator intervention.

A Growth System...With Integrity

One of the most desirable properties of an operating system is its ability to ensure that one program cannot interfere with or modify another program's (system or other users) intended execution unless it is authorized to do so. From a reliability and system availability viewpoint, this capability is extremely important; from a data security viewpoint, it is essential. VS2 Release 2 provides this capability in the form of system integrity support.

All known system integrity exposures have been removed in VS2 Release 2. Any additional integrity exposures that are identified in VS2 Release 2 will be accepted as valid APARs (see *System Integrity* in the Features and Facilities section).

A Growth System...With Performance

The performance of VS2 Release 2 depends on many factors including system configuration, available real storage, and type of workload; because of these variables, no specific performance statements can be made. (See *Performance* in the Compatibility section).

Some of the changes related to performance characteristics are described in the following paragraphs. Further details can be found in other sections of the manual.

Turnaround Response

Since all user jobs are handled by one of the job entry subsystems, the installation can establish separate priorities for the selection, execution, and distribution of jobs. In VS2 Release 2, the installation can also specify service rates for different groups of users. With these facilities, it is possible to ensure that selected user jobs get very good response time in relation to other user jobs.

Use of Resources

VS2 Release 2 is designed to give each user a "fair share" of the resources, as determined by the requested service rate, priority, and system workload level.

A revised I/O load balancing algorithm attempts to dynamically adjust the use of I/O resources to correct detected imbalances.

Resource allocation in VS2 Release 2 has been designed to provide unserialized processing of the most common types of device allocation requests.

System Design

In VS2 Release 2 the use of multiple copies of virtual storage may lead to a greater degree of multiprogramming with more jobs executing concurrently. The system adjusts the level of multiprogramming dynamically to maintain paging at a defined level. A page-replacement algorithm considers users selectively with the intention of reducing paging frequency on a per user basis.

Many system functions are located in virtual storage so that they can be shared by all users. These reenterable functions eliminate the need for duplicate copies of system code for each user, reduce the need to access external system libraries, and save the time required to load these functions with each program.

In addition, a new job scheduler work area in virtual storage replaces the job queue data set and contains the majority of the user's scheduler control blocks. This is designed to reduce job scheduling serialization and the time required for allocation and for opening and closing data sets. If sufficient real storage exists or the frequency of use is high enough, the scheduler work area pages will reside in real storage, reducing paging activity.

The new catalog structure in VS2 Release 2 can handle both VSAM and previous data set types. It will permit the use of multiple private catalogs and the master catalog in one step, job, or session. The catalog structure is intended to provide reduced access time to very large catalogs.

IBM Program Products

Program products are available from IBM for a license fee. For further information about program products, see your local IBM sales representative.

Features and Facilities

This section provides a general description of the various functions of VS2 Release 2 with special emphasis on the features and facilities that are new or changed in this release. The following items are covered:

- System features such as virtual storage, multiprocessing, and time sharing.
- Functions such as handling the job stream, handling data sets, operating the system, and managing the system.
- Facilities provided for reliability, availability, and serviceability (RAS), and other support programs.

The VS2 Release 2 System

VS2 Release 2 is a virtual storage system with time-sharing and multiprocessing support integrated into the control program. In addition, virtual storage, time sharing, and multiprocessing support are enhanced in Release 2.

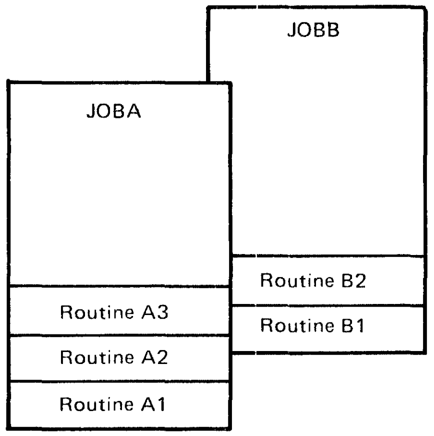
A Virtual Storage System

Virtual storage is simply an address range that can far exceed the actual range of addresses in real (main) storage. To the programmer, virtual storage appears as real storage; therefore, a programmer can write programs for the capacity of virtual storage, not real storage, and the frustration of limited real storage is greatly diminshed.

In VS2, virtual storage is an address range of 16,777,216 bytes. Release 1 of VS2 provided one virtual address range; jobs were assigned regions from this address range just as jobs were assigned regions in real storage in MVT. VS2 Release 2 provides multiple virtual address spaces: each user (i.e., background job, foreground job, and certain system components) receives its own copy of virtual storage, minus the space used for certain system functions (e.g., the nucleus).

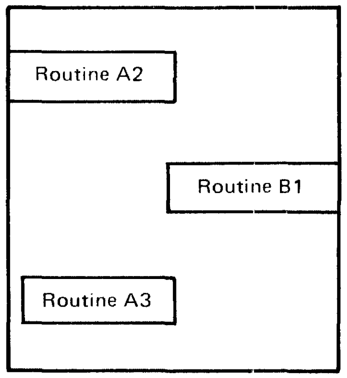
Programs are actually stored on auxiliary storage, called *external page storage*, which is divided into 4K blocks called *slots*; similarly, programs themselves are divided into 4K blocks called *pages* and real storage is divided into 4K blocks called *frames*. The system transfers pages of programs from external page storage to real storage as they are required for execution, automatically translating the program's virtual storage addresses to actual addresses in real storage. The pages do not necessarily exist contiguously in real storage. This *paging* is transparent to the user. Figure 3 illustrates private address spaces in VS2 Release 2 and the relationship between virtual storage, external page storage, and real storage.

The new virtual storage design provides the potential for greater multiprogramming, including the concurrent execution of more large jobs and a greater mix of time-sharing and background jobs. Also, since each job is isolated in a private address space, inter-region virtual storage fragmentation is eliminated and protection features are extended.



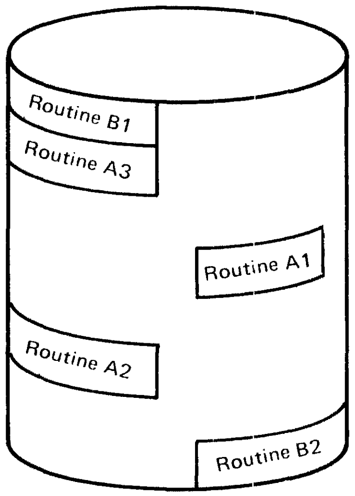
JOBA and JOBB in Virtual Storage

Both JOBA and JOBB have their own private address space -- to the user it appears as if the jobs exist in contiguous space in real storage.



JOBA and JOBB in Real Storage

The pages containing the routines of JOBA and JOBB that are currently being executed exist in real storage.



JOBA and JOBB in External Page Storage

JOBA and JOBB are actually stored on auxiliary storage. Instructions and data are transferred to real storage as required for execution.

Figure 3. Relationship Between Virtual Storage, Real Storage, and External Page Storage

A Multiprocessing System

A multiprocessing system is a computing system that uses two or more connected processing units to execute programs simultaneously. Until the advent of multiprocessing, a computer installation had two possible growth paths: to increase the size of a single uniprocessor or to use more than one uniprocessor. Both growth paths had distinct advantages. Multiprocessing combines and extends many of these advantages, as illustrated in Figure 4.

In VS2 Release 2, the programming support for multiprocessing is incorporated into the system control program, providing for easier growth to larger and more complex systems. Release 2 will also provide flexibility in the way CPUs can be combined to form a multiprocessing system; basically, the two types of combinations are:

- Loosely coupled multiprocessing.
- Tightly coupled multiprocessing.

In a loosely coupled multiprocessing system, up to four VS2 Release 2 processors (any of which can be a multiprocessor) can be connected by means of channel-to-channel adapters, which are used to pass control information between the processors. One processor controls the activity of all the processors in the system. Job entry subsystem 3 (JES3), which is a successor of ASP, provides the support for loosely coupled multiprocessing; loosely coupled multiprocessing is described in greater detail under "Job Entry Subsystem 3" in the section "Handling the Job

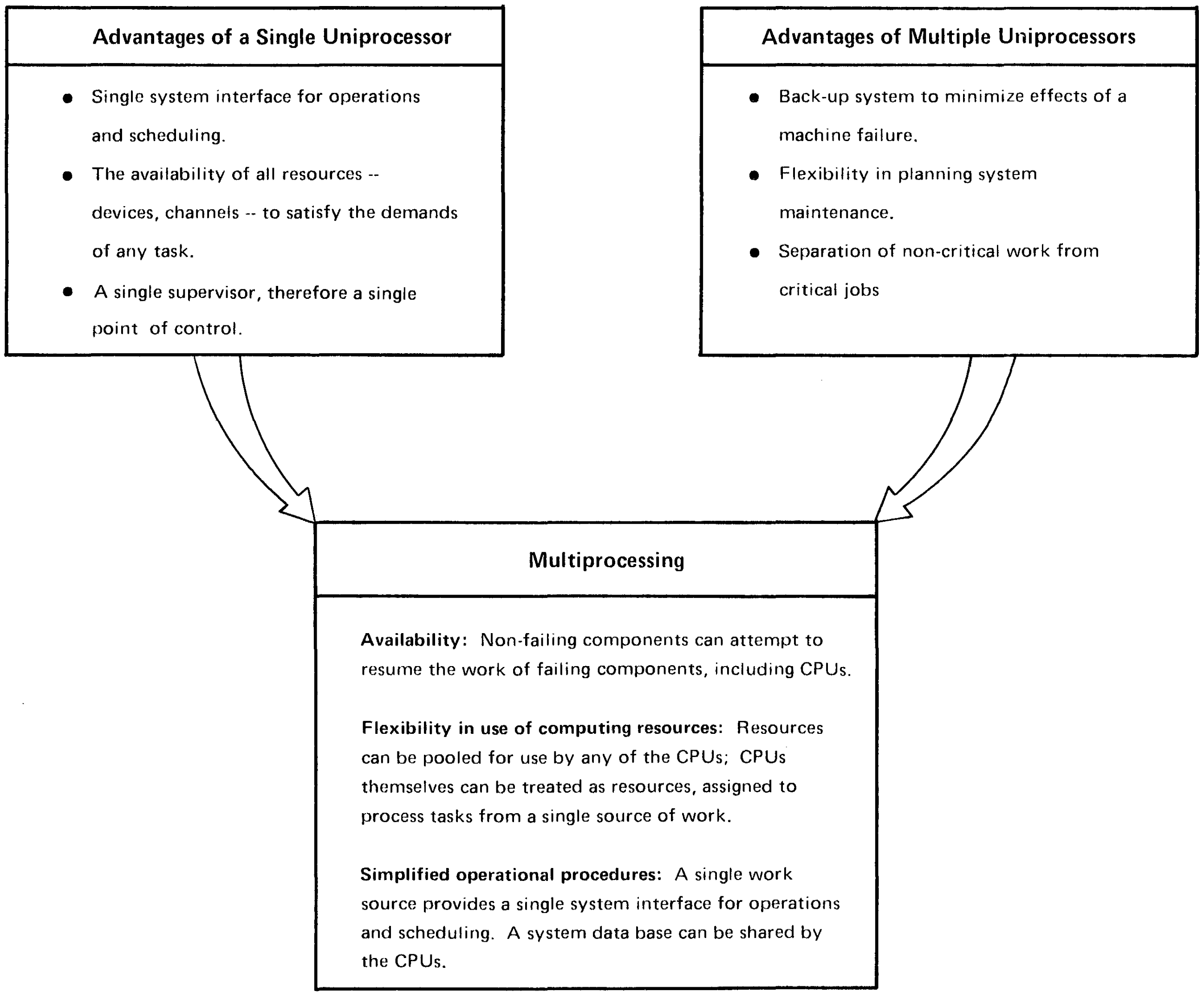

Advantages of a Single Uniprocessor
Single system interface for operations and scheduling.
The availability of all resources -- devices, channels -- to satisfy the demands of any task.
A single supervisor, therefore a single point of control.
Advantages of Multiple Uniprocessors
Back-up system to minimize effects of a machine failure.
Flexibility in planning system maintenance.
Separation of non-critical work from critical jobs
Multiprocessing
Availability: Non-failing components can attempt to resume the work of failing components, including CPUs.
Flexibility in use of computing resources: Resources can be pooled for use by any of the CPUs; CPUs themselves can be treated as resources, assigned to process tasks from a single source of work.
Simplified operational procedures: A single work source provides a single system interface for operations and scheduling. A system data base can be shared by the CPUs.

Figure 4. Combining Advantages of a Single Uniprocessor and Multiple Uniprocessors in a Multiprocessing System

Stream." Tightly coupled multiprocessing is described in the following paragraphs.

Tightly Coupled Multiprocessing

Tightly coupled multiprocessing supports either two Model 158 processors or two Model 168 processors with shared real storage. The two CPUs operate under a single control program and communicate with each other by means of an interrupt capability. The system control program treats the CPUs as resources, assigning them to process tasks. The two CPUs can be partitioned, and, while one continues operation, the other can be reinitialized. In this case, the two processors operate as independent systems. Figure 5 illustrates a tightly coupled multiprocessing system.

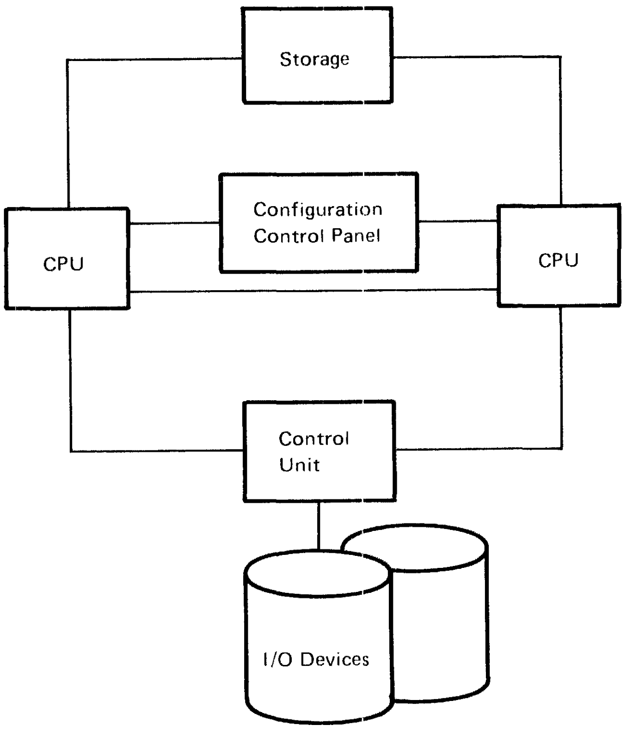


Figure 5. A Tightly Coupled Multiprocessing System

The following improvements to multiprocessing are included in Release 2:

- VS2 Release 2 provides for less disabled code through a new locking structure in the control program. (This locking structure is described in greater detail in the "Design Concepts" section under "Locking.")
- Real storage reconfiguration allows the isolation and bypassing of failing storage components in 4K blocks; only affected tasks are terminated.
- Recovery management is designed to reduce the impact of software and hardware failures.
- Asymmetric I/O support allows I/O devices that are physically attached to only one CPU to be available to jobs executing on either CPU.
- CPU affinity allows users to direct programs to a specific CPU.

A Time-Sharing System

Time sharing, which is an option (TSO) in OS/MVT and VS2 release 1, is a standard feature of the control program in release 2. All the facilities available with TSO are standard facilities with time sharing in release 2, with the exception of the command language, which remains optional.

In general, time sharing provides the following capabilities:

- It gives users access to the system through an optional *command language* which is entered at *remote terminals* -- typewriter-like keyboard-printer or keyboard-screen devices connected through telephone or other communication lines to the computer.
- It gives those who may not be programmers the use of data entry, editing, and retrieval facilities.
- It makes the facilities of the operating system available to programmers at remote terminals to develop, test, and execute programs conveniently, without the job turnaround delays typical of batch processing. Both terminal-oriented and batch programs can be developed at terminals.
- It allows the management of an installation to dynamically control the use of the system's resources from a terminal.
- It creates a time-sharing environment for terminal-oriented applications. Some applications, such as problem-solving languages, terminal-oriented compilers, and text-editing facilities, are available as IBM program products. Installations can add others suited to their particular needs.

Release 2 of VS2 enhances time sharing in the following ways:

- Each time-sharing job is assigned its own private address space, just as each background job is, when the user logs on the system.
- As a result of more efficient storage management, more time-sharing users may be active. In MVT and Release 1, time-sharing regions must be large enough to accommodate the largest user, so that much space is wasted for smaller users.

The data set handling commands are extended to allow allocation of multi-volume and multi-unit data sets, non-direct access data sets, VSAM and virtual I/O data sets, and concatenated data sets other than VSAM or ISAM. The installation can selectively authorize time-sharing users to allocate data sets requiring volume mounting; also under installation control, time-sharing users can direct SYSOUT data sets to remote stations.
- The installation can place frequently used time-sharing commands or service routines in the pageable link pack area (PLPA) without reducing the amount of real storage available. When time sharing is not active, the time-sharing programs in the PLPA do not require real storage.
- External page storage is used for swapping, thereby eliminating the swap data set.

Handling the Job Stream

OS/VS2 Release 2 offers several newly designed or enhanced facilities to improve the management of jobs with the resources available in the system.

Job Control Language

The job control language (JCL) specifies job information, data characteristics and device requirements for a program at execution time. The job control statements of OS/VS2 Release 2 remain basically unchanged from MVT and VS2 Release 1 for compatibility.

Job Entry Subsystems

Note: Information about JES3 is for advanced planning purposes only. JES3 will not be inlcuded in the initial version of VS2 Release 2.

In Release 2 of OS/VS2 two job entry subsystems will be available, JES2 and JES3; only one of these subsystems can be specified at system generation to run as a component of the control program. Both JES2 and JES3 perform the following functions:
- Reading jobs into the system.
- Scheduling jobs.
- Maintaining all data submitted with jobs.
- Supporting the system management facilities.
- Handling all output from batch jobs and time-sharing users.

Job Entry Subsystem 2 (JES2)

JES2 will perform functions formerly processed by HASPII, as well as many functions formerly performed by the job scheduler; these include:
- Reading job streams, including JCL and data from input devices.
- Reading input from remote work stations.
- Scheduling jobs.
- Starting initiators.
- Stopping initiators.
- Processing warmstarts.
- Printing or punching output to an end-use device.

In addition to these, several new features have been included in JES2:
- Complete integration of the checkpoint/restart facility; JES2 maintains a system job journal to support system restart, automatic step restart, and checkpoint restart.
- Full SMF support compatible with VS2 Release 1.
- A facility for adding user-written routines to drive end-use devices.
- A new output queuing facility to support all output classes, including time-sharing output.
- Immediate detection of JCL syntax errors, before processing of a job has begun.
- Dynamic allocation and deallocation of output data sets. The deallocation process is called spin-off; spun-off data sets are immediately available to time-sharing or batch users for printing.
- A facility to request printing and routing options by data set, as well as by job.

Job Entry Subsystem 3 (JES3)

JES3 is a job entry subsystem that will service the processing requirements of from one to four VS2 Release 2 uniprocessors or multiprocessors or any combination of both. JES3 supports any System/370 Model 145, 155II, 165II, 158, 168, 158MP, and 168MP, all of which must have at least one million bytes of real storage. The functions of JES3 are generallly compatible with those of ASP Version 3.

A major advantage of JES3 is improved system availability; JES3 is designed to minimize the potentially disruptive effect of failures in both hardware and software. In the event of a hardware failure, provision is made to continue operations with remaining system units. Software components are individually restartable. Therefore, the impact of control malfunctions and transient hardware problems can be localized.

When JES3 is used to manage a complex of processors only one processor controls the job scheduling and device allocation for the entire complex. The controlling processor is called the *global* system; the other processors are called *local* systems. A local system can function as a global system whenever a global failure occurs.

To dynamically assume global functions, a local processor must have access to peripheral equipment, either to its own equipment or, via a 2914 switching device, to the equipment formerly controlled by the failing global processor. The local processor must also be connected by a channel-to-channel adapter to the other local processors. The switch to the alternate global system is assisted by the operator. Normally, it can be accomplished without logically disrupting jobs executing on the local processors at the time of the global failure.

There are additional features of JES3 contributing to availability:

- Failing local processors can be restarted without disrupting other processors.
- The global system can be warmstarted, generally without disrupting work on local processors.
- Error exits are defined to allow continued system operation when failures can be confined to sub-functions.

In a complex of multiple processors, JES3 presents a single system image to a user; in other words, the complex appears to the user to be a single, large capacity processor. JES3 provides a single centralized job source for the entire complex and a single operator interface to functionally segmented operator consoles.

JES3 permits efficient use of system resources by providing:

- Automatic scheduling of work to multiple processors.
- Controlled scheduling of peripheral devices.
- Mounting and verifying of private volumes before a job is scheduled.
- Time-sharing support for all systems.
- Checkpoint/restart support for all systems.
- SMF support for all systems.
- Remote job entry facilities.
- A facility for adding user-written routines to extend subsystem capabilities.

JES3 provides several functions to manage the flow of jobs through the system. For example, table-driven algorithms provide for deadline scheduling and operator or user-controlled scheduling of jobs to remote JES3 systems. User-specified scheduling algorithms can also be used for JES3 systems.

JES3 provides new growth opportunities for users because it supports a variety of system configurations. For example, JES3 can connect up to four VS2 Release 2 systems and share the direct access and tape devices among the systems. In a JES3 complex all Release 2 processors share a set of direct access storage volumes containing SYSIN/SYSOUT data. Figure 6 illustrates a configuration supported by JES3.

JES3 can also attach ASP Version 3 main processors to a global processor. To schedule work to an ASP main processor, a user must supply a special control card designating that system.

Channel-to-channel (CTC) connections are required from all processors to the global processor. In Release 2 systems, CTC connections are used to transfer control information. Since ASP main processors do not share the SYSIN/SYSOUT volumes in these systems, CTC connections are used to exchange SYSIN/SYSOUT data as well as control information.

JES3 uses three levels of operator consoles: system consoles, device consoles, and global consoles.

System consoles are controlled by the operating system. Every processor in a complex must have its own system console. It is the responsibility of an operator at a system console to control all jobs scheduled by the global processor to his processor. A system console operator can convert his local system to a global system should a failure occur in the global processor.

Device and global consoles are controlled by global console support routines in JES3. The same physical console can be used as a device console and global console. A device console operator is advised at all times by the global system of the condition of the devices he is assigned to control. He may receive requests to mount and dismount forms, trains, carriage tapes, tape volumes, and disk volumes. The operator of the global console has control of the entire complex; he has access to the centralized job source, all local systems, and all devices. He can alter the scheduling algorithms that are in effect for the entire complex.

Scheduler Work Area

The SYS1.SYSJOBQE data set entries have been replaced in VS2 Release 2 by a pageable portion of virtual address space known as the scheduler work area (SWA). To reduce contention for a job queue, the SWA contains most of the control blocks that used to be part of the SYS1.SYSJOBQE data set. To enable recovery, the control blocks can be recorded on direct access storage. And since control blocks are not built until a job is selected for execution, pre-allocated auxiliary storage requirements are reduced.

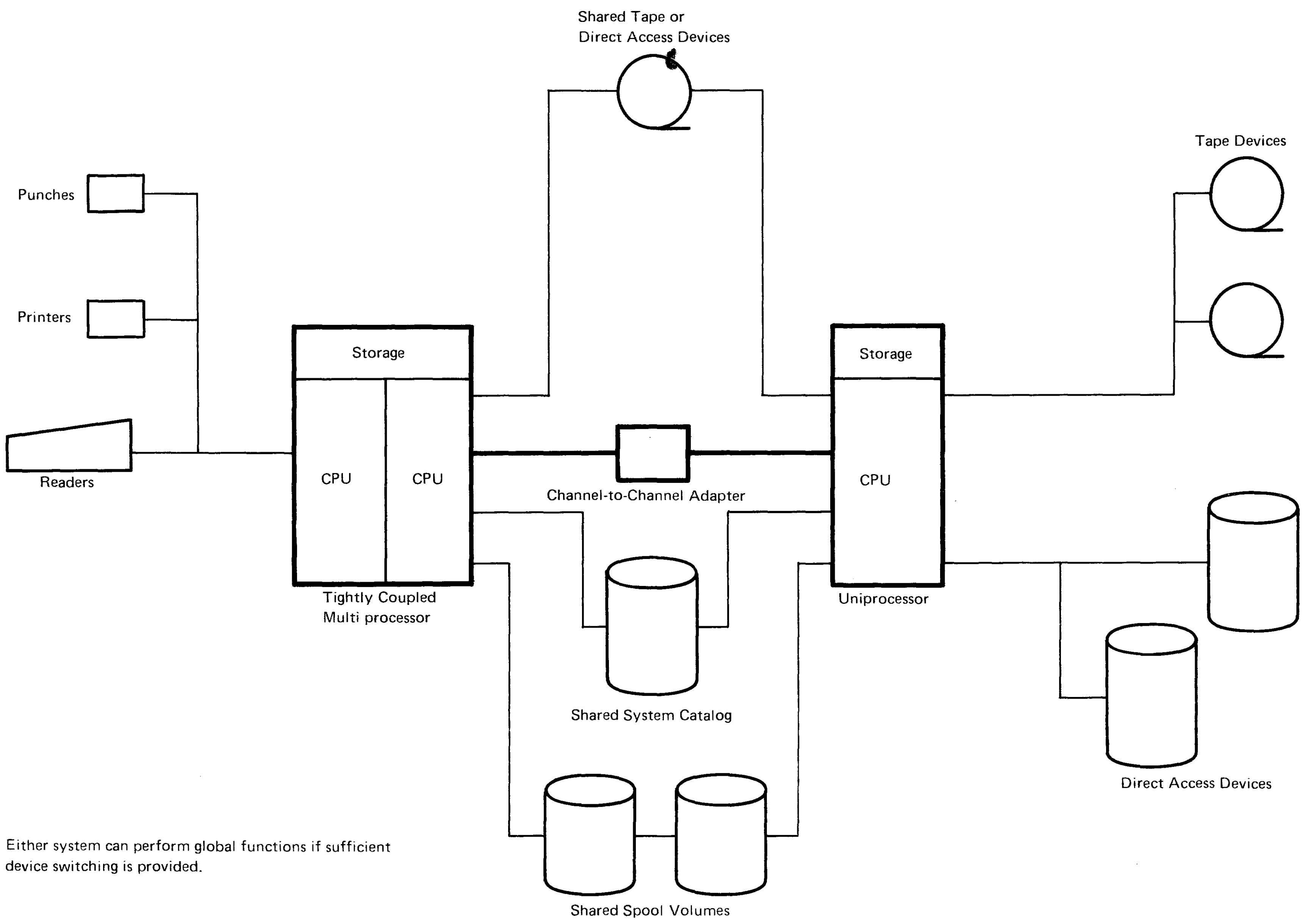

Shared Tape or Direct Access Devices
Tape Devices
Punches
Printers
Readers
Storage
CPU
CPU
Tightly Coupled Multi processor
Channel-to-Channel Adapter
Storage
CPU
Uniprocessor
Shared System Catalog
Direct Access Devices
Shared Spool Volumes
Either system can perform global functions if sufficient device switching is provided.

Figure 6. Example of a JES3 Installation

Dynamic Allocation

Dynamic allocation of auxiliary storage, which was previously available only through TSO, can now be invoked by both background and foreground jobs. In OS/VS2 Release 2, dynamic allocation provides:

- Multi-volume or multi-unit data sets.
- Generation data groups.
- Concatenation of data sets except VSAM and ISAM data sets.
- Optional freeing of data sets at CLOSE. This includes SYSOUT data sets which, when freed, will be processed by the job entry subsystems.
- Allocation of devices other than direct access devices.

Handling Data Sets

In OS/VS2 Release 2, the data management access methods are primarily responsible for moving information between virtual storage and external storage and maintaining it in external storage. The data management facilities in VS2 Release 2 have been designed to provide systematic and effective means of organizing, identifying, storing, cataloging, and retrieving of all data, including loadable programs processed by the system.

Virtual Storage Access Method (VSAM)

The virtual storage access method operates with direct access devices and supports both sequential and direct processing. VSAM creates and maintains two types of data sets.

One type of data set is organized by a key field within each record and is called a *key-sequenced* data set. Data records are located using the key field and an index that records key fields and the addresses of the associated data. Key-sequenced data records can also be located using the displacement from the beginning of the data set; the record displacement is called the relative byte address.

The other type of data set is organized according to the time of arrival of each record into the data set. This is called an *entry-sequenced* data set. These data records are located using the relative byte address.

VSAM offers users two major advantages. One is its data organization techniques, which minimize data movement and provide device-independent data sets designed for long-term stability and for data base applications. The other is a set of service routines designed to facilitate data management; these routines initially define data sets, copy and print data sets, delete VSAM entries from a catalog, and provide full data set portability between DOS/VS and OS/VS.

Conversion of data sets from ISAM or SAM format to VSAM format is another function of the service routines. An ISAM interface program is provided to map ISAM macro instructions into corresponding VSAM requests, so that most programs written for use with ISAM data sets can also be used with VSAM. In VSAM, access to data is controlled by assembler language macro instructions.

In addition to major improvements, certain device-dependent calculations have been automated in VSAM to minimize the programmer effort required to change device types. For example, the calculation of optimum block size for a device is now performed by the system. VSAM also offers multiple levels of password protection and an exit for user-written security routines to improve data set security.

In OS/VS2 Release 2, support for VSAM remains basically unchanged from previous support in VS.

Other Access Methods

In OS/VS2 Release 2, support for QSAM, BSAM, BPAM, BDAM, and ISAM remains unchanged from previous releases of OS/360 and VS.

Queued Sequential Access Method (QSAM)

In QSAM, logical records are retrieved or stored as requested. This access method anticipates the need for records based on their sequential order; normally, it will have the desired record in real storage, ready for use, before the request for retrieval. When writing data, a program using QSAM can continue as if a record had been written immediately, although the QSAM routines may have blocked that record with other logical records and deferred writing until the output buffer has been filled.

A new parallel GET routine is provided in OS/VS2 Release 2. To minimize implied waits, the new GET routine can scan several QSAM input data control blocks and select one for which a record is available.

Basic Sequential Access Method (BSAM)

In BSAM, data is sequentially organized and is stored or retrieved in physical blocks. The READ/WRITE macro instruction initiates input and output operations. The completion of these operations is tested by using synchronization macro instructions.

Basic Partitioned Access Method (BPAM)
BPAM, used in conjunction with BSAM, is designed for efficient storage and retrieval of discrete sequences of data (members) belonging to the same data set on a direct access device. The data set includes a directory that relates a member name to an address at which a sequence begins. Each member has a simple name. Members can be added to a partitioned data set as long as space is available in the directory and data set. All input/output operations for BPAM other than directory manipulation are performed by BSAM.

Basic Direct Access Method (BDAM)
In BDAM, records in a data set are organized on direct access volumes in any manner chosen by the programmer. Records are sorted or retrieved by actual or relative addresses within the data set. An address can be that of the requested record or of a starting point within the data set where a search for the record begins. BDAM is designed to support multiple users of a data control block or a data set in a multi-tasking environment.

Indexed Sequential Access Method (ISAM)
In ISAM, data records are arranged in logical sequence by a key field. Both queued and basic sequential access methods are available to process an indexed sequential data set.

Virtual Telecommunications Access Method (VTAM)

Note: Information about VTAM is for advanced planning purposes only. VTAM will not be included in the initial version of VS2 Release 2.

In OS/VS2 Release 2, telecommunications support has been extended by the addition of a new access method designed specifically to execute in a virtual storage environment. The major advantages of the Virtual Telecommunications Access Method (VTAM) are that it permits:

- Sharing of network resources with multiple application programs executing concurrently in the central computer.
- Concurrent execution of TCAM and VTAM application programs sharing the same telecommunications network.
- Operation with the IBM 3704 and 3705 Communications Controllers to reduce the number of functions performed in the central computer for remote devices.

Using the 3704/3705 communications controllers in network control mode, VTAM makes the telecommunications lines and communications controllers transparent to most application programs. VTAM is a terminal-oriented system; application programs communicate only with terminals. In this way, several application programs can communicate with different terminals on the same multipoint line, or with the same terminal although not simultaneously. Network operator-control facilities are provided that enable a user to monitor and dynamically reconfigure his network to meet fluctuating requirements.

VTAM is similar to BTAM in that it provides the same direct-control level of support but with greater functional capabilities. Queued level support is obtained through TCAM since VTAM permits sharing of its network resources with TCAM programs. In a shared TCAM/VTAM network, when a TCAM program schedules a read or write operation, the request is routed to VTAM. To the TCAM application, the request appears as if it were handled only by TCAM.

VTAM supports the IBM 3704 and 3705 Communications Controllers in network control mode only, and locally attached IBM 3270 Information Display Systems.

In OS/VS2 Release 2, VTAM, TCAM, and BTAM can coexist in the same operating system with separately defined telecommunication networks. In addition, a single application program may use both BTAM and VTAM to support separate telecommunications networks provided that all of the requirements of both access methods are met.

VTAM's design and use of tailored RAS facilities provide a reliable telecommunications system designed for long-term stability and simplified teleprocessing growth. VTAM and VSAM are companion access methods on which to build data-base/data-communications systems.

VTAM facilities include a teleprocessing online test executive program (TOLTEP). For further information on TOLTEP, refer to the section of this publication entitled "Reliability, Availability, and Serviceability."

Other Teleprocessing Access Methods

Telecommunications Access Method (TCAM)
TCAM is a general purpose teleprocessing support program. It provides unified management of local and remote terminal devices, including binary synchroneous communications (BSC) devices, through a single message control program that interfaces with input/output services (IOS). Within

an application program, a user can specify the source and destination of terminal I/O and can issue TCAM operator control commands. In OS/VS2 Release 2, TCAM does not support QTAM programs.

TCAM has been modified in OS/VS2 Release 2 to support changes made to the 3704/3705 Communications Controllers Network Control Program (NCP); however, TCAM does not support the 3704/3705 Remote Program Loader Feature. This feature will be available for TCAM through VTAM.

The TCAM message control program (MCP) serves as an interface between remote terminals and user-written application programs. The MCP insulates application programs from the complex device dependencies inherent in a teleprocessing environment; it controls the flow of messages to and from terminals and to secondary storage devices on which messages are queued until their destinations are ready to receive them. Messages with a common destination can be queued from more than one source and held until the destination terminal is no longer busy sending or receiving another message. Destination queues can be in real storage or on auxiliary storage. Auxiliary storage queues permit a high volume of concurrent terminal operations without requiring excessive amounts of real storage for buffering.

In addition, TCAM offers the following service facilities:

- A set of commands enabling a user to determine the status of his teleprocessing system and to alter, activate, or deactivate portions of the system by entering commands from the system console, remote terminals, or application programs.
- A checkpoint/restart facility to restore the message control program environment following a system failure or closedown.
- An online test facility (TOTE) to test transmission control units and remote terminals without closing down the MCP or deallocating the device being tested.
- Debugging aids, including error recovery and event recording facilities and dumping utility programs.
- A facility for selectively logging incoming or outgoing messages or message segments.

Basic Telecommunications Access Method (BTAM)

BTAM provides the tools to write a telecommunications program. It includes facilities for:

- Creating and modifying terminal lists.
- Initiating and answering calls to and from terminals or switched networks.
- Polling and addressing terminals on non-switched multi-point lines.
- Transmitting and receiving messages.
- Translating code.

In OS/VS2 Release 2, BTAM provides the same functions that it did in OS. Error recovery procedures, diagnostic error services, and online terminal tests are optional. BTAM can execute application programs through a separate network; it supports start/stop and binary synchronous communications devices, as well as all other devices supported in VS2 Release 2 except the 3704/3705 Communications Controller in Network Control Program mode.

Graphic Programming Services

The graphic programming services handle graphic input and output and include macro instructions and problem-oriented routines to be used as building blocks in the construction of graphic processing programs. Graphics services support the IBM 2250 display unit, models 1 and 3, and the IBM 2260 direct attachment.

In VS2 Release 2 GAM support remains basically unchanged from previous releases of OS/360 and VS.

Shared Direct Access Storage Devices

The shared direct access storage device option (shared DASD) allows one or more direct access devices to be shared between two or more CPUs. The devices must be connected to a control unit that has a path to each CPU. With shared DASD, systems can share and consolidate data; no changes to existing records, data sets, or volumes, are necessary. The results are improved flexibility and a reduction of DASD space requirements.

In OS/VS2 Release 2, up to four processors can share a pool of 2314/2319 direct access devices or a pool of 3330 series devices. Two CPUs can share 2305 devices.

Virtual I/O (VIO)

In Release 2 of OS/VS2 temporary data sets can be handled by a new facility called virtual I/O (VIO). Data sets for which VIO is specified reside within the paging data sets. However, to a problem program and the access method, the data sets appear to reside on some other real direct access storage devices. The physical record sizes used by the access method are transparently changed to 4K pages.

During system generation new and/or existing unit names can be defined for VIO. These unit names must be coded on the JCL statements defining the data sets to specify VIO for system-named temporary data sets.

As a data set is created, VIO allocates page-size (4K bytes) blocks in external page storage. These blocks are not necessarily contiguous, so the data set may reside on several volumes of external page storage. When a user accesses a VIO data set, channel programs are intercepted and the desired data is paged in and out of real storage as required. The blocks in external page storage are released when the data set is deleted and the space is immediately available for paging.

VIO provides these advantages:

- Elimination of some of the usual I/O device allocation and data management overhead for temporary data sets.
- Generally more efficient use of direct access storage space.
- Use of the I/O balancing capability of the paging mechanism.

VIO processing is compatible with the BDAM, BPAM, BSAM, QSAM and XDAP macro interfaces; no changes to user-written code are required. Minor code changes may be required for EXCP macro interfaces. Recovery processing currently available for temporary data sets is also provided for VIO data sets. However, VIO temporary data sets cannot be made available for deferred restart.

The VSAM Catalog

In OS/VS2 Release 2 all catalogs are VSAM catalogs capable of handling both OS and VSAM data sets. VSAM catalogs are key-sequenced VSAM data sets and can be password protected to prevent unauthorized access to them.

In Release 2, as in OS, only one catalog per system can be accessed at system initialization. Therefore, the catalog must contain entries for all the system data sets; this catalog is called the master catalog. It is no longer a requirement that the master catalog be located on the IPL volume.

The OS/VS2 Release 2 master catalog can contain pointers to other catalogs called private catalogs. These catalogs correspond to OS catalogs that do not reside on the IPL volume (CVOLs). The pointers to VS2 private catalogs correspond to CVOL pointer entries. The first level of qualification in a data set name can indicate that a catalog entry for a data set exists in a specific private catalog.

In Release 2 a user can change the normal order of catalog search by using new DD statements; he can define a catalog to be used for a single job or jobstep. The order of search is first through the step catalog or the job catalog, then the private catalog indicated by the first level qualifier in the data set name, and finally, the master catalog. This user-controlled search strategy offers increased flexibility over that of the current catalog structure.

The new multifunction access method services provide the facilities to convert an OS catalog to a VS2 Release 2 catalog. A new utility program can be used to update an existing OS catalog with changes made to a VS2 Release 2 catalog.

The changes in the Release 2 version of the catalog in most cases should not require any changes to current job control language statements. The new catalog management routines support the OS catalog management interfaces for catalog, uncatalog, recatalog, and locate-by-name functions.

Access Method Services

The access method services comprise a multi-function service program for VSAM data sets; it performs these functions:

- Defines a VSAM data set or catalog, and catalogs non-VSAM data sets.
- Defines and deletes aliases for catalog names and non-VSAM data set names.
- Lists VSAM and non-VSAM catalog entries or records of a data set.
- Prints data set records.
- Moves or copies a VSAM catalog.
- Converts an OS catalog or an OS/VS2 Release 1 catalog to a Release 2 catalog.
- Converts a sequential or an indexed sequential data set to the VSAM format.
- Converts a VSAM data set to sequential format.
- Makes a data set portable from one operating system to another.
- Copies a data set for reorganization.
- Creates a back-up data set.
- Supports generation data groups (GDGs).

Data Set Utilities

The data set utility programs reorganize, modify, or compare data at the data set and/or record level and are required for the proper generation and maintenance of the system control program. The general function performed by each utility is described in the following list.

- IEBCOPY copies, compresses, merges, loads and unloads partitioned data sets.
- IEBGENER copies a sequential data set or members of a partitioned data set, or converts a data set from sequential to partitioned organization.
- IEBPTPCH prints or punches records residing in a sequential or partitioned data set.
- IEBCOMPR compares two identically organized sequential or partitioned data sets at the logical record level.
- IEBISAM copies, prints, reorganizes, loads and unloads indexed sequential data sets.
- IEBUPDTE updates a symbolic library.
- IEBEDIT produces an edited input job stream data set from a master job stream data set.
- IEBTCRIN constructs records from input read from an IBM 2495 Tape Cartridge reader.
- IEBDG (data generator) creates output data sets with either internally generated test data or externally supplied input.

Operating the System

This section describes features that are significant to the operation of the VS2 Release 2 system.

Operator Commands

In VS2 Release 2, the operator command language supports the following enhancements:

- The job entry subsystem (JES2 or JES3).
- The system resources manager.
- Multiple private address spaces.
- The virtual telecommunications access method (VTAM).
- Multiprocessing.

For example, as in MVT, the operator can physically change the configuration of a multiprocessing system by means of the VARY CPU, VARY CHANNEL, VARY PATH, and VARY STORAGE commands.

In addition, certain commands have been modified to give the operator better control over the information he requests, thereby reducing the amount of unnecessary information and the size of the displays.

Initialization

VS2 Release 2 includes facilities aimed at increasing the speed and flexibility of system initialization. These facilities include:

- Elimination of messages for information only.
- Optional bypassing of macro instructions that result in a wait state, such as WTORs. The system programmer at an installation implements this option.
- Optional multiple members in SYS1.PARMLIB for SMF, for the link library list (LNKLST), and for defining volume attributes, as decided by the system programmer at an installation.
- Optional placement of operator commands used during IPL into SYS1.PARMLIB; the commands are internally issued at the completion of IPL. This also is implemented by the system programmer.
- An option whereby the system programmer can predefine system parameters in SYS1.PARMLIB.

Most of these changes attempt to decrease the amount of operator intervention required during initialization.

NIP will attempt to initialize the system regardless of errors in the specification of system parameters. If a system parameter is incorrectly specified, the processor for that parameter will prompt the operator for a correct specification. If the incorrect parameter came from SYS1.PARMLIB, the operator can invoke the EOB option: the system will select system defaults, if available, or else cancel the option.

Multiple Console Support

Multiple console support (MCS), a standard feature of VS2 Release 2, allows an installation to use one master console and multiple (up to thirty-one) secondary consoles. The master console is the basic console required for operator-system communication: it alone can accept all operator commands, change the status of the hardcopy log and the messages to be recorded on it, switch to a different master console, and receive all messages not specifically assigned to another console. Secondary consoles can be dedicated to one or more system functions, such as a tape library, disk library, or teleprocessing control.

MCS services all consoles concurrently, creating an environment for operator/system interaction that gives each console the appearance of being the only console on the system. Each console operator receives only those messages related to the commands entered at that console, or related to the function of that console.

MCS provides the ability to:

- Route messages to selected functional areas.
- Allow a user-written exit routine to modify the message's routing and description codes before the message is issued.

- Switch to an alternate console if a primary console should fail.
- Allow automatic message deletion on devices such as display tubes (graphics).
- Support a hardcopy log for the recording of routed messages, operator commands, and system responses.

System Log and Hardcopy Log

VS2 Release 2 provides a system log on which programmers and operators can record information and a hardcopy log that provides a permanent record of system activity.

The system log can contain the following kinds of information:

- Job time, job step time, and data from the JOB and EXEC statements of a job that has ended.
- Operating data entered by problem programs using a write-to-log (WTL) macro instruction.
- Description of unusual events that occur during processing.
- Write-to-operator (WTO) and write-to-operator-with-reply (WTOR) messages.
- Accepted replies to WTOR messages.
- Commands issued through operator's consoles and the input stream, and commands issued by the operating system.

The hardcopy log provides a permanent record of all the messages issued by or to the system. It is required for systems with an active graphic console or multiple active consoles; for other systems, the primary console device serves as the hardcopy log. The hardcopy log can be kept on another, non-graphic console device or can be included in the system log.

Since multiple console support allows the use of more than one console in a system, the hardcopy log provides a place to collect all the messages from and to all the consoles; therefore, an installation can review system activity by reviewing message activity.

Device Independent Display Operator Console Support (DIDOCS) and Status Display Support (SDS)

Device independent display operator console support (DIDOCS) and status display support (SDS) are two standard features of VS2 Release 2 that allow and enhance the use of graphic display devices as operator consoles. Both DIDOCS and SDS are automatically included in the system when an installation uses a display console.

DIDOCS provides uniform operator console support for a range of display devices, resulting in faster communication between the system and the operator than can be achieved with the standard printer keyboard or composite console devices. The operator can:

- Respond to a message or enter a command while messages are being written to the screen.
- See action messages (e.g., WTOR or MOUNT) and delete any he no longer needs.
- Use the light pen or cursor, when available, to delete messages and perform other display-oriented functions.
- Initiate automatic command entry either with the selector pen or with the program function keyboard (PFK), when available.

Status display support (SDS) is designed to provide a clear and understandable presentation of information to a system operator. The operator can obtain an in-line display or an out-of-line display within specified display screen areas and can obtain a dynamic status display by means of the new TRACK command, which replaces functions the MONITOR command formerly provided.

Automatic Volume Recognition (AVR)

Automatic volume recognition (AVR) is a standard feature of VS2 Release 2 that allows the operator to premount volumes on any available tape or disk device prior to the initiation of the job step that requires the volumes. The operator might therefore reduce the wait time lost in performing job set-ups. In release 2, the AVR facility will no longer issue generic device type volume mounting instructions.

Managing the System

As computer systems become more complex, the ability to control the users in the system and to make the best use of system resources becomes more important. VS2 Release 2 includes facilities designed to help the installation manage the system.

Use of Resources

The design of VS2 Release 2 provides the potential for more jobs, both foreground and background, to be active concurrently in the system. With more jobs competing for system resources, the installation will become increasingly concerned with resource distribution -- both with optimizing the use of resources and with assuring acceptable response or turnaround time to individual jobs.

To aid the installation in solving the problem of resources distribution, there is a need to provide facilities to:

- Predict and control the response or execution time of any particular job in relation to the system workload.
- Handle various users differently, at certain times allowing some jobs to be favored over others.
- Assure an acceptable level of performance to important jobs.

In VS2 Release 2, a new facility called the *system resources manager* attempts to provide such facilities.

The objective of the system resources manager is to keep in real storage those address spaces that both best use the system resources and meet installation-specified performance objectives, at any instant of time and under any workload in the system. To meet its objective, the system resources manager has two major functions:

- Managing the workload according to installation-specified performance objectives.
- Managing the use of system resources.

Managing the Workload

In VS2 Release 2 an installation can specify, in measurable terms, the performance that any member of any subset of its users is to receive, under any system workload conditions and during any period in the life of a job. The system resources manager is responsible for tracking and controlling the rate at which resources are provided to users in order to meet the installation's requirements.

The installation sets up an installation performance specification (IPS) in a member of SYS1.PARMLIB. (Optionally, the system can define more than one IPS, although only one can be used at a time.) In the IPS, the installation defines:

- Performance groups -- subsets of users that should be managed in distinguishable ways.
- Performance objectives -- distinct rates, called service rates, at which CPU, I/O, and real storage resources are provided to users in a performance group at a certain workload level in the system.

Service rate is the number of service units per second a user should receive; a service unit is a measure of processing resources. The system resources manager monitors the rate at which service is supplied to a user in order to ensure that the installation performance specification is met.

The average user is not concerned with the activity of the system resources manager or with the IPS. To take advantage of the system resources manager, he simply identifies the performance group in which he is to be included, as prescribed by the installation.

The following paragraphs describe in greater detail the concepts of service, performance groups, and performance objectives.

Service--The Measure of Performance: Service units are used to measure the amount of processing resources provided to each address space. They are computed as a combination of the three basic processing resources:

- CPU execution, where one unit is the execution of 10,000 instructions.
- I/O measure; *i.e.*, the SMF I/O event count unit.
- Real storage occupancy; *i.e.*, one frame occupied for some multiple of CPU execution units.

The installation can supply coefficients by which to multiply each of these factors, as illustrated by A, B, and C in the following formula:

service units = A(CPU) + B(I/O) + C(frames)

When an installation specifies performance objectives, it specifies one or more service rates, how many service units per second a user should receive. The installation is *not* specifying any particular amount of the *individual* resources that a user is to receive; it is assumed that different users will use CPU, I/O, and real storage resources in different proportions. However, by supplying coefficients to be multipled by each resource, the installation can adjust the relative importance of CPU, I/O, or real storage resources within the service definition. For example, if real storage is a critical resource, the installation can increase the value of C in the service formula. Therefore, a job that uses a great deal of real storage will accumulate more service units. Likewise, if real storage is not a critical resource, C can be assigned the value 0, and a user will not accumulate service units at all by using real storage -- the real storage factor will drop out of the service definition. Defaults will be supplied for each of the coefficients.

Once the installation has determined the service definition (*i.e.*, supplied the coefficients or accepted the defaults), the installation should normally be interested only in the number of service units and the service rates each user receives, not in particular amounts of CPU, I/O, or real storage used. System management facilities (SMF) will record service unit measurements. From SMF reports, the installation can interpret service unit measurements in terms of response or turnaround time.

Performance Groups: The purpose of performance groups is to group user transactions that the installation considers to have similar performance requirements. Basically, a user transaction in a batch environment is a job or job step; in a time-sharing environment, an interaction. The installation can define as many as 255 performance groups, each identified by a distinct performance group number.

Each performance group can further be divided into as many as eight periods. By dividing a performance group into periods, an installation can associate different performance objectives with different periods in the life of a transaction. The duration of a period can be specified either as a number of real-time seconds or as a number of accumulated service units. For example, one performance group might be defined for short compile-load-go jobs that should have a very rapid turnaround time. The installation divides the performance group into two periods. The first period is associated with a high service rate and lasts until N number of service units are accumulated, where N has previously been determined to be sufficient service units to complete a short compile-load-go job. If a job in the group is not complete after N service units are accumulated, it enters the second period, which can specify a lower service rate for the duration of the job.

Performance Objectives: A performance objective states service rates, how many service units per second an associated transaction should receive under different system workload conditions. The installation can define as many as 64 performance objectives, each identified by a distinct number from 1 to 64.

The motive for specifying different performance objectives is to give certain users better service at the expense of other users. However, the relative importance of this motive is dependent on the size of the system workload, *i.e.*, the degree to which the demand for resources exceeds the supply. If the system workload is very light (*i.e.*, the demand for resources does not exceed the supply), all users can be assured satisfactory service rates. However, if the system workload is heavy, the installation will want to assure an acceptable service rate to high priority users and lower the service rate for low priority users. Under very heavy workload conditions, the installation might completely sacrifice the service rate of low priority users (*i.e.*, assign a service rate of 0) to continue to provide acceptable service to the high priority users. By defining *workload levels*, the installation can express the varying service relationships between groups of users at different system workload levels.

The installation can define as many as 32 workload levels, identified by integers from 1 to 128. The numbers an installation chooses to identify the workload levels it defines are arbitrary. A higher workload level number, however, must always indicate a higher system workload and all workload levels defined must have a corresponding service rate included in each performance objective.

For example, an installation defines four performance objectives, numbered 3, 6, 9, and 12. When the system workload is light, each performance objective should assure a satisfactory service rate to the users associated with it: the installation assigns the number 10 to this workload level and assigns corresponding service rates to each performance objective, as illustrated in Figure 7. Under very heavy workload conditions, the installation wants to completely sacrifice the service rates for objectives 3 and 6, lower the service rate for objective 12, but still assure acceptable service to users associated with objective 9. This workload level is assigned the number 40; the corresponding service rates associated with each objective are illustrated in Figure 7. The two intermediate workload levels shown in Figure 7 are established to reflect the changing relationships between the performance objectives as the workload level increases from 10 to 40.

The system resources manager tracks the service rates provided to users and the average workload level of the system. As the level increases or decreases, it adjusts the service rates to maintain the relationships between the performance objectives (and therefore between the users associated with the objectives) that the installation has defined.

Associating a Transaction with a Performance Objective: User transactions are associated with performance objectives by means of performance groups and periods within each performance group: each period of a performance group definition includes a performance objective number.

For example, performance group 2 includes time-sharing student users; it is divided into two periods. The first period lasts until 200 service units are accumulated and is associated with performance objective 3 in Figure 8. At workload level 10, the user transaction will receive 40 service

Performance Objective Number	Service Rate for Workload Level 10	Service Rate for Workload Level 20	Service Rate for Workload Level 30	Service Rate for Workload Level 40
3	40	20	10	0
6	30	15	0	0
9	50	45	40	30
12	70	50	35	15

Figure 7. Performance Objectives

units per second; this rate drops, however, as the system workload level increases. At the heaviest workload level, the user transaction will not receive service. When the user transaction enters the second period, it is associated with performance objective 6. Figure 8 illustrates how the user transaction, the performance group definition, and performance objectives are related.

Managing the Use of System Resources
To manage system resources, the system resources manager serves as a centralized decision-maker. It monitors a wide range of data about the condition of the system, seeking to control such key variables as:

- Amount of real storage allocated.
- Distribution of I/O load.
- Swapping frequency.
- Level of multiprogramming.
- Paging rate.

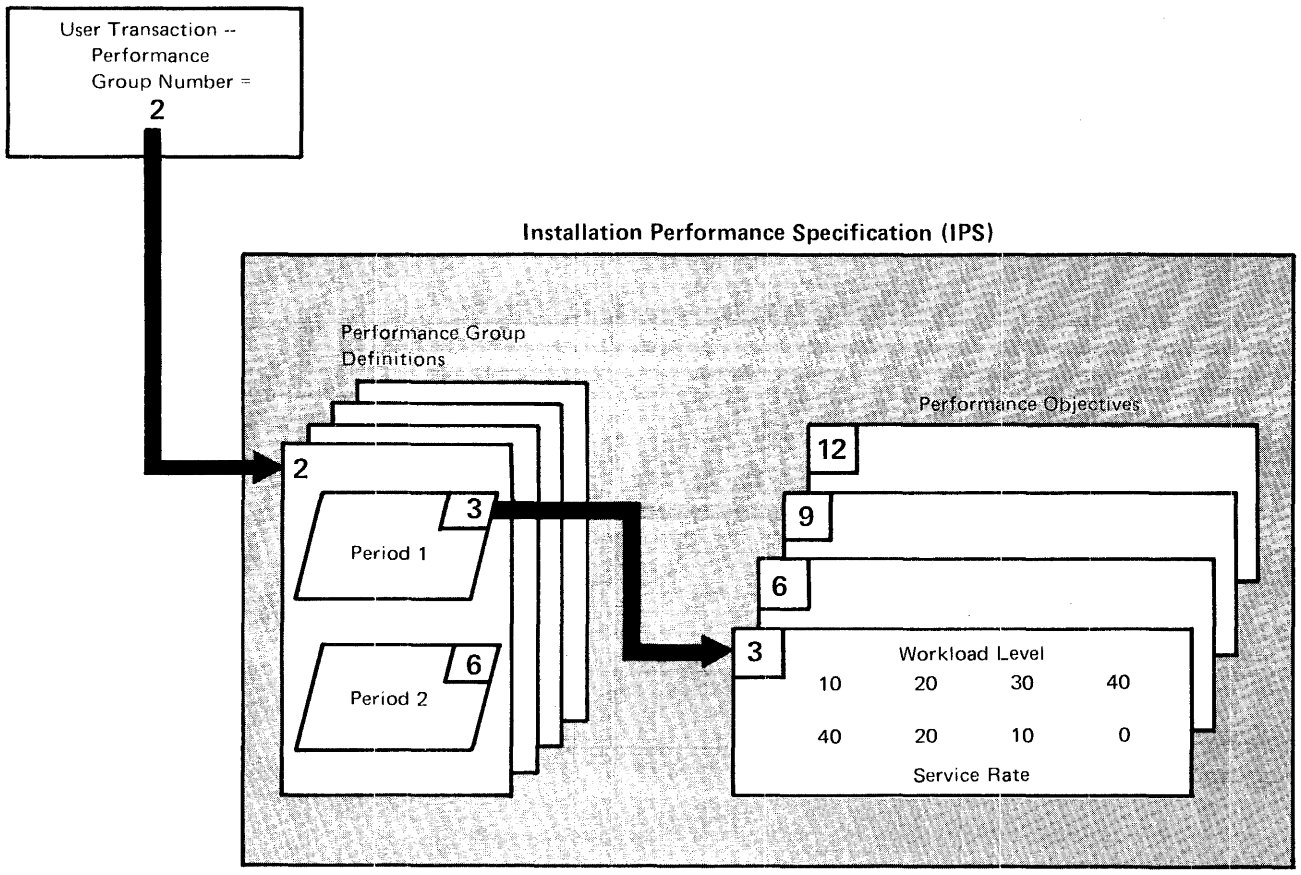

User Transaction -- Performance Group Number = 2
Installation Performance Specification (IPS)
Performance Group Definitions
2
Period 1
3
Period 2
6
Performance Objectives
12
9
6
3
Workload Level
10 20 30 40
40 20 10 0
Service Rate

Figure 8. Associating a User With a Performance Objective

Algorithms to control many of these variables exist in current systems, but they are not centralized. By centralizing the control of these variables, the system resources manager can better make decisions that will affect overall system performance.

Fulfilling the Objective
Figure 9 repeats the objective of the system resources manager, showing how each part of that objective is tied to the installation performance specification and to the functions of the system resources manager. Details on how the system resources manager fulfills its objective are included in the "Design Concepts" section under "System Resources Management."

Measurement Facilities

VS2 Release 2 includes two standard measurement facilities:

- System management facilities (SMF), which are an option in MVT and standard in VS2 Release 1, provide the means to gather and record information that can be used for user accounting or for evaluating system use. SMF is basically job and terminal-session related.
- The system activity measurement facility (MF/1), which is new in VS2 Release 2, collects information about system activity and produces trace records and reports. MF/1 is basically system-oriented; it is designed to aid the installation in improving system performance, analyzing system trends, and evaluating future system requirements.

System Management Facilities (SMF)
SMF data collection routines gather several types of information:

- Accounting information, such as CPU time and device and storage use.
- Data set activity information, such as EXCP count by device.
- Volume information, such as the space available on direct access volumes and error statistics for tape volumes.
- System use information, such as multiprocessor and I/O configurations.

To use the information collected, SMF provides exits to user-written routines that can monitor the operation of a job or job step and generate the installation's own SMF records. The exit routines can cancel jobs, write records to the SMF data set, open and close user-defined data sets, suppress the writing of certain SMF records, and enforce installation standards (such as identification of users). Dummy routines are automatically provided for all unused exits.

In VS2 Release 2, SMF provides two additional exits over those provided in MVT: a job purge exit that can be taken when a job is ready to be purged from the system; and an exit that can receive control before a record is written into the SMF data set.

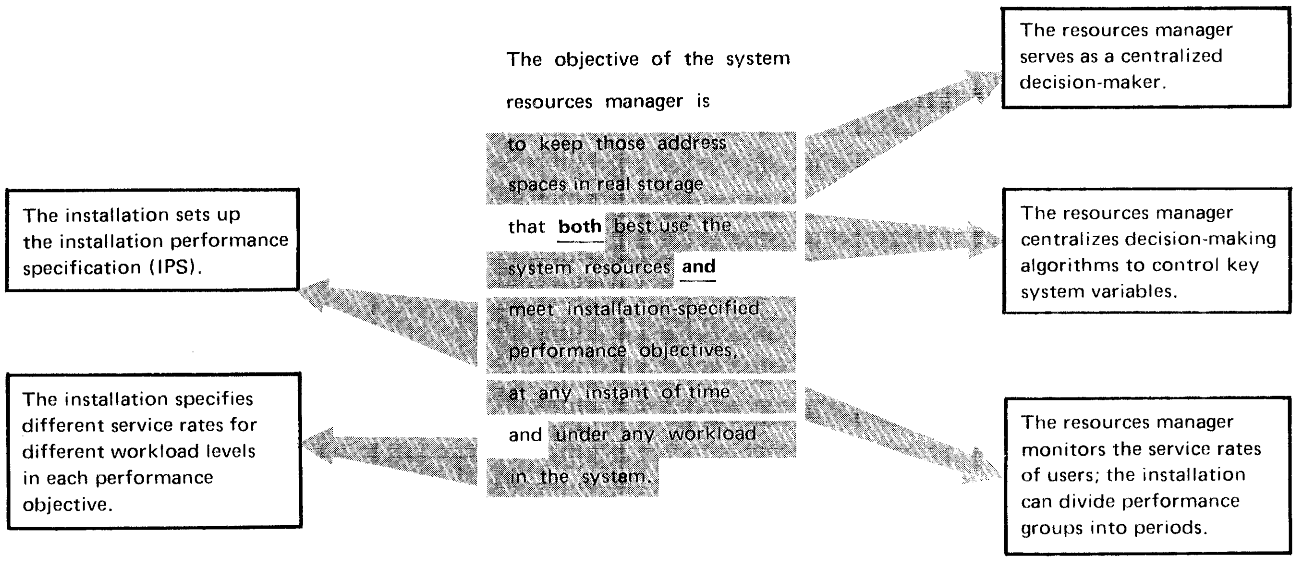

The objective of the system resources manager is
to keep those address spaces in real storage
that both best use the system resources and
meet installation-specified performance objectives,
at any instant of time
and under any workload in the system.
The resources manager serves as a centralized decision-maker.
The resources manager centralizes decision-making algorithms to control key system variables.
The resources manager monitors the service rates of users; the installation can divide performance groups into periods.
The installation sets up the installation performance specification (IPS).
The installation specifies different service rates for different workload levels in each performance objective.

Figure 9. The Objective of the System Resources Manager

SMF records are also used by the utility that maintains an OS-format catalog.

System Activity Measurement Facility (MF/1)
MF/1 is divided into three functions:

- Controlling the collection, tracing, and reporting of system activity measurements.
- Obtaining measurements of system activity.
- Producing formatted reports from the information gathered.

These functions are controlled by keyword parameters in the operator commands START and STOP. Measurements can be gathered independently for CPU activity, channel and channel-CPU overlap activity, I/O device activity and contention, paging activity, and workload activity. This allows the installation to choose which measurement gathering routines should be active in the system; although MF/1 is a standard feature, the execution of any or all of its routines is completely optional.

System Integrity

System integrity refers to the ability of a system to protect itself against unauthorized user access to the extent that security controls cannot be compromised. That is, system integrity prevents unauthorized problem programs using any system interface from:

- Bypassing store/fetch protection -- *e.g.*, reading or writing in other users' areas.
- Bypassing password checking -- *i.e.*, accessing password-protected data for which the correct password is not supplied.
- Obtaining control in an authorized state.

VS2 Release 2 is designed to eliminate system integrity exposures. IBM will accept as valid any APAR that describes an unauthorized program's use of any system interface (defined or undefined) to bypass store or fetch protection, bypass password checking, or obtain control in an authorized state. An authorized program in VS2 Release 2 is one that uses a system key (keys 0-7), executes in the supervisor state, or is authorized via the authorized program facility (APF).

The authorized program facility (APF) is a feature of VS2 that allows the installation to identify system or user programs that are permitted to use restricted functions. In Release 2 of VS2, APF is expanded to:

- Recognize the additional system protection keys 0 through 7 as an indication of APF program authorization.
- Allow an installation to specify certain libraries (with corresponding volume information) in addition to SYS1.LINKLIB, SYS1.SVCLIB, and SYS1.LPALIB, which can contain APF authorized programs.

Store/fetch protection, a standard feature of OS/VS2 supported CPUs, prevents unauthorized users from gaining access to virtual or real storage not assigned to them. In VS2 Release 2, the prevention of unauthorized access of address spaces is provided through the combination of store/fetch protection and the isolation of each user in a separate private address space.

Additional facilities and techniques used in VS2 Release 2 to improve system integrity include the following:

- Only authorized programs are allowed to use those control program services (certain SVCs and paths through SVCs) that bypass normal controls over access to resources.
- Control program data which indicates the allocation and use of system resources is maintained in areas of storage which are store protected from the user.
- The control program verifies the address and use of protected control blocks when a user or user control block supplies the address.
- The control program maintains and utilizes sufficient data on any given resource to distinguish it from other similar resources. The unauthorized user is prevented from doing such things as substituting user code or data for system code or data.
- Control program routines in system key (0-7) validate user storage areas to ensure that a given store/fetch operation will not cause a store/fetch violation.

Although VS2 Release 2 is designed for system integrity, the installation must follow certain guidelines to ensure that system integrity is maintained. These guidelines will be provided at a later date.

System Utilities

System utility programs are used to maintain collections of data and system control information. These programs, which are invoked through the use of job control statements and utility control statements, execute in the pageable area of real storage. The following are the system utility programs:

IEHDASDR: The IEHDASDR program initializes direct access volumes for use with the

operating system and dumps data from or restores data to these volumes.
IEHINITT: The IEHINITT program writes volume label sets in EBCDIC, in BCD, or in ASCII code on magnetic tapes.
IEHATLAS: The IEHATLAS program locates and assigns an alternate track to replace a defective track and copies usable records from the defective track to the alternate track.
IFHSTATR: The IFHSTATR program selects, formats, and writes information from Type 21 ESV (error statistics by volume) records.
IEHLIST: For Release 2, the LIST catalog function is not supported. It has been replaced by access method services. The LISTPDS and LISTVTOC functions are continued.
IEHMOVE: For Release 2, some IEHMOVE functions are replaced by access method services or no longer supported. The MOVE/COPY functions for non-VSAM data sets and volumes are continued, but without DSGROUP support. The PDS merge functions are also continued.
IEHPROGM: For Release 2, some IEHPROGM catalog functions are replaced by access method services or no longer supported. Scratch, rename, and password functions are continued.

Independent Utilities

Independent utility programs are used to prepare direct access devices for system use and to ensure that any permanent hardware errors incurred on a direct access device do not seriously degrade the performance of that device. The user controls the operation of independent utility programs through utility control statements. Independent utility programs do not operate with the VS2 control program, but they support OS/VS2 with the following services:

- The IBCDASDI program initializes direct access volumes for use with OS/VS2.
- The ICAPRTBL program performs stand-alone buffer loading for the IBM 3211 printer.
- The IBCDMPRS program dumps or restores data between a direct access storage device and a removable volume.

These independent utilities are not supported for the 3066 System Console. The *OS/VS Utilities* manual describes the use of IBCDASDI and ICAPRTBL with the 165II and the 168 models of System/370.

Reliability, Availability, and Serviceability

With Release 2 of OS/VS2 IBM has provided improved reliability and advanced availability, and serviceability (RAS) facilities that can be used to promote continuous system operation. The purpose of RAS facilities is to reduce the frequency and impact of system interruptions that are caused by programming of hardware failures. RAS components, standard and optional in VS2 Release 2 are summarized in the following paragraphs.

Online Test Executive Program (OLTEP)

The online test executive program (OLTEP) directs the selection, loading and execution of online test sections (OLT) within the Release 2 environment. While other jobs continue running, OLTEP, via the OLTs, tests the input/output hardware components of a system; it can:

- Check I/O hardware.
- Exercise a device requiring dynamic adjustments.
- Verify I/O hardware repairs and engineering changes.
- Diagnose I/O errors.
- Transfer diagnostic test results to a RETAIN/370 center.
- Allow a RETAIN/370 center to modify diagnostic test request and options.

OLTEP and the OLTs normally execute in a minimum of 76K bytes of non-paged real (V=R) storage. To execute the logout analysis OLT which analyzes machine checks, OLTEP can operate in the pageable (V=V) portion of real storage. The online test sections (OLTs) are not distributed with the VS2 Release 2 software package.

Teleprocessing Online Test Executive Program (TOLTEP)

Note: Information about TOLTEP is for advanced planning purposes only. TOLTEP will not be included in the initial version of VS2 Release 2.

The teleprocessing online test executive program (TOLTEP) controls the selection, loading and execution of teleprocessing online terminal tests (OLTTs) for all control units and terminals in a VTAM network. Using VTAM capabilities for line sharing and remote reporting and test requests TOLTEP performs control services, device accessing, and configuration updates. It allows OLTs to run concurrently with the operating system, VTAM, and other processing programs. TOLTEP is automatically included in a system when VTAM is generated; however, the OLTTs are not distributed with the VS2 Release 2 software package. It is initiated and stopped with VTAM as well.

Although TOLTEP provides testing facilities for the VTAM network, TOTE and OLTEP are still used for appropriate non-VTAM networks. Since TOLTEP does not support dedicated testing of a locally attached 3704/3805 Communications Controller, that function is processed by OLTEP.

Dynamic Support System

The dynamic support system (DSS) is a debugging program that assists the IBM Field Engineering Program Systems Representative and user authorized maintenance personnel to identify and correct causes of programming failures. DSS uses the program event recording (PER) hardware feature of the System/370.

DSS has its own input/output capability. It has access to most of virtual storage and to each private address space and the CPU-related data of each processor in an MP environment. When running, DSS has control of the system. However, it allows the system control program to continue processing while it monitors execution. DSS regains control via its own monitoring functions.

Since DSS takes control of the system on activation, time dependencies cannot be maintained. Therefore, DSS should not be invoked while a time-dependent program is running. Any modification made to the system while DSS is activated will not be carried over to the next cold-start IPL. DSS cannot be used to modify itself, IPL, or NIP.

Checkpoint/Restart

The checkpoint/restart facilities gather and record enough information about the status of a job and its related control blocks to allow a restart, should one be necessary. Execution can resume at the beginning of a job step (step restart) or from a place within a job step (checkpoint restart). Automatic restart can be authorized by the operator at the console. Deferred restarts take place when a programmer resubmits a job.

Included in Release 2 checkpoint/restart are the facilities to support the job entry subsystems, extended allocation, and VIO. In addition, checkpoint/restart now re-verifies passwords for protected data sets at restart time. Installations can also establish procedures to prevent access to checkpoint data sets. This will prevent modification of any sensitive system data contained in checkpoint data sets.

Missing Interruption Checker

The missing interruption checker (MIC) in VS2 Release 2 notifies the operator if a device and/or channel end interruption is not received within a specified period of time. When an interruption has not been received, it is possible that a mount message has not been satisfied or that a device has malfunctioned. Specific actions of an operator depend on the conditions he encounters. He may be required to ready a device on which a volume has been mounted, examine indicator lights on the device for abnormal signs, or terminate the job.

Machine Check Handler

The machine check handler (MCH) gathers information about all machine checks which is recorded on the SYS1.LOGREC data set. It determines if recovery from a malfunction was successfully completed by System/370 hardware facilities. If recovery attempts were unsuccessful, MCH performs a limited analysis, and then invokes appropriate software routines.

In the multiprocessing environonement of Release 2, if MCH is unsuccessful in a failing CPU, it will attempt to initiate further recovery processing by marking the failing CPU offline and invoking alternate CPU recovery (ACR) in the non-failing CPU.

Channel Check Handler

The channel check handler (CCH) receives control after the detection of a channel data check, channel control check, or interface control check. CCH builds an error control block for later use by the error recovery procedure and constructs a record of the error environment which is recorded in the SYS1.LOGREC data set.

When CCH is entered due to an error affecting an entire channel, it will invoke the I/O restart function to attempt to recover active I/O on the failing channel.

Software Error Recording

In Release 2 of OS/VS2, records of software failure and recovery activity will be written in the SYS1.LOGREC data set, via a new software recording facility. The recorded data can then be retrieved, formatted, and printed in the same way that hardware data records have been handled in OS and in previous releases of VS. Software error records will aid the customer engineer and the user in isolating system failure and evaluating the effectiveness of program changes.

Alternate CPU Recovery

Alternate CPU recovery (ACR) is a new feature in OS/VS2 Release 2. When a CPU in a shared real storage multiprocessing environment can no longer function, a signal is emitted before the CPU enters a permanent wait or stopped state. The signal can indicate a hardware malfunction alert or a software emergency signal; in either case, ACR is invoked.

When ACR receives control, it attempts to transfer work which was in progress on the failing CPU to the non-failing CPU. It cleans up work that cannot be transferred by processing the failure as an abnormal termination. ACR resets I/O on channels connected to the failing CPU and, for symmetrically connected I/O, attempts to restart it through channels to the working CPU. In the case of symmetrically connected I/O (two channel switch through channels on two CPUs), ACR invokes the recovery termination management routines to recover the program in progress at the time of the failure.

Alternate Path Retry

When one channel path develops an error the alternate path retry (APR) routines attempt to process the I/O on an alternate channel path. The alternate path must previously have been assigned to the device performing the I/O operation. An operator can vary a path to a device online or offline using the VARY PATH command. He can vary offline all paths except those to shared direct access devices with an outstanding RESERVE request and the last path to an allocated device.

Dynamic Device Reconfiguration

When dynamic device reconfiguration (DDR) receives a request from the operating system or the operator, it allows a demountable volume to be moved from one device to another. The system initiates a reconfiguration request to bypass permanent I/O errors; DDR is processed without necessitating a re-IPL or even abnormally terminating the affected job. DDR is available for paging volumes as well as normal auxiliary storage volumes.

Service Aids

The service aids are programs that facilitate the diagnosis of system or application program failures. The service aids described here are standard features of VS systems.

Generalized Trace Facility

The generalized trace facility (GTF) is a standard feature in VS2. It assists users in performing problem determination and diagnosis by tracing system events, user events, or both. GTF consists of two major functions: the generalized trace function and the trace edit function.

The generalized trace function can be started from the master console. The user has the option of tracing internally in the GTF private address space or externally to a data set on an auxiliary device. In OS/VS2 Release 2, the following improvements have been made to the generalized trace function:

- Internal trace mode now includes event selectivity, user event tracing via the GTRACE macro instruction, and comprehensive event tracing.
- Trace event records contain more information.
- The user can control the number of trace events to be edited by ABDUMP/SNAP or by the EDIT function when trace events from a SYS1.DUMP data set are being edited.

The trace edit function operates as a feature of the AMDPRDMP service aid and as an extension to ABDUMP/SNAP processing. It provides a user with selective data reduction capability for the trace data set and for AMDSADMP dumps, the SYS1.DUMP data set, and ABDUMP/SNAP dumps. When operating as a feature of the AMDPRDMP service aid, EDIT runs as a problem program and can be invoked by JCL. In OS/VS2 Release 2, these new functions have been added to trace edit processing:

- Selective editing of trace event records from AMDSADMP dumps and from the SYS1.DUMP data set.
- Editing of trace event records, including user records, during ABDUMP/SNAP processing when GTF is tracing in external trace mode.
- Selective editing of events associated with one or more specified address spaces.
- A format appendage interface that permits ABDUMP/SNAP and EDIT to use a common format appendage.

AMAPTFLE
The AMAPTFLE service aid program is used to apply program temporary fixes to the operating system; it has two functions. The application function generates control statements and dynamically invokes the linkage editor to apply PTFs in one operation. The generate function produces the required JCL and control statements. The user must execute these in a later, separate step.

AMBLIST
AMBLIST is a program which produces formatted listings for diagnostic purposes. Depending on what options are specified on AMBLIST control statements, the following types of listings can be produced:

- Formatted object module listings.
- Formatted load module listings.
- Load module map and cross-reference listings.
- Load module map and cross-reference listings showing addresses relocated relative to user-supplied addresses.
- Load module summary data including the entry point address, the authorized program facility (APF) access code, module attributes, and the contents of a module's system status index.
- Listings of the data stored in the CSECT identification records (IDRs) of load modules.
- Program modifications to a load module library.
- A map of the system link pack area.
- A map and cross-reference listings of the system nucleus.

AMASPZAP
AMASPZAP is a service aid that enables authorized users to:

- Inspect and modify data, including instructions, in any load module that exists as a member of a partitioned data set.
- Inspect and modify data in a specific record that exists in a direct access data set.
- Dump an entire data set, a specific member of a partitioned data set, or any portion of a data set residing on a direct access device.

AMDSADMP
The AMDSADMP service aid enables a user to generate a stand-alone dump program specifically tailored to his needs. AMDSADMP can generate two types of dumps, one high-speed, the other low-speed. The high-speed version writes the control registers for each CPU, the contents of real storage, and portions of paged-out virtual address space onto a tape volume. The low-speed version writes the control registers and the contents of real storage to a tape volume or to a printer. In OS/VS2 Release 2, the high-speed version of AMDSADMP unconditionally dumps virtual storage.

AMDPRDMP
The AMDPRDMP service aid program formats and prints data sets containing real or virtual storage dumps produced by AMDSADMP or other system programs. A new AMDPRDMP control statement has been added to OS/VS2 Release 2; with the new SUMMARY statement, a user can obtain a synopsis of the storage ranges contained in a dump data set and a summary of the system status as shown on the dump data set. An interface has been provided in AMDPRDMP for user-written formatting programs.

IFCDIP00
The IFCDIP00 program is used to:

- Initialize the SYS1.LOGREC data set during system generation.
- Reinitialize the SYS1.LOGREC data set if an error has resulted in the destruction of its header record.
- Reallocate the SYS1.LOGREC data set to increase or decrease the space allocation.

IFCDIP00 is controlled by JCL only; it requires no utility control statements.

IFCEREP0
The IFCEREP0 service aid is used to:

- Select and format environment records from the SYS1.LOGREC data set and write them to an output device.
- Accumulate SYS1.LOGREC records on a history data set.
- Write accumulated records to an output device.
- Summarize information contained in the SYS1.LOGREC or history data sets.
- Accumulate, edit, summarize, and write records on different machine models.
- Gather system initialization and system termination records and place them on the SYS1.LOGREC data set.

Other Support Programs
VS2 Release 2 includes three processing programs -- the OS/VS2 system assembler, linkage editor, and loader.

OS/VS2 System Assembler

The OS/VS2 system assembler allows users to write programs in assembler language for System/370 CPUs with virtual storage; it is the only language translator that is a standard component of VS2.

The OS/VS2 system assembler is a macro assembler that enlarges the language that was available under the OS/360 Assembler F; for example, the OS/VS2 system assembler provides improved messages to diagnose user errors and support for relocate and multiprocessing instructions. The language supported by the OS/VS2 system assembler is compatible with that supported by Assemblers E and F. In addition, it is a subset of, and compatible with, the language supported by the program product Assembler H.

The OS/VS2 system assembler can be used to program any type of application. All services provided by the VS2 system control program are available when using the system assembler.

Linkage Editor

The linkage editor combines separately compiled or assembled object modules into a single program that is ready to be loaded and executed. It also combines previously edited load modules with each other or with object modules, and enables changes to be made in a program without recompiling (or reassembling) the complete program; only those sections that are changed need to be recompiled. The VS2 linkage editor provides new control statements to take advantage of virtual storage and the authorized program facility, in addition to those functions provided by the MFT and MVT linkage editor.

Loader

The loader combines the basic editing functions of the linkage editor and the loading function of program fetch in one job step. It loads object modules produced by language translators and load modules produced by the linkage editor for execution. It is designed for high performance loading of modules that do not require the specific facilities of the linkage editor and program fetch. The loader does not produce load modules for program libraries.

Design Concepts

OS/VS2 Release 2 extends the virtual storage capability of Release 1 to give each user its own address space; at the same time, Release 2 completely integrates TSO and multiprocessing into its design. This section highlights those features of the control program design that are different from the design of the MVT or Release 1 control programs.

Virtual Storage Concept

In a system with *virtual storage*, the address space available to programs is limited by the addressing scheme of the computing system, not by the amount of processor (real) storage available in the system configuration. For example, System/370 uses a 24-bit binary address scheme, so an address space as large as 16,777,216 bytes can be supported.

With this design, a virtual storage system can support an address space larger than the actual amount of real storage available. To accomplish this, the system control program stores the contents of virtual storage -- instructions and data -- on direct access storage; it brings the instructions and data into real storage (from direct access storage) only when they are required by an executing program. Likewise, the system returns altered instructions and data to direct access storage when the real storage they occupy is needed and they are no longer being used. Thus, at any time, real storage contains only a portion of the contents of virtual storage.

Virtual Storage In Release 2

Virtual storage is divided into address space for the control program and address space for user programs. In Release 1, each user job (and some system components) is allocated a region in the user address space in virtual storage (see Figure 10). In Release 2, each user job (and some system components) receives its own private user address space. That is, Release 2 supports *multiple user address spaces* (see Figure 10).

The control program creates private address spaces for the following users and system components:

- Each batch job scheduled by an initiator.
- Each logged-on time-sharing job.
- The master scheduler.
- The job entry subsystem.
- The virtual telecommunications access method (VTAM).
- The auxiliary storage manager.
- Every program started via a START command.

Virtual Storage Layout

Although each user job is given its own private address space, it does not have control over all of it -- each address space is divided into the system area, the user area, and the common area (see Figure 11).

The system area contains the nucleus, which is fixed in real storage then mapped in the low addresses of virtual storage. The common area is the high addresses of virtual storage. The common area is composed of the system queue area (SQA), the pageable link pack area (PLPA), and the common service area (CSA). The SQA contains tables and queues relating to the entire system. The PLPA contains SVC routines, access methods and other read-only system programs, and any reentrant read-only user programs selected by the installation that can be shared among users of the system. The CSA contains data for communication among the private user address spaces.

The user's private address space begins immediately after the nucleus and extends up to the common area. Thus, all users have the same amount of private address space.

Figure 11 shows the structure of the user address space in virtual storage. Space is assigned to user programs from the low address up. Space is assigned from the high address down for the local system queue area (LSQA), the scheduler work area (SWA), and subpools 229 and 230. As in Release 1, the LSQA contains tables and queues associated with the user's job and address space. New in Release 2, the SWA contains control blocks and tables created during JCL interpretation and used by the initiator during job step scheduling. Thus, each initiator has its own scheduler work area within the user's private address space, eliminating the system job queue that exists in MVT and Release 1. Subpools 229 and 230 are used for user control blocks (for example, DEBs), but can be obtained only by authorized programs. The remainder of the private address space is available to its user, with space being allocated from the low address up.

Some user programs must remain in real storage during execution. These programs are assigned the

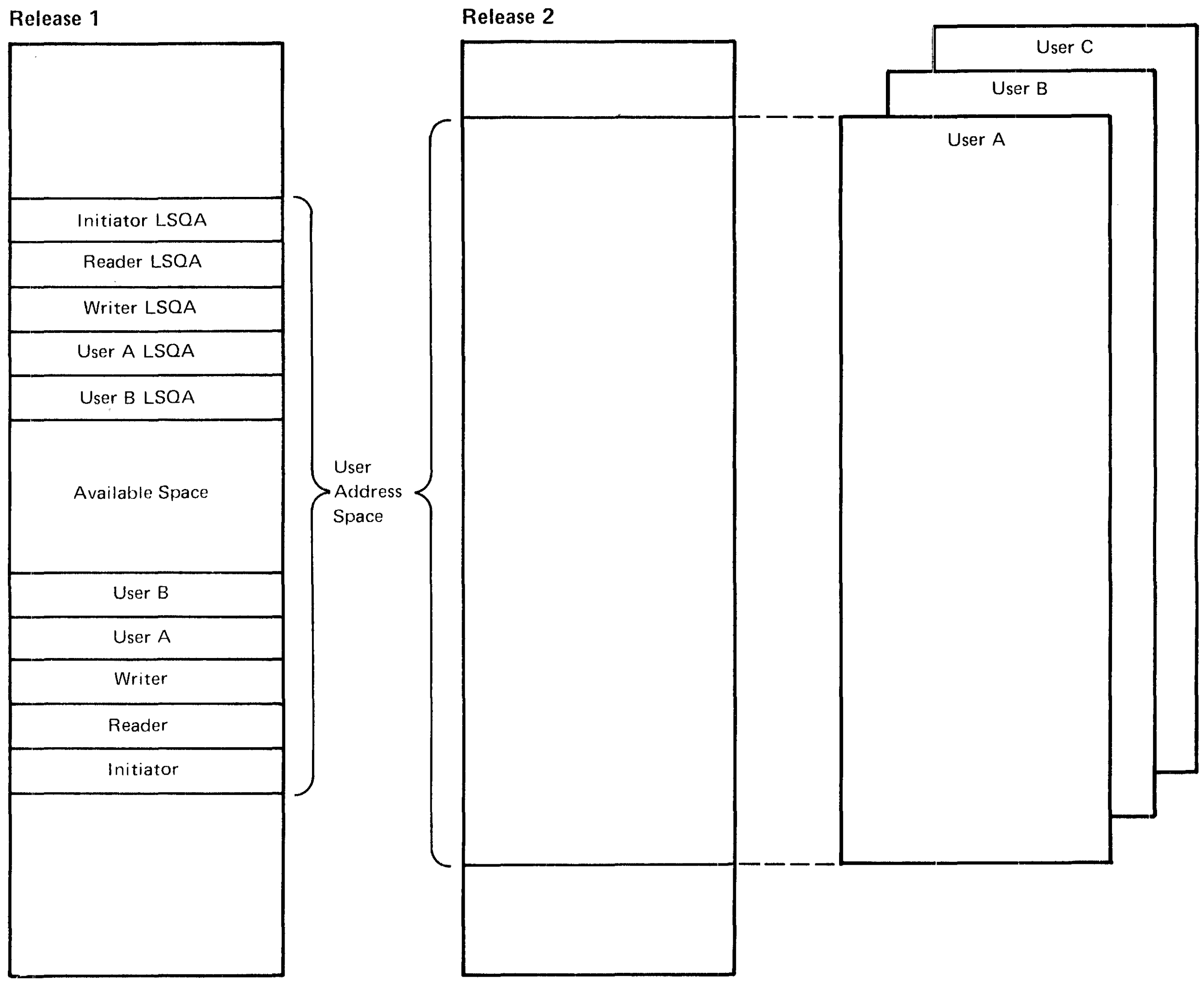


Figure 10. Virtual Storage Overviews

low addresses in the user's private address space. At execution time, they are loaded into real storage at the same address; that is, the programs have identical virtual and real storage addresses.

Storage Organization

For ease in storage management, virtual storage, real storage, and direct access storage used to contain virtual storage contents are divided into contiguous, fixed-length sections of equal size.

Virtual storage is divided into 64K-byte *segments*. (A maximum virtual storage of 16,777,216 bytes, therefore, contains 256 segments.) Each segment of virtual storage is divided into 4K-byte virtual storage *pages*; thus, each segment contains 16 pages.

Real storage is divided into 4K-byte *page frames*. Hence, a page frame is a block of real storage that can contain one page at a time.

The direct access storage that is used to contain virtual storage contents is called *external page storage*. External page storage is divided into physical records called *slots*. A slot is 4K-bytes long; therefore, a slot can contain one page at a time.

In summary, a page of data or instructions is assigned a virtual storage address. The page occupies a slot when it is in external page storage and a frame when it is in real storage.

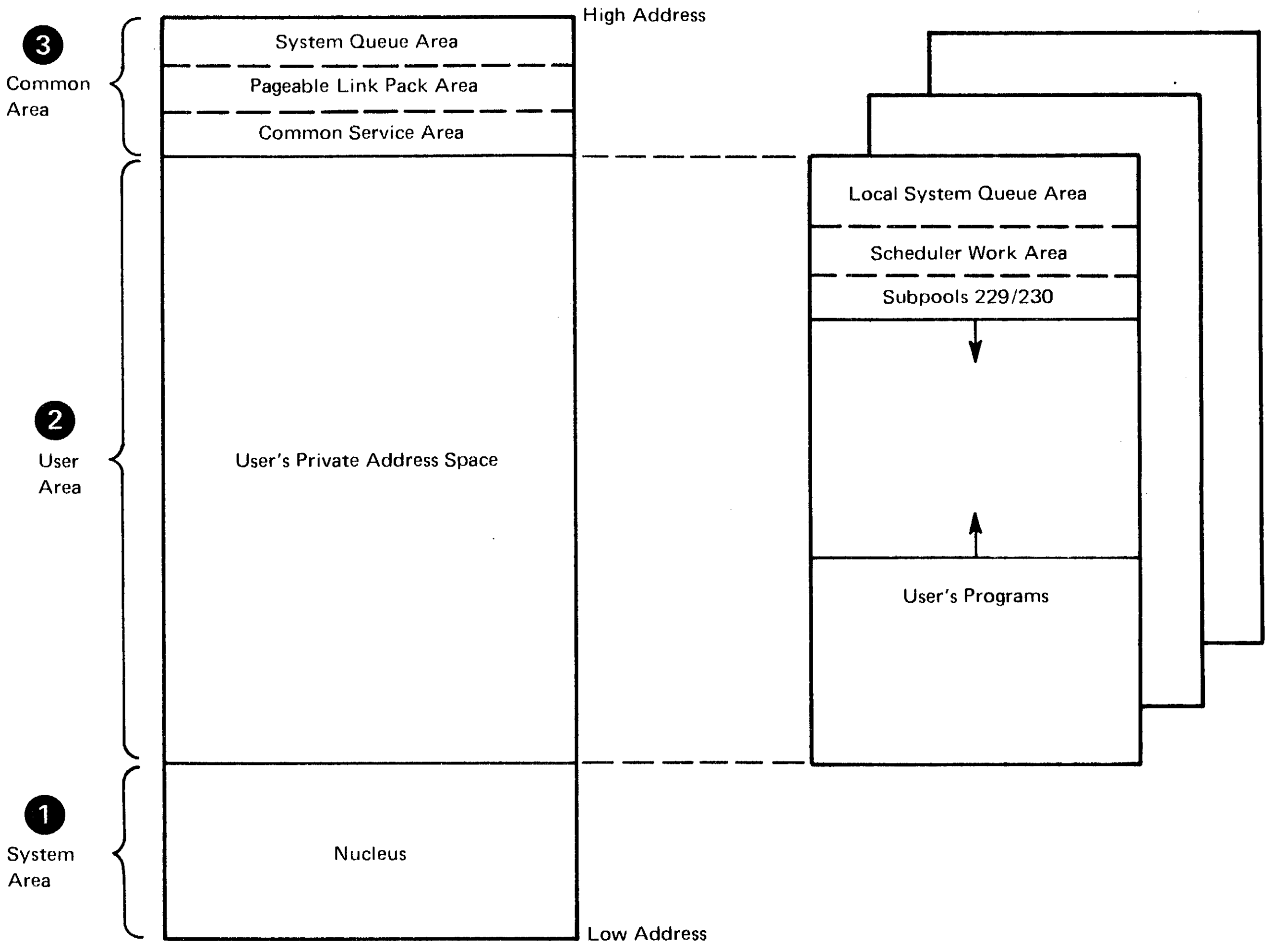


Figure 11. Virtual Storage Layout

Address Translation

When coding a program, the programmer refers to data and instructions by names or labels without knowing their physical addresses. In a virtual storage system, the control program assigns each name or label a virtual storage address that can be used to locate the data or instruction. By comparison, real storage addresses are actual physical locations in processor storage where data and instructions can be placed for processing by the CPU.

A mechanism is required to associate the virtual storage addresses of data and instructions with their actual locations in real storage. This changing of a virtual storage address to its real storage address is *address translation*. In System/370, the *dynamic address translation (DAT)* hardware feature in the CPU performs address translation.

To translate the addresses, DAT uses tables in real storage. These tables, which are maintained by the control program, are the segment table and page tables. One segment table and a corresponding set of page tables exist for each address space in the system (see Figure 12).

A segment table contains one entry for each segment in the address space that the table describes. A segment table entry defines the number of pages allocated in the segment and points to the real storage location of the page table for the segment.

There is one page table for each segment in the address space. A page table contains one entry for each page in the associated segment. It indicates which pages are currently in real storage and the real storage locations of those pages. As pages are transferred between real and external page storage, the control program changes the corresponding page table entries.

DAT translates the virtual storage addresses contained in an instruction during execution of the

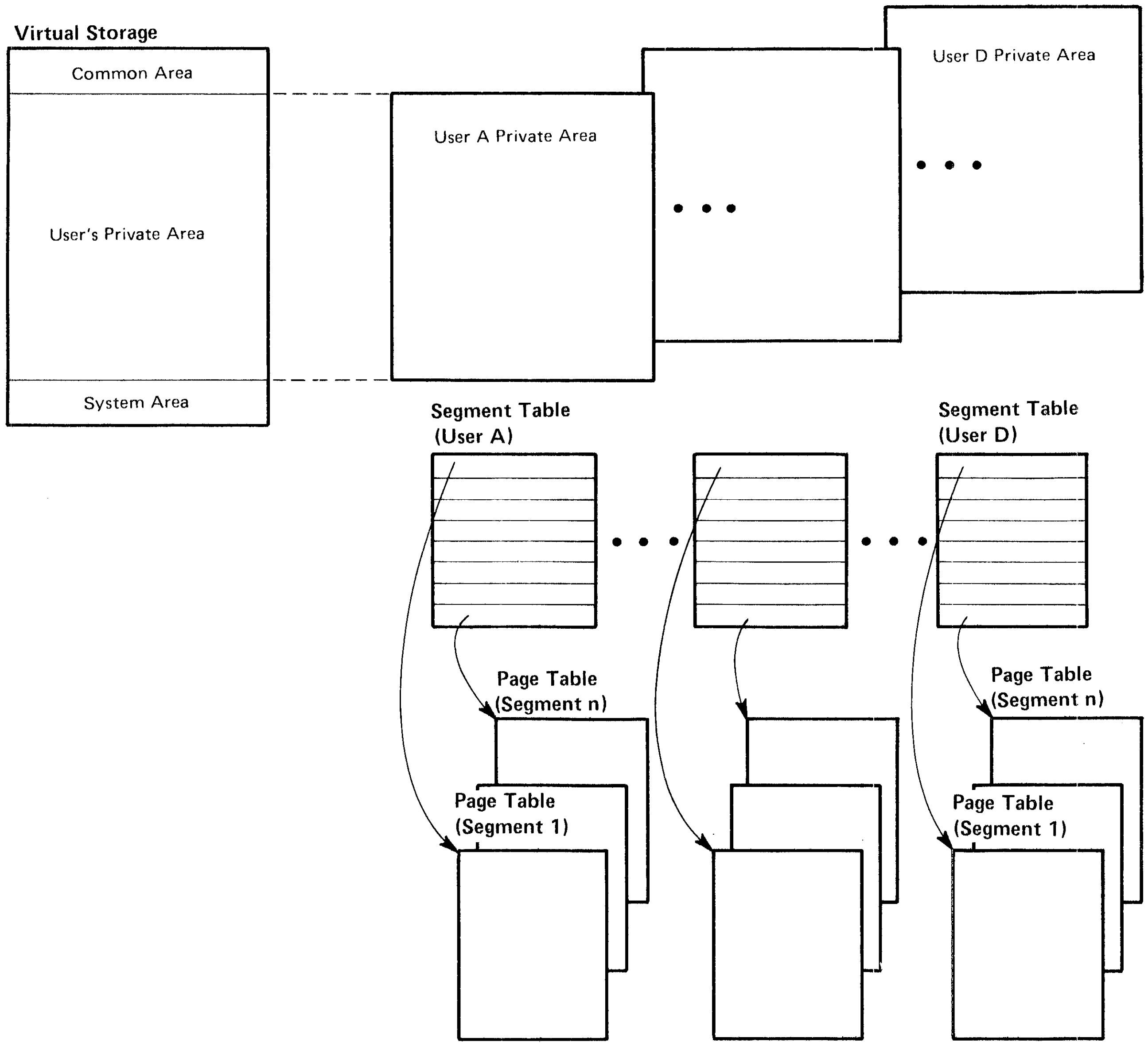


Figure 12. Segment and Page Tables

instruction. Translation occurs after the 24-bit effective virtual storage address has been computed, as usual, by adding base, displacement, and any index values together. (The format of the effective virtual storage address is included in Figure 13.)

The translation process is shown in Figure 13. First, DAT obtains the address of the appropriate segment table from a system control register. To this segment table address, DAT adds the segment address bits (from the effective virtual storage address) to obtain the segment table entry. Next, DAT obtains the page table address from the segment table entry and adds the page address bits to it in order to obtain the page table entry. Finally, DAT forms the 24-bit real storage address by appending the displacement (from the effective virtual storage address) to the page frame address in the page table entry.

To reduce the amount of time required for address translation, DAT retains up to 128 previously translated addresses in a translation lookaside buffer (TLB). Prior to performing a translation using segment and page tables, DAT searches the TLB for the required translated address.

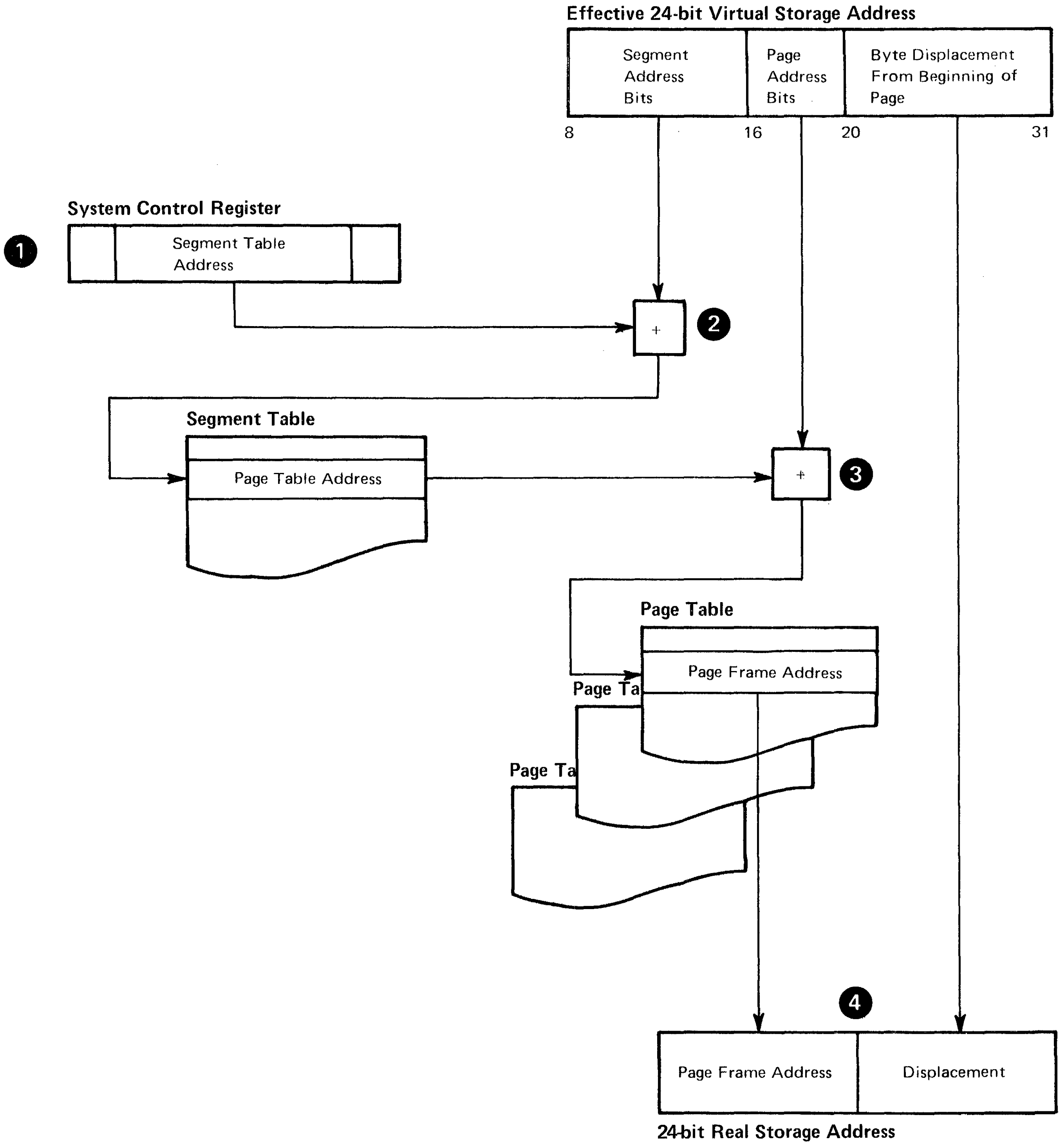


Figure 13. Dynamic Address Translation Procedure

Paging

A program interruption occurs during address translation if DAT attempts to translate a virtual storage address to a real storage address and the required page is not in real storage. This interruption, called a *page translation exception* or *page fault*, alerts the control program that the page must be loaded from external page storage into a page frame of real storage.

Data and instructions are transferred between external page storage and real storage as needed on a page basis. This transfer process is called *demand paging*, and the part of the control program that is responsible for paging is *storage management*.

The transfer of a page into real storage is a *page-in*. The page-in process is shown in Figure 14. First, when a needed page is not in real storage (indicated by a bit in the page table entry), storage management automatically goes to the corresponding entry in an *external page table*. (One external page table corresponds to each page table in the system.) The external page table entry gives the slot location for the page.

Next, storage management selects a frame in real storage to hold the required page. To do so, it refers to the *page frame table*, which indicates which frames are allocated. Storage management finds an available frame and brings in the required page from its slot in external page storage. To complete the page-in process, storage management updates the appropriate page frame table entry and page table entry.

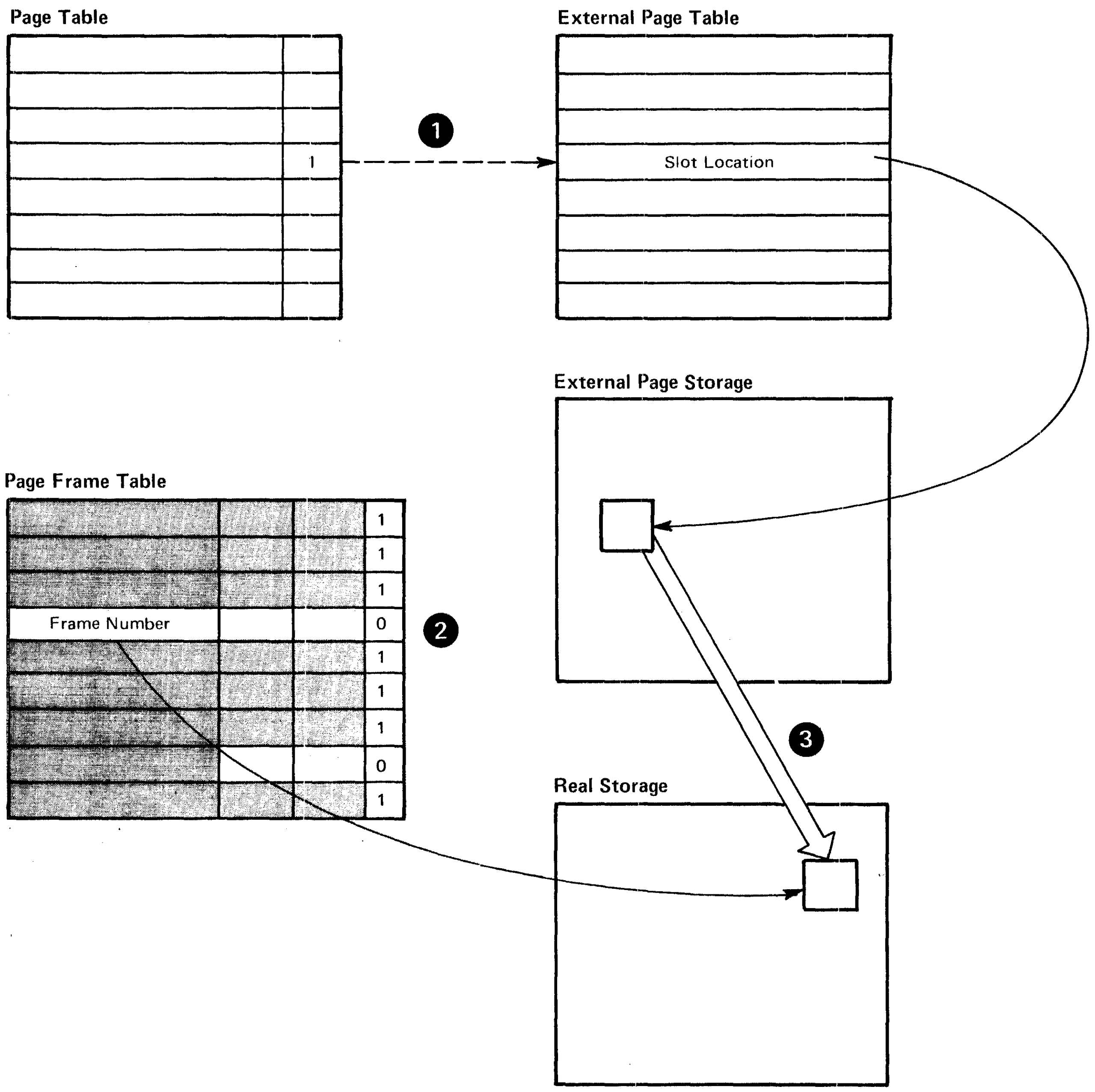


Figure 14. Page-in Process

In order to keep a supply of frames available for page-in, the control program removes pages from real storage that have not been recently referenced. Prior to removing a page from a frame, the control program determines whether the page contents were modified during processing. If so, storage management performs a *page-out*; otherwise, an exact copy of the page already exists in external page storage. A page-out copies the modified page from its real storage frame to a slot in external page storage; the slot need not be the one that contains the old version of the page. Storage management need only update the external page table entry to designate the new slot.

For various reasons, certain pages should not be paged out of real storage. For example, pages that contain I/O buffers must remain in real storage while the buffers are being referenced during an I/O operation. A page that cannot be paged out is called a *fixed* page.

Pages that are fixed for the duration of a job or job step are *long-term fixed*. For example, pages that contain certain control blocks related to a job are long-term fixed for the duration of the job. Pages that must be fixed for only a portion of the time they are in real storage are *short-term fixed*. For example, a page containing an I/O buffer is fixed prior to the start of the I/O operation; after the I/O operation completes, the page is unfixed and is eligible for page-out. Only the control program can selectively fix pages; however, it can selectively fix user pages. For instance, the control program can short-term fix user I/O area pages.

The user can fix pages, but not selectively, by labeling a job step *virtual equals real (V = R)*. All the programs in the job step are allocated a contiguous real storage area of the same size with addresses identical to their allocated virtual storage area. That is, their virtual and real storage addresses are the same. Since these programs are not paged, they do not occupy external page storage. The entire program is loaded into real storage when it is initiated, and all its pages are fixed.

Tightly Coupled Multiprocessing

In Release 2, tightly coupled multiprocessing consists of two central processing units (CPUs) of the same model (either Model 158 or Model 168) sharing one real storage and controlled by one control program. The CPUs also share I/O devices; that is, an I/O device connected to one CPU can be used by the other CPU.

The following paragraphs describe CPU functions that are characteristic of a tightly coupled multiprocessing system.

Prefixed Storage Area (PSA): Every CPU has a 4K-byte prefixed storage area (PSA) in real storage, which contains status information such as the PSW. In a uniprocessing system, the PSA begins at location 0. Since each CPU in a multiprocessing system functions independently, each must have its own separate PSA; that is, both CPUs cannot use the one PSA at location 0. Therefore, each CPU in a multiprocessing system has a separate PSA in real storage (not necessarily at location 0) pointed to by a prefix register. Each CPU still attempts to address its PSA by referencing an address from 0 to 4095; however, the hardware translates the address so that the CPU gets the corresponding location in its PSA.

Interprocessor Communication: In order to coordinate system activities, such as the reading of data from an I/O device connected to only one CPU, the two CPUs must communicate with one another. This *interprocessor communication* is accomplished through the use of the signal processor (SIGP) instruction. Using a SIGP, one CPU may request the status of the other CPU or it may request the other CPU to execute a program. For example, if one CPU has no channel available, it may request the other CPU to initiate a SIO to a particular device. A SIGP instruction appears as an external interruption to the receiving CPU.

Clock Synchronization: Both CPUs contain a time-of-day (TOD) clock. In order to provide common timing for the system, one CPU's TOD oscillator provides the pulses for both CPUs' timers.

Locking

A conflict could arise if one CPU could use a serially reusable resource before the other CPU finished with it. In MVT and VS2/Release 1, interruption of the use of these resources is prevented by disabling the CPU. In a multiprocessing system, disabling a CPU which is using a serially reusable resource does not prevent the other CPU from trying to use the resource. For that purpose, a *locking* procedure is used.

In MVT/M65MP, a designated storage location -- the lock byte -- is set by a CPU before it uses any serially reusable resource. The lock is released by the CPU only when execution involving the resource is completed. Generally, the only action a CPU can take when it finds the lock is already set (it contains the ID of the other CPU), is to continue testing the lock until it is released. Thus, locking influences system performance, since the locked-out CPU just continues testing the lock and performs no productive work.

To enhance performance, Release 2 assigns a separate lock to each of thirteen kinds of serially reusable resources; that is, Release 2 supports *multiple locks*. In this way, a lock held by a CPU on one resource does not prevent the other CPU from using a different resource.

Two *classes* of locks exist. A *global lock* protects a serially reusable resource that relates to the entire system (for example, a unit control block). A *local lock* protects a serially reusable resource that pertains to only one user address space. When a CPU holds a local lock, the queues and control blocks associated with the user address space can be manipulated only by the CPU holding the lock; however, the other CPU may process other address spaces.

The *type* of a lock is determined by what happens when a CPU makes an unconditional request for a lock that is held by the other CPU. If the requesting CPU enters a loop that keeps testing the lock until the other CPU releases it, the lock is a *spin lock*. If the request is queued and the requesting CPU goes on to do other work, the lock is a *suspend lock*.

Only privileged programs may hold a lock. In order to obtain and release a lock, the program issues a SETLOCK macro instruction. The SETLOCK macro replaces the use of the SSM instruction and some functions of the Release 1 MODESET macro instruction. Locking is not intended to replace the use of the ENQ/DEQ macro instructions.

To prevent a deadlock between the CPUs, the locks are arranged in a hierarchy, and a CPU may unconditionally request only locks higher in the hierarchy than locks that it currently holds.

Sequence of Operation

Figure 15 shows an overview of the Release 2 control program operation. This section describes the following control program components, which are new in this release or have changed from MVT and Release 1:

- System initialization.
- START/LOGON/MOUNT Processing.
- Task Management.
- Recovery Management.
- System Resources Management.
- I/O Supervisor.
- Virtual Input/Output (VIO).

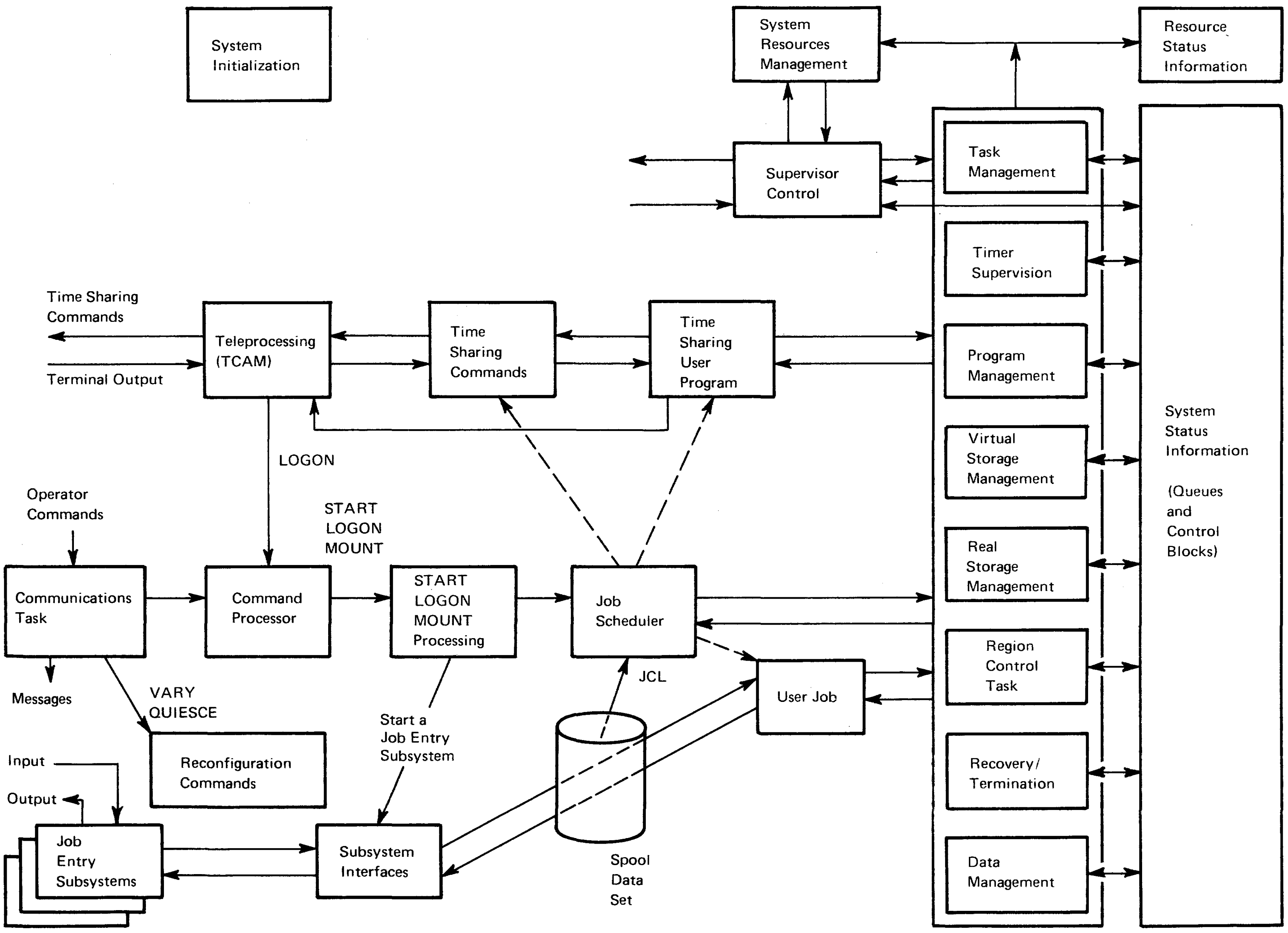


Figure 15. VS2 Release 2 Control Program Overview

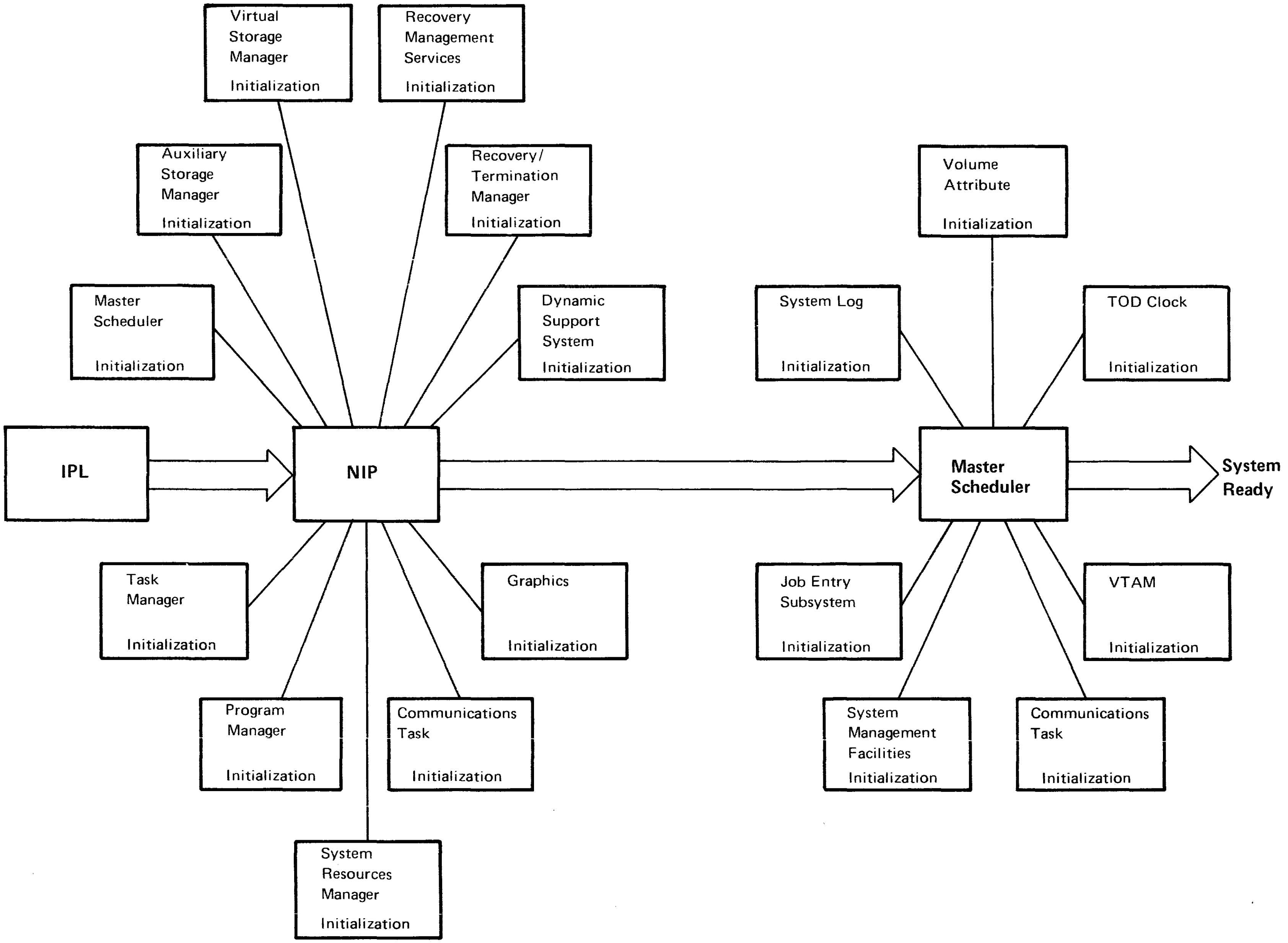

IPL
NIP
Master Scheduler
System Ready
Virtual Storage Manager Initialization
Recovery Management Services Initialization
Auxiliary Storage Manager Initialization
Recovery/ Termination Manager Initialization
Master Scheduler Initialization
Dynamic Support System Initialization
Task Manager Initialization
Graphics Initialization
Program Manager Initialization
Communications Task Initialization
System Resources Manager Initialization
Volume Attribute Initialization
System Log Initialization
TOD Clock Initialization
Job Entry Subsystem Initialization
VTAM Initialization
System Management Facilities Initialization
Communications Task Initialization

Figure 16. System Initialization Programs

System Initialization

System initialization is the process whereby the control program and its environment are prepared to handle work. The process begins when the system operator pushes the CPU LOAD key[1] and ends with the operating system ready for work.

The initialization process has three parts (shown in Figure 16): initial program loading (IPL), nucleus initialization (NIP), and master scheduler initialization. The initialization process is shown in Figure 17 and described in the following paragraphs.

IPL: When the LOAD key[1] is pushed, an IPL routine is loaded from the system residence volume into real storage starting at location 0. This program searches the IPL volume for the nucleus. When it finds the nucleus, it relocates itself so that the nucleus can be loaded at location 0. Then it passes control to the nucleus initialization program (NIP).

[1]For Model 158, the operator uses a light pen and cathode ray tube to initiate the LOAD process.

NIP: NIP initializes the control program components and the operating environment by passing control to other initialization programs. In Release 2, the initialization programs invoked by NIP are part of the system components that they initialize. (In MVT and Release 1, all the initialization programs invoked by NIP are part of the NIP component.) NIP processing includes the setting up of a virtual storage that contains a fixed area and a common area. The common area is loaded with the system queue area (SQA) and the pageable link pack area (PLPA). NIP processing also loads the master scheduler and the communications task (including an LSQA) into the private user area of virtual storage. When NIP is completed, it passes control to the master scheduler initialization program.

Master Scheduler Initialization: Master scheduler initialization routines initialize the system timer, system log, communications task, and system

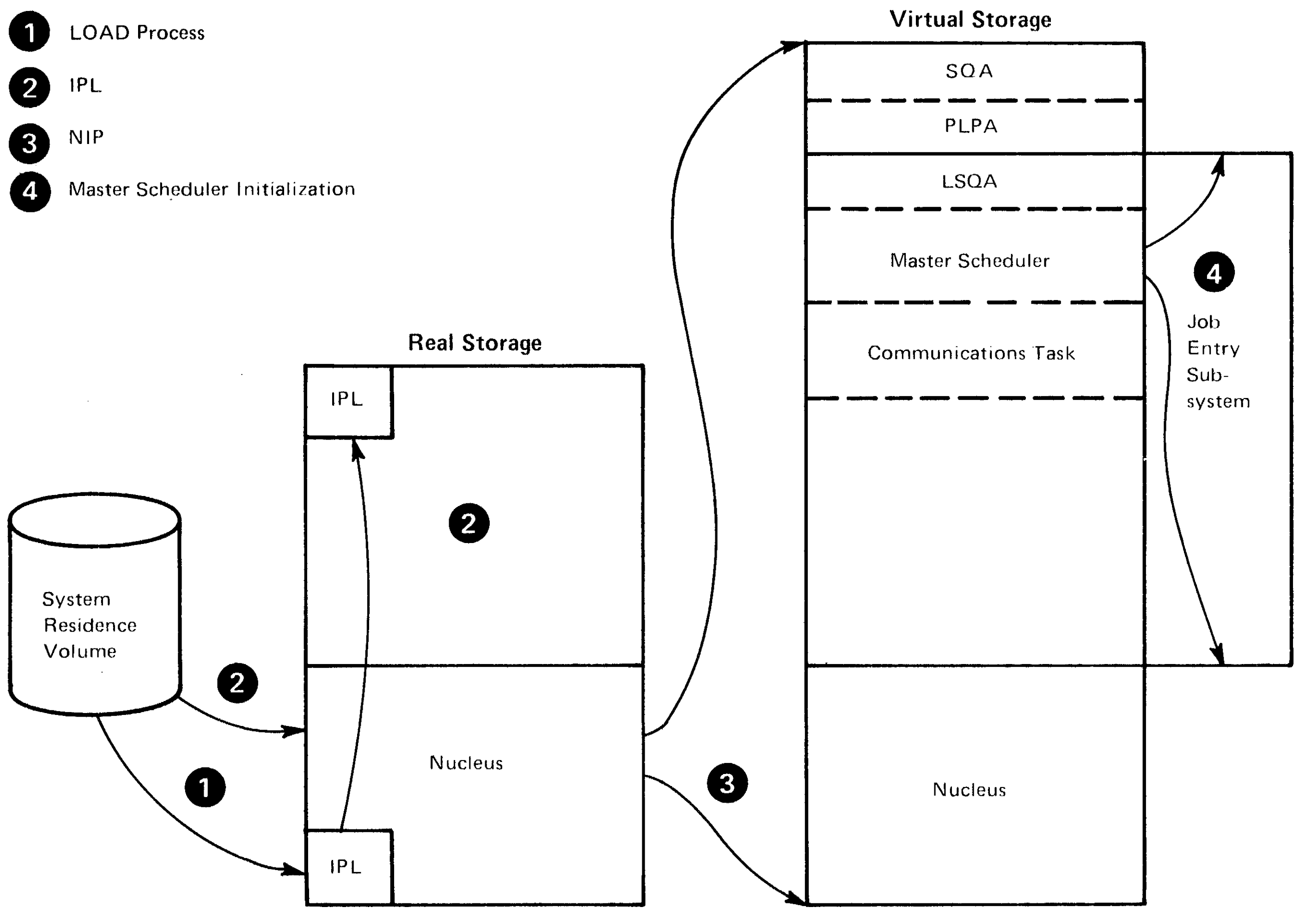


Figure 17. System Initialization Process

management facilities (SMF). They also cause creation of a private user address space for the job entry subsystem, then start the job entry subsystem.

START/LOGON/MOUNT Processing

After the system is initialized and the job entry subsystem is active, jobs may be submitted for processing. To schedule a *batch job*, the job entry subsystem issues a START command for an initiator. To schedule a *timesharing job*, a user issues a LOGON command. (Prior to LOGON, the operator must have started TCAM with the time-sharing parameter.) As a result of START and LOGON (and MOUNT) a new address space is required.

The *address space creation routine* notifies the system resources manager that a new address space is to be created. The *system resources manager* decides, based on factors like priorities and the number of already existing address spaces, whether or not a new address space is advantageous. If not, the new space will not be created until the system resources manager finds conditions suitable. If the new address space is acceptable, the address space creation routine invokes *virtual storage management (VSM)* to assign virtual storage and set up addressability for the address space.

VSM builds an LSQA and sets up a segment table, page tables, and external page tables in it. VSM also creates control blocks to operate the control task for the address space (the *region control task*).

Then the region control task (RCT) receives control. One region control task exists for each address space. When the address space is created, the RCT is the only task associated with it. The RCT builds control blocks that further define the address space, then attaches the *started task control (STC) routine*.

STC builds in-storage JCL for the task associated with the command being processed. It passes the JCL to the *job entry subsystem*. The job entry subsystem reads the job, scans and spools the JCL, invokes the *converter* to transform the spooled JCL into internal text, queues the job in an internal queue, and assigns a job ID which it returns to STC.

Next, STC uses an *initiator* as a subroutine to select the job. The initiator passes the job ID to the job entry subsystem. The job entry subsystem invokes the *interpreter* to build scheduler control blocks in the scheduler work area (SWA) for the address space. Upon return from the job entry subsystem, the initiator performs allocation and issues an ATTACH for the task related to the address space: the terminal monitor program (LOGON), the MOUNT command processor (MOUNT), or any started program (START).

Batch Job Scheduling

If the START command is for an initiator, the initiator asks the job entry subsystem for a problem program job that is ready for execution. The job entry subsystem calls the interpreter to build the scheduler tables in SWA. When the initiator receives control again, the problem program is attached.

Task Management

Task management is performed by the *supervisor*. Basically, the supervisor controls the use of the CPU(s), real storage, and virtual storage.

All supervisory activity begins with an interruption. An interruption can occur either because a program has requested a control program service or because an event has occurred. The supervisor interruption handler saves critical information (such as register contents and PSW information) necessary to return control to the interrupted program after the interruption is processed. In most cases, the interruption handler passes control to one of the following routines to process the interruption.

- *Task supervisor*, which performs services (such as attaching and detaching a subtask) requested by tasks and allocates CPU time among competing tasks.
- *Contents supervisor*, which locates requested programs, fetches them to virtual storage if necessary, and schedules their execution.
- *Timer supervisor*, which supplies current date and time of day to programs and performs interval timing.
- *Real storage manager*, which directs the movement of pages between real storage and external page storage.
- *Auxiliary storage manager*, which handles external page storage including virtual I/O data sets.
- *Actual block processor*, which initiates the actual paging I/O necessary to transfer pages in and out of real storage.
- *Virtual storage manager*, which services GETMAIN and FREEMAIN by allocating and deallocating storage within the virtual address space.

- *Service manager*, a new function in Release 2, which improves system response through a new dispatching technique that allows internal system functions to run enabled, unserialized, and in parallel on a multiprocessing system.

Recovery Management

Recovery is designed to reduce the number of errors that result in system reinitialization.
Recovery is controlled by a new component, the *recovery/termination manager*. The recovery portion of the recovery/termination manager replaces the ABEND/STAE interface and portions of ABEND and ABTERM as they exist in Release 1. When an error occurs, the recovery/termination manager selects the appropriate recovery or termination process according to the status of the system and the requests of its invokers.
Recovery routines have four objectives:

- To isolate the error.
- To assess the damage and attempt to confine it to one user.
- To indicate the actions, such as dumping, that should be taken.
- To repair the error and clean up the routine for reinvoking.

The following paragraphs describe the four types of recovery routines: supervisor control program recovery, STAE/STAI, ESTAE/ESTAI/ESTAR, and functional recovery routines (FRR).

Supervisor Control Program Recovery: Recovery is provided to critical supervisor subcomponents that have high availability requirements and run disabled. Among these are the dispatcher, the lock manager, exit effectors, and the first level interruption handlers. Upon entry, each of these subcomponents notifies recovery management. If an error occurs, a recovery routine is entered, which can handle recovery requirements for each of these routines.

STAE/STAI: As in Release 1, a task issues a STAE macro instruction or ATTACH with the STAI option to intercept a scheduled ABEND. The recovery/termination manager gives control to the user's STAE/STAI recovery routine, which may perform pre-termination processing, diagnose the cause of ABEND, or specify a retry address to avoid termination. STAE/STAI recovery routines are task-oriented.

ESTAE/ESTAI/ESTAR: ESTAE/ESTAI/ESTAR extends the capabilities of STAE/STAI. ESTAE/ESTAI/ESTAR routines may receive control in more cases than STAE/STAI routines. Also, the recording of errors in SYS1.LOGREC is provided as an option to ESTAE. Like STAE/STAI, these recovery routines are task-oriented.

Functional Recovery Routines (FRRs): While STAE and ESTAE routines are task-oriented, functional recovery routines (FRRs) are function-oriented. FRRs are provided for locked control program functions and system services. They are established by using the SETFRR macro instruction. Each FRR is placed in a stack in the system; each stack represents a single path through the control program. When an error occurs in a path, the recovery/termination manager passes control to the FRRs in the associated stack.

Percolation

If a recovery routine cannot cause recovery for a failing routine, or if no recovery routine is associated with the failing routine, the recovery/termination manager passes the error to a recovery routine at the previous level of control. This process is called *percolation*. Percolation is supported throughout all ESTAE routines associated with a task: the routines will receive control, one at a time in LIFO order, until the error is handled or all routines have been unable to recover. Percolation also applies to FRRs: the FRRs in a stack are entered in LIFO order until the stack is exhausted or the error has been handled.

System Resource Management

The control program is designed to keep those address spaces in real storage which *both* best use system resources and meet the installation performance specifications (IPS), at any instant of time and under any workload in the system. To achieve this objective, the supervision of system resources is centralized in one component, the *system resources manager*.

The system resources manager includes the following functions, which are part of MVT and Release 1:

- The TSO driver.
- Automatic priority grouping (APG).
- Page replacement algorithm.
- Paging rate control.
- Real storage frame shortage prevention.
- Auxiliary storage slot shortage prevention.
- I/O load balancing.

In addition, the system resources manager includes the following new functions:

- Support of installation performance specifications (IPS).
- Control of real storage occupancy.
- Prevention (for a period of time) of swap-out of a private address space holding system resources (via ENQ) requested by another function.

The system resources manager is divided into the following four subcomponents:

- Interface program.
- Control algorithm.
- Workload management algorithm.
- Resource use algorithms.

Interface Program: The interface program handles most input to and output from the system resources manager so that the other subcomponents are isolated from changes to the control program. Authorized control program functions issue the SYSEVENT macro instruction to notify the system resources manager of events involving system resources. An example of such an event is an address space entering terminal wait. The SYSEVENT macro replaces the TSEVENT macro used in MVT and Release 1 to interface with the TSO driver.

Control Algorithm: The control algorithm gathers recommendations from the various other algorithms (for example, the workload management algorithm) and analyzes them to determine the swapping action that results in the best assignment of system resources under existing circumstances. An installation can influence this swapping decision by specifying weighting factors called *resource factor coefficients* to be applied to the recommendations of the resource use algorithms. To an installation with an unbalanced hardware configuration, the resource factor coefficients provide the means to give additional weight to the recommendations for the most limited hardware resource.

Workload Management Algorithm: The workload management algorithm ensures that the rate at which resources are provided to users corresponds to installation performance specifications (IPS). Using IPS, an installation specifies the service rate which its users should receive under any system workload conditions and during any period in the life of a job or command. (IPS is described in greater detail in the Features and Facilities section of this publication under the topic "Use of Resources.") The workload management algorithm replaces the tuning capabilities of the TSO driver, which is part of MVT and Release 1.

Resource Use Algorithms: The following algorithms are implemented in this subcomponent -

- External page storage and SQA shortage prevention, which prevents address space creation and directs swap-out when slot or SQA shortages exist.
- Device allocation, which attempts to prevent allocation to heavily utilized logical channels and attempts to place allocations for the same user on the same logical channel.
- ENQ/DEQ, which prevents (for a period of time) swap-out of users that are enqueued on system resources requested by other functions.
- Real storage frame shortage prevention, which directs swap-out of users when the number of available page frames falls below a specified limit.
- Real storage occupancy, which directs the swapping of users in order to keep the number of allocated frames near a specified value.
- I/O load balancing, which directs swapping of users to correct detected imbalances in the I/O subsystem. This algorithm is active only if SMF data set activity recording is active, since the algorithm uses SMF data set activity measurements.
- CPU load balancing, which ensures that all dispatchable address spaces are dispatched at reasonable intervals.
- Page replacement, which attempts to remove from real storage pages that are not likely to be referenced in the near future so that the frames are available to other users.
- Automatic priority group algorithm, which reorders a subset of the dispatching queue to give higher priority to users that quickly release the CPU.

I/O Supervisor

Redesigned management of I/O operations divides the I/O supervisor into two distinct areas: basic IOS and a set of IOS drivers. This division of responsiblity is designed to provide efficient support for several different types of I/O requests as well as flexibility and efficiency in a multiprocessing, virtual storage environment.

Basic IOS manages the I/O devices, control units, and channels. It queues unrelated requests, starts the I/O when a path is available, and performs initial interruption handling. All request parameters for basic IOS are centralized to reduce page-fixing requirements.

IOS drivers function as interface routines which provide the validity checking, page fixing, virtual to real translation, and related request queuing that is unique to a particular type of I/O request. All application programs interface with IOS through a driver, rather than directly. A separate driver has been defined for each of the following services:

- EXCP -- This general purpose driver processes a full range of I/O requests using the existing control block structure, provides the interface to VIO, and provides general compatiblity with MVT. Most IBM access methods use the EXCP interface.
- VSAM and paging requests -- This driver makes efficient use of the structure of VSAM channel programs.
- TCAM and VTAM -- These drivers provide telecommunications services.
- Program Fetch -- This separate driver is provided for performance.
- OLTEP -- This driver gives OLTEP the control necessary to effectively run diagnostic programs.

Virtual Input/Output (VIO)

An input/output operation reads data from or writes data to a data set on an I/O device. Release 2 supports a *virtual I/O (VIO)* operation that uses paging to transfer data into and out of a data set located in external page storage.

To use virtual I/O, the installation specifies one or more units (generic or esoteric) for VIO at system generation time. When an executing job or access method builds channel programs to create a system-named temporary data set on a device that was specified for VIO, VIO intercepts the channel program. Instead of allowing the data to pass over a channel to a device, VIO moves (via the MVC instruction) the data from the channel program buffers to a special buffer in the user's address space called a *window* (see Figure 18). This window is in addition to the channel program buffers and contains sufficient contiguous virtual pages to contain all the data that could be placed on a track of a real device. (Two pages are required for a 2314 track and four pages for a 3330 or 2305 track.)

When VIO issues the first MVC instruction, a page fault causes frames to be assigned to the window. One or more channel programs are used to fill the window. When the executing job or access method determines that the track is full, it builds another channel program to place data on another track. When VIO detects this track switch, it writes the contents of the window as 4K-byte

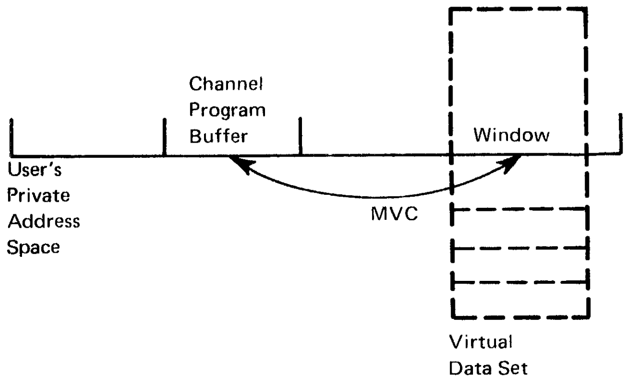


Figure 18. Virtual I/O

pages in external page storage using standard paging. The system provides special support to keep VIO data set pages in real storage after the page-out and disconnects the window from those frames containing the VIO data set pages. When VIO moves new data (the second track) into the window, a page fault occurs causing fresh frames to be assigned to the window.

As the data set is created and external page storage is assigned, VIO keeps a record of the locations of each page of the data set. The pages are not necessarily contiguous; they are allocated dynamically throughout external page storage as the data set is created.

When an executing job or access method builds channel programs to retrieve data from the VIO data set, VIO determines which pages contain the desired tracks. If VIO encounters a channel program that requires data from a track that is not in the window, VIO changes the appropriate page table entries to point to the required pages in external page storage. Then VIO uses a move character (MVC) instruction to move data from the window to the channel program buffers. This causes a page fault, and the proper page is brought into real storage and made addressable through the window.

Thus, VIO uses paging rather than explicit I/O to transfer data. VIO eliminates IOS channel program translation, page-fixing required for EXCP, and DADSM overhead. It also provides dynamic allocation of DASD space in 4K-byte increments. Another advantage of VIO is that the data set may remain in real storage after it is created; in this case, no I/O is required to retrieve data from the data set.

System Requirements

The following System/370 models are supported by OS/VS2 Release 2.

Model 145
Model 155II
Model 158 (including MP models)
Model 165II
Model 168 (including MP models)

The minimum real-storage requirements are
768K bytes concurrent batch, time sharing, JES2.
1,024K bytes concurrent batch, time sharing, JES2 or JES3.

The functional workload of a system, in conjunction with a minimal hardware configuration, may limit the number of user address spaces available for concurrent operation.

System Configuration

The minimum configuration for VS2 Release 2 (768K, JES2) is described below:

- One System/370 CPU and real storage. On the Model 145, the Clock Comparator and CPU Time feature (#2001), and the Advanced Control Program Support feature (#1001) are required.
- One multiplexer channel
- One selector or block multiplexer channel
- Three 3330 or 3333 Disk Storage devices
  or
  Four 2314 Direct Access Storage devices
  or
  Four 2319 Disk Storage devices
- One card reader.
- One printer.
- One hardcopy console, or one display console with hardcopy log.
- One 9-track tape drive for all distributions of program libraries and updates.
- One additional tape drive is recommended for the output of the high speed stand alone dump program (AMDSADMP). However, the drive mentioned above for the distributions may be used for the dump.
- If TCAM/VTAM/BTAM is specified, at least one transmission control unit or communications controller is required for operation of remote terminals (VTAM requires a 3704/3705 in NCP mode).

INPUT/OUTPUT Devices Supported

The following devices and control units are supported by OS/VS2 Release 2.

Direct Access Storage Devices
IBM 2305 Model 1 Fixed Head Storage (Models 165II and 168 only)
IBM 2305 Model 2 Fixed Head Storage
IBM 2314 Disk Direct Access Storage Facility
IBM 2319 Disk Storage
IBM 3330 Disk Storage Models 1 and 2
IBM 3333 Disk Storage

Direct Access Storage Control Units
IBM 2835 Storage Control Model 1 (Models 165II and 168 only)
IBM 2835 Storage Control Model 2
IBM 2844 Auxiliary Storage Control*
IBM 3830 Storage Control Model 2 (including 2-channel switch additional *8171)
* A maximum of 2 paths to a device from 1 CPU are permitted with this release.
IBM 3345 Storage and Control Frame Models 3, 4, and 5 (Model 145 only)
Integrated Storage Controls (Models 158 and 168 only)

Magnetic Tape Devices
IBM 2401 Magnetic Tape Unit
IBM 2420 Magnetic Tape Unit
IBM 2495 Tape Cartridge Reader
IBM 3410 Magnetic Tape Unit (Models 145, 155II, and 158 only)
IBM 3411 Magnetic Tape Unit and Control (Models 145, 155II, and 158 only)
IBM 3420 Magnetic Tape Unit

Magnetic Tape Control Units
IBM 2816 Switching Unit Model 1

Printers
IBM 1403 Printer Models 2, 7, and N1
IBM 1443 Printer Model N1
IBM 3211 Printer

Printer Control Units
IBM 2821 Printer Control Unit Models 1, 2, 3, and 5
IBM 3811 Printer Control Unit

Paper Tape Devices
IBM 2671 Paper Tape Reader
IBM 2822 Paper Tape Reader Control

Card Readers and Punches

IBM 2501 Card Reader Models B1 and B2
IBM 2520 Card Reader Punch
IBM 2540 Card Reader Punch
IBM 3505 Card Reader Models B1 and B2
IBM 3525 Card Punch

Reader and Punch Control Units

IBM 2821 Models 1, 5, and 6

MICR/OCR Devices

IBM 1287 Optical Reader
IBM 1288 Optical Page Reader
IBM 1419 Magnetic Character Reader (Dual Address Adapter #7730 and Expanded Capability #3800 features required.) This device must be run V=R.
IBM 3886 Optical Character Reader, Model 1

Consoles

IBM 2150 with 1052 Printer-Keyboard Model 7
IBM 2250 Display Unit Models 1 and 3
IBM 2260 Display Station Model 1
IBM 2740 Communications Terminal (Remote only)
IBM 3066 System Console (Models 165II and 168 only)
IBM 3210 Console Printer-Keyboard, Models 1 and 2 (or 155II only)
IBM 3213 Console Printer (Model 158 only)
IBM 3215 Console Printer-Keyboard
IBM 3270 Information Display System (local only)
System Console for the Model 158

BTAM and TCAM Terminals

Start/Stop Terminals
IBM 1030 Data Collection System
IBM 1050 Data Communication System
IBM 1060 Data Communication System
IBM 2740 Communication Terminal, Models 1 and 2
IBM 2741 Communication Terminal
AT & T 83B3 Terminal
System/7 (as 2740 Communication Terminal Model 1 with checking, leased and private lines only)
TWX Models 33 and 35
Western Union 115A Terminal
World Trade Telegraph Terminal
Binary Synchronous Terminals
System/3 Processor
System/360 Processor (including Model 25 Integrated Communications Adapter)
System/370 Processor (including Model 125 ICA and 135 ICA)
S/360 Model 20 Processor
1130 Computing System Processor
IBM 1800 Data Acquisition and Control System Processor (BTAM support only)
IBM 2770 Data Communication System
IBM 2780 Data Transmission Terminal
IBM 2790 Data Communications System/2715 Transmission Control Unit Model 2
IBM 2972 General Banking System Models 8 and 11 (BTAM support only)
IBM 3670 Brokerage Communication System (TCAM support only)
IBM 3270 Information Display System
IBM 3735 Programmable Buffered Terminal
IBM 3780 Data Transmission Terminal
Locally-Attached Terminals
IBM 2250 Display System (GAM and GSP support only)
IBM 2260 Display Station (GAM, GSP, and TCAM support only)
IBM 3270 Information Display System
Telecommunications Control Units
IBM 2701 Data Unit Adapter
IBM 2702 Transmission Control Unit
IBM 2703 Transmission Control Unit
IBM 2715 Transmission Control Unit, Model 1
IBM 2955 Data Adapter Unit (RETAIN 370 only)
IBM 3704 Local Communications Controller -- Emulation Program Mode or Network Control Program Mode (TCAM only)
IBM 3705 Local Communications Controller -- Emulation Program Mode or Network Control Program Mode (TCAM only)
IBM 7770 Audio Response Unit Model 3 (2721 and 2730 support included) (TCAM support only)

VTAM Terminals (attached via 3704/3705)

Except as noted, application programs can communicate with VTAM-supported terminals by using VTAM directly, or indirectly via TCAM.
Start/Stop Terminals (remotely attached)
AT & T 83B3 Terminal
Communicating Mag Card Selectric typewriter (as 2741 terminal)
System/7 processor station supported as a 2740 Model 1 with checking. Requires Asynchronous Communication Control (#1610)

- TWX Models 33 and 35
- Western Union 115A Terminal
- World Trade Telegraph Terminal
- IBM 1050 Data Communications System
- IBM 2740 Communication Terminal Models 1 and 2

Binary Synchronous Terminals (remotely attached)

- System/3 Processor Station
- System/370 Processor Station (including Model 125ICA and 135ICA)
- IBM 2770 Data Communication System
- IBM 2780 Data Transmission Terminal
- IBM 2792 General Banking System Models 8 and 11
- IBM 3270 Information Display System
- IBM 3735 Programmable Buffered Terminal
- IBM 3780 Data Transmission Terminal (2772 Compatible)

Locally-Attached Terminals

- IBM 3270 Information Display System

Telecommunications Control Units

- IBM 3704 Local Communications Controller - NCP Mode
- IBM 3704 Remote Communications Controller - NCP Mode
- IBM 3705 Local Communications Controller - NCP Mode
- IBM 3705 Remote Communications Controller - NCP Mode

Current Devices Not Supported In OS/VS2 Release 2

1013	1017/1018
1070	1255
1259	1270
1404-2	1442-N1,N2
2245	2301
2303	2311
2321	2402
2403	2404
2415	2560-A1
2596	2841
3881	

System Generation

The IBM-supplied starter system is used for the process of OS/VS2 Release 2 system generation the first time. This starter system will be an OS/VS2 Release 2 system, containing JES2. Any VS2 Release 2 system, with either JES2 or JES3, can be used for subsequent system generation.

Compatibility

OS/VS2 Release 2 is generally upward compatible with MFT, MVT, VS1 and VS2 Release 1. Object programs and load modules that operate in MFT, MVT, or VS1 and follow IBM programming conventions described in *OS/VS Supervisor Services and Macros* GC27-6979, *OS/VS Data Management Services Guide* GC26-3786, and *OS/VS Data Management Macros* GC26-3793, will operate without change in this release. Some exceptions are:

- Programs sensitive to the PSW format (if migration is from an OS system).
- Programs that modify, are modifications of, or depend upon implementation of, the MFT, MVT, VS1, or VS2 Release 1 control program.
- Programs sensitive to the OS catalog structure.
- Programs that reference areas not normally available to a problem program through system functions.
- In a multiprocessing environment with shared real storage, multi-tasking programs which reference the same area should be reviewed for synchronization dependencies.
- Programs dependent upon MVT or VS2 Release 1 integrity exposures.

The capability to execute with virtual addresses equal to real (ADDRSPC=REAL) for programs that must operate entirely in real storage is provided for:

- programs that use EXCP or XDAP with user written appendages.
  Note: Programs that use EXCP should observe the conventions outlined in the channel characteristics manual for the host system.
- programs that dynamically modify channel programs.
- programs that are time dependent (such as MICR).

Performance

The performance of OS/VS2 Release 2 depends on many factors, including workload characteristics, hardware/software system configurations, available real storage, etc.

The overall design objective of VS2 Release 2 is to have a positive effect on performance. Functions designed to accomplish this objective include such items as VIO, VSAM catalog, SWA, and the System Resources Manager. Additional functions have been added however, which require increased system resources. These functions, which include CCW and dynamic address translation, improved system integrity, enhanced recovery, etc., will tend to have a negative effect on system performance. Because of the effect of the complex interrelationships between the above factors, the extent of the use of OS/VS2 Release 2 enhancements, and individual system environments, no specific performance statement can be made.

Emulators

The following emulators (Figure 19) will be provided as system control programs. In an MP environment, the emulator features may be asymmetric, i. e., they may be installed on one specific CPU.

	145	158 155II	168 165II
1401/1440/1460	x	x	
1410/7010	x	x	
7070/7074		x	x
7080			x
709/7090/7094/7094-II			x
DOS	x	x	

Figure 19. Emulators

Functions Not Supported

MVT functions not supported in VS2 Release 2 are shown in Figure 20. Unsupported VS2 Release 1 functions are shown in Figure 21.

Unsupported MVT Function	Comparable VS2 Release 2 Function
Automatic SYSIN batching ASB reader	Job entry subsystem (JES2 or JES3)
BACKSPACE, XDAP, and EXCP for SYSIN/SYSOUT	----
Conversational remote job ertry (CRJE)	Time sharing (TSO)
Dedicated work files	Virtual I/O
Direct system output (DSO)	----
Graphic job processor (GJP)	----
IEBUPDAT utility program	IEBUPDTE utility program
IEHIOSUP utility program	Pageable link pack area
IEHMOVE, IEHLIST, IEHPROGM (most catalog functions)	Access method services
IMCJQDMP service aid	----
IMCOSJQD service aid	----
Main storage hierarchy support	----
OS readers and writers	Job entry subsystem (JES2 or JES3)
Queued telecommunications access method (QTAM)	Telecommunications access method (TCAM)
QTAM message processing programs	----
Remote job entry (RJE)	JES2 or JES3 remote job entry
Rollout/rollin	Paging
Satelite graphic job processor (SGJP)	----
Scatter load	Paging
System environment recording routines (SER0 and SER1)	Recovery management support (RMS)
OS system catalog (SYSCTLG)	VSAM master catalog
SYS1.ACCT data set	System management facilities (SMF) facilities (SMF)
SYS1.SYSJOBQE	JES input and output queues and the schedule work area (SWA)
TESTRAN program	----
Transient areas	Pageable link pack area
TSO Driver & Trace	System Resources Manager & MF/1
Tape output for SMF	----
IMBMDMAP	AMBLIST
CVOL catalog	VSAM user catalog

Figure 20. Unsupported MVT Functions

Unsupported VS2 Release 1 Function	Comparable VS2 Release 2 Function
BACKSPACE, XDAP, and EXCP for SYSIN/SYSOUT	- - - -
IEHMOVE, IEHLIST, IEHPROGM (most catalog functions)	Access Methods Services
IMCOSJQD service aid	- - - -
OS readers and writers	Job Entry Subsystem (JES2 or JES3)
QTAM message processing programs	- - - -
SYSCTLG (system catalog)	VSAM system catalog
SYS1.SYSJOBQE	JES input and output queues and the scheduler work area (SWA)
TSO Driver & Trace	System Resource Manager & MF/1

Figure 21. Unsupported VS2 Release 1 Functions

Glossary

This glossary provides definitions of OS/VS2 terms. For definitions of terms not included, see IBM Data Processing Glossary, GC20-1699.

ABP: Actual block processor.

ACR: Alternate CPU recovery.

actual block processor: One of several programs that translates I/O requests into the proper format for the I/O supervisor.

address space: The virtual storage assigned to a batch or terminal job, a system task, or a task initiated by the START command. Each address space consists of the same range of addresses.

address space identifier: A unique, system-assigned identifier for an address space.

address translation: The process of changing the address of an item of data or an instruction from its virtual address to its real storage address. See also dynamic address translation.

alternate CPU recovery: a facility that attempts system recovery when a CPU fails by transferring work to another CPU.

APF: Authorized program facility.

APG: Automatic priority group.

ASID: Address space identifier.

ASM: Auxiliary storage manager.

ASP: Attached Support Processor. See JES3.

Asymmetric I/O: The I/O devices can be started by any CPU, although physically attached to only one.

authorized library: A library that can contain authorized programs.

authorized program: A system program or user program that is allowed to use restricted functions.

authorized program facility: A facility that permits the identification of programs that are authorized.

automatic priority group: A group of jobs, each having its own address space at a single installation-specified priority level, that are dispatched according to a System Resources Manager algorithm that attempts to provide optimum use of CPU and I/O resources.

auxiliary storage: Data storage other than main storage.

auxiliary storage management: A facility that controls the allocation and release of pages on external page storage, and schedules I/O operations on the page data set.

basic control (BC) mode: A mode in which the features of a System/360 computing system and additional System/370 features, such as new machine instructions, are operational on a System/370 computing system. See also extended control (EC) mode.

BC mode: Basic control mode.

block processor: See actual block processor.

change bit: A bit associated with a page in real storage. The change bit is turned "on" by hardware whenever the associated page in real storage is modified.

channel program translation: In a channel program, replacement by software of virtual addresses with real addresses.

common area: The area of virtual storage that is addressable by all address spaces.

common service area: A part of the common area that contains data areas addressable by all address spaces, but protected during its use by the key of the requester. Abbreviated CSA.

communication task: A function that handles all communication between programs and operator consoles.

control program keys: Protection keys (0-7) that are reserved for control program use.

control registers: A set of registers used for operating system control of relocation, priority interruption, program event recording, error recovery, and masking operations.

converter: The routine that transforms spooled JCL statements into an internal format, and processes commands entered through the input stream.

CPU affinity: In a multiprocessing configuration, a function that allows a unit of work to be directed to a particular CPU for execution.

cross memory services lock: A global suspend lock that is used for services that apply to more than one private address space.

CSA: Common service area.

DAT: Dynamic address translation.

demand paging: Transfer of a page from external page storage to real storage at the time it is needed for execution.

disabled page fault: A page fault that occurs when interruptions are disallowed by the CPU.

DSS: Dynamic support system.

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

dynamic area: The portion of virtual storage that is divided into address spaces that are assigned to job steps and system tasks.

dynamic support system (DSS): An interactive debugging facility that allows authorized maintenance personnel to monitor and analyze events and alter data.

EC mode: Extended control mode.

enabled page fault: A page fault that occurs when interruptions are allowed by the CPU.

ESTAE/ESTAI exit routines: Task recovery exit routines from which percolation to a higher-level exit may occur.

extended control (EC) mode: A mode in which all the features of a System/370 computing system, including dynamic address translation, are operational. See also basic control (BC) mode.

external page storage: The portion of auxiliary storage that is used to contain pages.

external page table (XPT): An extension of a page table that identifies the location on external page storage of each page in that page table.

external slot: See slot.

fixed: Not capable of being paged out. A fixed page in a private address area can be swapped out.

fixed BLDL table: A BLDL table that the user has specified to be fixed in the lower portion of real storage.

fixed link pack area: An extension of the link pack area that occupies fixed pages in the lower portion of real storage.

fixed page: a page in real storage that is not to be paged out.

fragmentation: The inability to assign real storage locations to virtual addresses because the available spaces are smaller than the page size.

frame: Same as page frame.

FRR: Functional recovery routine.

functional recovery routine: A recovery routine that is used by locked programs, SRBs, and supervisor control routines.

global: Pertaining to more than one private address space.

global lock: A lock that protects serially reusable resources related to more than one private address space.

global services: Services that apply to more than one private address space. Also used when referring to multiple JES3 processors.

graphics access method (GAM): A facility that supports the 2250 Display Unit and the 2260 Display Station through the use of graphic programming services (GPS) and the graphic subroutine package (GSP).

hardware error recovery management system: A facility that attempts recovery from hardware malfunctions. It consists of the machine check handler (MCH) and the channel check handler (CCH).

HASP: See JES2.

installation performance specification (IPS): A set of installation-supplied control information used by the Workload Manager. An IPS includes performance group definitions, performance objectives, and coefficients used to establish the service rate.

internal writer: A facility in the job entry subsystem (JES2 or JES3) that allows user written output writers to write data on devices not directly supported by the job entry subsystem.

internal reader: A facility that transfers jobs to the job entry subsystem (JES2 or JES3).

interprocessor communication: In a multiprocessing system, a function that provides communications between CPUs that are under control of the same control program.

interval service value: In the SRM, a category of information contained in a period definition, which specifies the minimum amount of service that an associated job will receive during any interval.

invalid page: A page that cannot be directly addressed by the dynamic address translation feature of the central processing unit.

IPC: Interprocessor communication.

IPS: Installation performance specification.

JES2: A functional extension of the HASP II program that receives jobs into the system and processes all output data produced by the job.

LCH: Logical channel queue.

LGN: Logical group number.

link pack area (LPA): An area of virtual storage containing selected reenterable and serially reusable routines that are loaded at IPL time and can be used concurrently by all tasks in the system.

link pack update area: An area in virtual storage containing modules that are additions to or replacements for link pack area modules.

local lock: A suspend lock that protects the resources assigned to a particular private address space.

local service: A supervisory service that applies to only one private address space. Also used when referring to a single JES3 processor.

local system queue area (LSQA): One or more segments associated with each address space that contain system control blocks.

lock: A means of serialization used by supervisory control program routines. See global lock, local lock, spin lock, suspend lock.

logical channel queue: A queue of operations to be performed on a channel. Abbreviated LCH.

logical group: A collection of pages that are related to each other. An address space may be composed of multiple groups: one for the LPA, one for the SWA, and one for the private address space.

logical group number: An identifier of a logical group. It is the base value used by the real storage manager and the auxiliary storage manager to compute the identifier of a logical page. All the pages of a VIO data set may be represented by a single LGN.

LPA: Link pack area.

LSQA: Local system queue area.

master address space: The virtual storage used by the master scheduler task.

MCP: Message Control Program.

Message Control Program: A program that is used to control the sending or receiving of messages to or from remote terminals.

MF/1: System activity measurement facility.

MIC: Missing interruption checker.

missing interruption checker: A program that periodically tests active I/O operations in order to detect missing I/O interruptions and inform the operator ob abnormal conditions before they can affect system operation.

MP: Multiprocessing

multiple address space: A feature that provides each user with a private address space.

node: An addressable point in a teleprocessing network.

page: (1) A fixed-length block of instructions, data, or both, that can be transferred between real storage and external page storage. Note that page and page frame are not synonymous. It is important to preserve the distinction between a page -- the data which is transferred between real and external page storage -- and the frame or slot in which it can reside. (2) To transfer instructions, data, or both, between real storage and external page storage.

page data set: A data set in external page storage, in which pages are stored.

page fault: A program interruption that occurs when a page that is marked not in real storage is referred to by an active page. Synonymous with page translation exception.

page fixing: Marking a page nonpageable so that it remains in real storage.

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

page-in: The process of transferring a page from external page storage to real storage.

page-out: The process of transferring a page from real storage to external page storage.

page reclamation: The process of making addressable the contents of a page in real storage that has been marked invalid.

page stealing: To take away an assigned page frame from a user to make it available for another purpose.

page table (PGT): A table that indicates whether a page is in real storage and correlates virtual addresses with real storage addresses.

page translation exception: A program interruption that occurs when a virtual address cannot be translated by the hardware because the invalid bit in the page table entry for that address is set. Synonymous with page fault. See also segment translation exception, translation specification exception.

paging: The process of transferring pages between real storage and external page storage.

paging device: A direct access storage device on which pages (and possibly other data) are stored.

paging rate: The average number of page-ins and page-outs per unit of time.

PER: Program event recording.

percolation: In error recovery, the passing of control to a higher-level recovery routine from another recovery routine along a preestablished path.

performance group: A class, specified by an installation, which regulates the turnaround time of the user's jobs, job steps, and interactions.

performance objective: A category of information contained in an IPS. Each performance objective specifies the service rate(s) that an associated job is to receive, for a number of different workload levels.

period definition: In the SRM, a category of information contained in a performance group definition, that indicates which performance objective is to be followed, either during a particular real-time period or until a particular amount of service has been accumulated by an associated job.

PGT: Page table.

private address space: An address space assigned to a particular user.

program event recording (PER): A hardware feature used to assist in debugging programs by detecting and recording program events.

program management: Routines that perform supervisory services related to the moving of programs from the system library or user libraries to virtual storage.

quick cell facility: A high-performance storage allocation technique using a fixed block size.

RAS: Reliability, availability, serviceability.

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

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.

real storage management: Routines that control the allocation of pages in real storage. Abbreviated RSM.

reconfiguration: The process of placing a CPU, main storage, or channel offline for maintenance; and adding or removing components while the system is in operation.

recovery routine: A routine that is entered when an error occurs during the performance of an associated operation. It isolates the error, assesses the extent of the error, indicates subsequent action, and attempts to correct the error and resume operation.

recovery termination manager: A program that handles all normal and abnormal termination of tasks by passing control to a recovery routine associated with the terminated function.

reference bit: A bit associated with a page in real storage; the bit is turned "on" by hardware whenever the associated page in real storage is referred to. There is a reference bit in each of two storage keys associated with each page frame.

relocate hardware: See dynamic address translation.

relocation interrupt: See page fault.

remote terminal access method (RTAM): A facility that controls operations between the job entry subsystem (JES2 or JES3) and remote terminals.

response/throughput bias (RTB): In the System Resources Manager, a category of information contained in a period definition, which indicates how the workload manager is to weigh trade-off between satisfying some system throughput objective, and satisfying the IPS-specified service rate.

RSM: Real storage manager.

RTAM: Remote terminal access method.

RTB: Response/throughput bias.

scheduler work area (SWA): One or more segments in virtual storage that contain most of the job management control blocks, such as the JCT, JFCB, SCT, and SIOT. There is one SWA for each initiator.

secondary storage: Auxiliary storage.

segment: A contiguous 64K area of virtual storage.

segment table (SGT): A table used in dynamic address translation to control user access to virtual sstorage segments. Each entry indicates the length, location, and availability, of a corresponding page table.

segment translation exception: A program interruption that occurs when a virtual address cannot be translated by the hardware because the invalid bit in the segment table entry for that address is set. Synonomous with program interruption code 16.

service: See service rate.

service rate: In the workload manager of the SRM, a measure of the rate that system resources (service) are provided to individual jobs. It is used by the installation to specify performance objectives, and used by the Workload Manager to track the progress of individual jobs. Service is a linear combination of CPU, I/O, and main storage measures which can be adjusted by the installation.

service unit: Same as service.

serviceability: The capability to perform effective problem determination, diagnosis, and repair on a data processing system.

slot: A continuous area on a paging device in which a page can be stored.

software recording facility: A facility used by functional recovery routines (FRRs) to write system error records.

spin lock: A lock that prevents a CPU from doing work until the lock is cleared. Contrast with suspend lock.

SQA: System queue area.

SRB: Service request block.

SRM: System resources manager.

suspend lock: A lock that prevents requesters from doing work on a CPU, but allows the CPU to continue doing other work. Contrast with spin lock.

SWA: Scheduler work area.

swapping: A paging technique that moves the active pages of a job and its LSQA to external page storage, and moves pages of another job from external page storage into real storage.

symmetric processors: Processors with identical configurations.

symmetric storage configurations: Machine configurations with identical storage units.

system activity measurement facility: A facility that collects information such as paging activity and the use of the CPU, channels, and I/O devices, to produce trace records and reports.

system key: A key that protects system data from damage or modification by unauthorized users.

system queue area (SQA): An area of virtual storage reserved for system-related control blocks. It contains fixed pages and is assigned protection key zero. Abbreviated SQA.

system resources manager (SRM): A group of programs that controls the use of system reousrces in order to satisfy the installation's performance objectives.

thrashing: A condition in which the system can do little useful work because of excessive paging.

timer supervision: Routines that provide the date and time of day, measure time intervals, schedule activities, and set the interval timer.

TPIOS: A facility that supports programmed telecommunications control units (TCUs) and generates channel programs for the channel scheduler.

transaction: A batched job, job step, or a terminal interaction identified for performance measurement.

translation tables: Page tables and segment tables.

uniprocessing: Sequential execution of instruction by a central processing unit or independent use of a CPU in a multiprocessing system.

VIO: Virtual I/O.

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

virtual equals real address area (V=R): An area of virtual storage that has the same range of addresses as real storage, and is used for a program or part of a program that is not to be paged or swapped during execution.

virtual I/O: A facility that allows a page-formatted data set to be used as if it were a data set on a virtual device. Abbreviated VIO.

virtual memory: See virtual storage.

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 management (VSM): Routines that allocate address spaces and virtual storage areas within address spaces, and keep a record of free and allocated storage within each address space.

VSAM: An access method for direct or sequential processing of fixed and variable length records on direct access devices.

VSM: Virtual storage management.

V=R area: See virtual equals real address area.

workload manager: A part of the system resources manager (SRM) that allows an installation to determine the performance that any group of users will receive, monitors the workload, and schedules resources accordingly.

Index

Indexes to OS/VS publications are consolidated in the **OS/VS Master Index**, GC28-0602, and the **OS/VS Master Index of Logic**, GY28-0603. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

Introduction to OS/VS2 Release 2

GC28-0661-1

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

Cut or Fold Along Line

What is your occupation? ____

Number of latest Technical Newsletter (if any) concerning this publication: ______________

Please indicate in the space below if you wish a reply.

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.

GC28-0661-1

Your comments, please . . .

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Cut or Fold Along Line

Fold Fold

First Class
Permit 81
Poughkeepsie
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department D58, Building 706-2
PO Box 390
Poughkeepsie, New York 12602

Fold Fold

Introduction to OS/VS2 Release 2 (S370-34) Printed in U.S.A. GC28-0661-1

IBM®

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

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