File Number S/360-00
Form C20-1647-0

IBM Systems Reference Library

System/360 Model 67

Time Sharing System

Preliminary Technical Summary

This document is being furnished for System/360 planning use only. It contains preliminary technical information, and the contents are subject to change. The content represents "best available information" as of the publication date. This material is to be considered automatically replaced by the normal SRL publication upon its release.

The System/360 Model 67 Technical Summary is a self-contained description of the system, its components, and the Time-Sharing System programming support. The objective of the document is to provide consistent material to support branch office preparation of customer proposals. Performance information unique to the Model 67 is provided. A brief section on reliability and maintainability is included.

Copies of this and other IBM publications can be obtained through IBM branch offices. Address comments concerning the contents of this publication to IBM, Technical Publications Department, 112 East Post Road, White Plains, N. Y. 10601

CONTENTS

SYSTEM FEATURES

The basic architecture of the IBM System/360 makes it ideally suited to processing in a multiprogramming and multiprocessing environment. The Model 67 extends this basic architecture to provide the additional capabilities of an advanced time-sharing system, offers multiple users real-time processing, efficient conversational-mode operation, and almost instantaneous turnaround. The Model 67 is designed to:

- Enable all processors in a system to access directly or to control all components of the system, including core storage units, I/O channels, and I/O control units. The facilities of the system can be dynamically allocated to meet changing job requirements.
- Allow direct connection of channels to the processor storage units. Thus, I/O transfers to storage are independent of the processor.
- Enable each processor of a multiprocessor system to operate:
  1. As a single processor with its own I/O subsystem.
  2. Jointly with other processors in a symmetric multiprocessing configuration.
- Allow extensive growth without impacting programming support or programming compatibility, and without disrupting the total system operation. For example:
  1. For additional computing power growth, a total of four processors can be attached.
  2. For additional core storage requirements, up to eight processor storage units can be attached to provide more than 2 million bytes of physical core storage.
  3. To gain additional system throughput, many channels, channel controllers, and hundreds of I/O devices can readily be added.
- Enable system components to be physically partitioned from the rest of the system by centrally-located switches. Multiple computing systems can therefore operate independently.

SYSTEM COMPONENTS

Processor

The System/360 Model 67 time-sharing processor has the full capabilities of the System/360 processor plus additional features to strengthen its capabilities for time sharing. Some of these features are:

- Extended control capability. Masking facilities are extended so that a processor can control all channels in a system. The system can address up to 28 channels.
- Dynamic address translation (dynamic relocation). Special hardware and associated support programming permit each user to program as though he had sole use of a large, contiguously addressable virtual memory, which may be very much larger than the size of the physical core memory present. The programmer need not be concerned with the size of physical memory or with other users who have simultaneous claims on memory. Only active blocks (pages) of each program reside in physical memory. They may be placed temporarily on a secondary storage device and returned to different areas of core storage regardless of their execution status. Without this feature, it is very costly in time and space to activate many programs simultaneously as is necessary in a true time-sharing environment. The address translate feature eliminates the need for the programmer to organize large programs into overlays and allows simultaneous responses to a large number of users even though their total memory demands far exceed actual memory.
- Extended storage protection. The System/360 provides protection of storage from erroneous updating. This is especially important in a time-sharing mode. The Model 67 has extended protection so that even unauthorized reading of storage is prevented. Two bits are added to the store and fetch protect feature which monitor the use of blocks of memory (reference and change).
- High-resolution timer. A timer is provided with a 13-microsecond interval. Automatic interruption occurs on clock runout to permit control to be transferred rapidly between programs. Only with this feature can the monitor effectively assign resources in a time-sharing environment.

Processor Storage

The Model 67 utilizes core storage units with a cycle time of 750 nanoseconds per doubleword (64 bits + 8 parity). These units are available with a capacity of 262,000 bytes. The total core storage capacity supported ranges from 262,000 bytes to

over 2 million bytes. References within each unit can be interleaved to obtain effective storage cycles approaching one-half the cycle time of the unit. Although there are delays caused by storage access conflicts and cable length considerations, these storage units are capable of supplying the demands of multiprocessor configurations with large amounts of I/O activity without serious system performance degradation.

I/O Components

Increase in system throughput is highly dependent upon the ability of the system to increase the number of channels and I/O devices. The Model 67, with its powerful I/O control element (channel controller), provides the ability to greatly increase system throughput and offers maximum flexibility in system configuration.

Some of the I/O highlights are:

- Channel controller provides for maximum data rates up to 6,400,000 bytes per second
- Four channel controllers can be attached to a system
- As many as 28 channels in a system (7 per channel controller)
- I/O data transfer proceeds concurrently with processor operation
- Each processor in a system can address all channels in the system (addressing capability provided for 28 channels)
- I/O devices can be shared by the interconnecting channels or device

Remote Devices

A wide range of terminals for both low-volume and high-volume jobs can be remotely attached to the System/360 Time-Sharing System. The terminals include manual keyboards, printers, and visual displays.

In addition, smaller computers, such as the IBM 1800 Data Acquisition and Control System and the Models 20 and 30 of the System/360, can be remotely attached through the IBM 2701 Data Adapter Unit.

PROGRAMMING SYSTEM

A fundamental of IBM programming systems support has always been to furnish a computer system with in-depth programming systems so that the computer is used not only efficiently but also conveniently.

This becomes increasingly important with a time-sharing system that offers its full facilities to a large number of users.

The Time-Sharing System (TSS) employs new concepts, such as dynamic relocation, multiple access to system components, and extended storage protection, and furnishes a wide range of remote terminals.

To give users fast, efficient service the operating system must be designed to take full advantage of these features. Also, to permit easy, remote use of the computing system, the supervisor incorporates a wide range of user-oriented programs and languages. Close man-machine relationship is essential.

IBM is furnishing such a supervisor with the Time-Sharing System--a time-sharing, multiprogramming supervisor with assemblers, compilers, and debugging facilities designed specifically for it.

The new Time-Sharing System contains many of the features found in Operating System/360. Much of the task scheduling and interrupt handling is similar to Operating System/360. In addition, TSS contains special features specifically designed for efficient use of terminals, multiple processors, and virtual storage.

The system features a flexible and powerful command set for remote terminal users and a conversational mode of operation with terminals to perform line-by-line editing during input. With minor exceptions, the system is source-language-compatible with Operating System/360 language processors.

The Time-Sharing System includes:

- Time-sharing control program for efficient facilities scheduling
- A mnemonic assembly language compiler with macro capability -- batch and conversational syntax analysis modes
- A FORTRAN IV compiler -- batch and conversational syntax analysis modes
- A Programming Language One (PL/I) compiler -- batch and conversational syntax analysis modes
- COBOL compiler -- batch mode
- A sort/merge program
- A terminal command language to remotely process and manipulate data and programs
- A library of mathematical and utility programs, open-ended so that user-developed programs and routines can be added easily
- Data management and cataloging facilities

The Monitor

The Time-Sharing System is the interface between the equipment and the users. Many of the functions traditionally considered part of the supervisory system are implemented in TSS as service routines subject to relocation in user's memory. For this

purpose, the system is structured into two principal subsystems: a supervisor that supervises and allocates the equipment configuration, and command programs that enable remote users to process and manipulate data and programs. The programming design is organized to separate machine-oriented functions from user-oriented functions. This results in a compact monitor that:

1. Maintains status indices of system facilities (attachments, assignments, and usages), processing times, storages (primary core units and secondary drums), I/O devices (tapes, peripheral equipment, and terminals), and files (disks).

2. Supervises all interrupts.

3. Records basic data for job scheduling and for determining individual user system charges.

4. Allocates the resources of the system to achieve short response times to the users of the system. To meet this requirement, the time-sharing monitor responds rapidly to all interrupts and has an efficient processing scheduling algorithm that controls the processing time-slice allotted to active programs. The monitor is designed so that this scheduling algorithm can be modified to meet any changing requirements of the operating environment.

5. Achieves high utilization of core storage through the use of reentrant coding that allows multiple independent programs to use the same copy of a subroutine, compiler, etc.

6. Provides the nucleus of a recovery program so that when some portion of the system malfunctions, the rest of the system can keep operating. This function may be expanded by the user at his option so that, for example, the system can maintain a fast response time for high-priority users, or retain the same number of terminal users with a slower response time.

Other functions, such as error routines, checkpoints, and restarts, are separated from the monitor and are performed by a modular set of routines that are:

- Independent of the monitor.
- Not necessarily permanently resident in core.
- Capable of shared usage -- that is, they are reentrant.
- Identical with the problem programs written by system users except for priority (for processor time and space) and for a privileged authorization to use and change the status indices.

This approach is used because:

1. Core space for system residence is kept to a minimum.

2. The startup-shutdown, checkpoint-recovery, and reconfiguration procedures are simplified, since a minimum of information must be periodically recorded.

3. Complete flexibility is provided for the installation to revise or write its own routines.

Program Support

Development of the supervisor is only one step in the development of a practical real-time, time-sharing system. To be convenient to use, the system must also have conventional program support such as compilers and utility routines. This support must be capable of conversational use with remote terminals.

A number of programs are designed specifically for support of remote terminals or consoles:

- An assembly program is designed to operate under the time-sharing monitor. The mnemonics, data definitions, macros, etc., are for the most part similar to those of Operating System/360. However, this assembly program is reentrant and allows for entry through a terminal of source language programs with editing and diagnostic checking in a conversational mode. Symbolic updating of the program is possible at any time after initial entry of the program into the program library; symbolic listings are available at any time. Storage is allocated at execute time instead of during assembly time.
- The FORTRAN IV provided with TSS is designed to be reentrant and to produce reentrant code. The FORTRAN IV system may be used in either batch or conversational syntax analysis mode, and is source-language-compatible with the OS/360 FORTRAN IV. Some debugging commands of the QUIKTRAN type are provided.
- TSS includes a PL/I compiler, designed for use in batch and conversational syntax analysis modes. The compiler is reentrant and produces reentrant code.
- The COBOL may be used in batch/background mode only and is similar to OS/360 COBOL.
- The sort/merge program provides a capability for this function under the time-sharing mode of operation.
- The provided terminal command language is open-ended and may be extended indefinitely. It provides online functions for program or data manipulations and program debugging facilities, and offers the user tremendous flexibility in working with the time-sharing system.

The time-sharing supervisor and the programming systems provided with it constitute an effective means of utilizing the data processing power of the System/360 Model 67 multiprocessor system.

SYSTEM PHILOSOPHY

IBM's Model 67 Time-Sharing System provides a solution to the problem of optimizing the relationship of the man and his problem to the computer and its facilities. Through a combination of machine and program features, the system described appears to everyone using it during the same time period as a complete computing system. These users present a wide variety of applications to the system, ranging from conversational compiling to conventional batch-processing jobs. Since each application receives a share of the available time, many jobs are performed simultaneously within a given time period.

Using TSS, the individual job may take more time than if it occupied the entire computer facilities exclusively until completion, but far more use of the computing system is made. The major saving of time sharing, however, is in the reduction of the overall time between problem definition and solution (turnaround time). The use of terminals at remote locations permits the person defining a problem to call in the facilities of a complete computing system to his desk (the terminal). Thus time sharing eliminates much of the normal delay in human effort associated with the turnaround time problem of a batch system.

The Model 67 system is characterized by a close interaction of machine and program features. The objective is a time-sharing, multiple-access, multiprocessor, general purpose, online computing system. A description of these various characteristics follows.

- Time sharing. This technique interleaves many different tasks in a computer system by using a timing mechanism, in conjunction with a supervisor program's scheduling and dispatching routines, to signal the end of the time interval given to one task and permit the transfer of a processor to another task. Although it is analogous to the communications technique of time-division multiplexing, computer time sharing makes use of hundreds of thousands of instruction executions per time division, whereas communication multiplexing is at the bit level, per time division. It differs, also, in that communications multiplexing is for fixed intervals over a fixed number of channels, whereas time sharing is for varying intervals over a varying number of tasks. In conventional multiprogramming, control is taken from one task and given to another only when the first task is incapable of using, or indicates that it does not wish to use, a processor. Time sharing also uses this means of task switching. By additionally using the interval timer to transfer control from one executing task to another, however, time sharing provides a service that generally makes the most efficient use of the total computer facility and also facilitates the online multiple-access nature of the system by providing response times compatible with human reaction times.

- Multiple access. In contrast to conventional computing systems, where only one task is performed in a specific period of time and only one system input and system output device need be defined, many such devices are defined in a multiple-access system at any one time, one for each task currently being performed in the system. A multiple-access system allows many users to share the total system facility concurrently, as if each had sole use of the system. It also permits terminal intercommunication, making possible several new ways of using computer systems, such as gaming situations, multiconsole teaching situations, and design team computer systems. One reason for providing multiple access in online systems is that one user does not usually require the total system facility on a continuous basis. If, in such a case, multiple access is not provided, a brief pause by a user idles the entire system and its capability to serve. This unused capability can be profitably deployed to serve some other user at the same time, the cost of doing this being small compared with the facility time recovered.

- Multiprocessor. The number of jobs that can be handled in a time-sharing system is increased by adding processors in parallel. The operation of processors in parallel is the mode normal to the system described in this manual. Other modes, such as partitioned systems and direct-coupled master-slave systems, are possible. In the parallel processor operation, the same monitor can be executed simultaneously by each of several processors independently executing programs, accessing memory, and controlling input/output. Programmed and wired-in interlocks are provided to prevent interference between processors. The monitor itself is reentrant (not changed by execution), and simultaneous usage presents no problems. Some sections of the monitor must modify tables of system data. To prevent confusion, these modifications must be made sequentially, rather than in parallel. In these circumstances, the first processor to begin modifying such tables will lock out all others until its modification has been safely completed. These lockout procedures are inserted only when necessary, and the lockout time is held to a minimum.

- General purpose. This system is not limited to operating in just one mode, such as online interpretive FORTRAN systems, QUIKTRAN, or "desk

calculator" systems. The system is capable of performing in every way as conventional batch-processing operating systems, and yet can do much more. Teleprocessing, real-time monitoring and multiprocessing types of jobs can be handled by the time-sharing system as by-products of its extremely versatile basic design.

- On line. This system was designed primarily to couple users of computation facilities more closely to computer systems. This close user-system contact is provided at several levels within the system described in this manual. For example, it is possible not only to economically employ console-type debugging techniques at a terminal, but also to employ interpretive and conversational modes of operation that may considerably shorten the overall time to complete the user's job. Furthermore, the immediate access time enables a great many relatively short jobs to be performed in a reasonably short period of time without incurring significant delays. In addition, by employing large, direct-access storage devices (typical of the system described here), users may gain access to the data and programs of interest without losing time because of operator intervention. Also, the online situation does not require physical proximity of the attached terminals. All the apparatus of communications technology (communication lines, dial connections, etc.) may be used to convey computational power over any distance.

GENERAL CONCEPTS

The IBM Model 67 Time-Sharing System is a general purpose, remote computing system. A remote computing system enables a terminal operator to use computing equipment at his own convenience, that is, at the times and places of his own choosing. A general purpose, remote computing system is one that will accommodate a wide variety of problems, expressed in a variety of computing languages (including the language of the computer itself) introduced from several different types of sources.

The user of a remote computing system operates an input/output device called a terminal. The terminal may be adjacent to the computer itself, or it may be hundreds of miles from the computer. In either case, communication between computer and terminal occurs via standard communication devices or direct cabling.

The simplest kind of terminal is a typewriter-like device, with the keyboard serving as the input unit to the system, and the printing mechanism as the output unit. More complex terminals, such as the IBM 2250 Display Unit, contain both a keyboard and a cathode ray tube display. These displays are powerful tools for the generation and control of graphic material.

Any remote computing system is capable of handling many terminals. The time-sharing type of remote computing system is designed to provide each user at each terminal with essentially immediate response. It does so by taking advantage of the fact that a modern computer can generate responses much faster than the terminal user can ask questions.

Time slicing is a technique used in remote computing systems to provide equal response to a number of simultaneous computer users. In a time-sharing system utilizing time slicing, the many programs that may be executed are (1) swapped successively into high-speed core storage from some auxiliary device, usually a drum, (2) executed for fixed periods of time, and (3) swapped out again. Each program thus gets a slice of computer time periodically. Within a short cycle of time, all programs are in some sense responded to.

Paging

The 24-bit addressing capability of the standard System/360 allows a maximum of 16 million addressable bytes. In practice, however, the programmer is restricted to using addresses that represent the actual physical storage on his machine. Thus, the programmer does not have the ability to write a program addressing 16 million contiguous bytes. If his total program size exceeds the physical core available, he must overlay his program. The System/360 Model 67 processors remove this restriction by allowing the problem program to address a virtual memory limited only by the page table handling capacity of the memory system.

The installation may establish the maximum number of addressable bytes that each programmer/user may address, representing his virtual memory. (Virtual memory is defined as the maximum addressing capability of a computer system.) A program's demand on virtual memory is a subset defined as logical memory, that is, the total claim actually made on virtual memory by a program.

In a multitasking environment, the supervisor has the job of placing active programs into whatever actual core storage is available. The program runs in core storage, but because of space limitations or relocation of programs, the logical program may be scattered throughout core storage. The execution of programs requires that this physical fragmenting of programs be transparent to the user, so he can think of his program as being in contiguous logical memory locations. Indeed, his listings depict these logical addresses to him.

The more users that are in core storage, the faster the control can be transferred to another waiting user and the lower the resulting overhead

cost. In Model 67 systems, the basic element for relocation (and fragmentation) is a 4096-byte unit called a page. By breaking programs up into pages, core memory may be allocated in page increments. Only the actively used pages of a program are brought into core storage. At execution time, core memory holds only a small part of a user's program, but, more important, it holds only the active parts.

The heart of the paging technique is a hardware development called dynamic address translation. It involves the transformation of one address into another by means of a table-lookup procedure implemented in hardware. The tables provide mapping of the task's virtual memory into core memory. Each task in the system requires, for its operation, one segment table and a page table for each segment used. These tables are developed by the supervisor, as the task is created and as new page requirements are made by the task. As a required page is fetched into core storage, the supervisor enters in the page table the actual location of the logical page.

During program execution, the equipment automatically does a table lookup on each address as it is referenced by the user, and the actual address is chosen for each logical address reference. If the user references a location in a page that is not in core storage, the supervisor receives an automatic interrupt; the supervisor then sets up a "page-turning" routine to fetch the missing page, and assigns the processor to another task in the queue. The waiting program is held in the wait status until its required page has been assigned and fetched somewhere in core storage. It is then returned to the queue of active users sharing processor time. This activity is not apparent to the user, being performed without his knowledge.

"Page turning" has the following effects:

- The entire program need not be in core storage to operate. Bits and pieces of many programs may be present, and several may be in operating condition. The system, therefore, has many opportunities to do useful work while swapping a page.
- Program "swap time" is greatly reduced as an overhead factor, since only active pages of a program ever require movement between core storage and auxiliary storage.
- The size of core storage ceases to have any significance to a programmer. Although written and executed as a classical set of contiguous instructions and working space, a program may exist in the machine as scattered pages. The programmer's concept of the program in execution becomes "virtual memory" rather than actual core storage.
- With so much virtual memory space available, it becomes possible to extend the page-turning concept to input/output operations upon prestored data files. If a program in virtual memory is operating upon a data set in the system's bulk files, the file space which the data set occupies can be treated as an extension of the virtual memory, and can be operated upon directly without explicit Read or Write commands. Again, as each "page" of the data set is referred to, it will be page-turned into core storage by the system. The effect, to the programmer, is the same as if the entire file were always present in high-speed memory.

The page-turning concept (Figure 1) provides for efficient use of the equipment, while allowing the system to handle many programs in rapid succession. In addition, it frees the computer user from arbitrary bounds of core memory size.

Segmentation

Just as space need not be allocated in core memory for currently inactive portions of a user's program, auxiliary storage space need exist only for portions of the user's virtual memory actually used by his program, that is, his logical memory. The user can take advantage of this in setting aside areas of virtual memory large enough to hold data sets or tables of very large size, without tying up storage for such tables except to the extent that they are actually used.

Such use of virtual memory is facilitated in the Model 67 by dividing virtual memory into 1,048,576-byte units (256 pages) called segments. Virtual memory consists of 16 (or optionally 4096) segments depending on whether 24- or 32-bit addressing is utilized, and logical memory thus can consist of up to 16 (or optionally 4096) segments, each of which can contain from 1 to 256 pages. A segment can contain programs and data, or can constitute a million-byte area for working space.

Segmentation is implemented by making the table lookup of address translation a two-stage

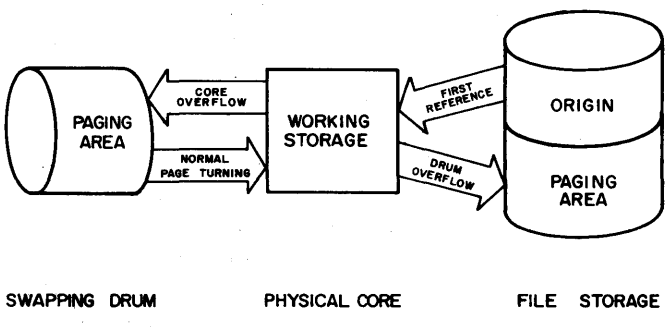


Figure 1. Page-turning concept

process. Instead of a single page table for the task, there exists a page table for each segment. The high-order bits of the page number are interpreted as a segment number, and a segment table entry is accessed to obtain the address of the page table to be used in the second stage of translation.

The advantage of implementing segmentation in the hardware is that it is possible to load virtual memory sparsely (thereby leaving room for data sets to grow), without requiring page table entries for unused pages.

Reentrant Code

The time-sharing system is based on reentrant coding. A reentrant program is one that is not in any way modified by execution. Such a program can be shared by several other time-shared programs that can be executed by one or more processors. All parameters, data locations, and working storage spaces are passed to the reentrant program. The integrity of each user's data is maintained regardless of the number of users who have entered the program.

Reentrant coding of programs enhances system efficiency. In situations where there is multiple usage of a single program or subroutine, only one copy need be present in memory or on the paging drum. It also reduces overhead for paging because of the read-only nature of the coding.

Data Set Management

To the user, a remote computing system can be a file space as well as a computing device. The user need not keep programs (that is, problem statements) or data files on his own premises. They can be introduced to the system, left within it, and recalled and manipulated at will.

Therefore the system must maintain large sets of files with complete safety and security. To this end, the system contains a catalog of all data files within its storage devices, and provides many modes of protection for these files. A user can specify that his files be accessible to himself only or to specific individuals or groups other than himself. Various modes of access can be specified. Some accesses may be for read-only purposes, others for both reading and writing.

The system can also regulate the location of files in accordance with the activity against them. In general, storage devices range from small-volume, fast-access units to large-volume, slow-access units. The large units store information at a lower unit cost, but usually at the price of increased access time. Files also vary in terms of the activity rate against them. It is generally desirable to keep frequently used files in a storage device that can deliver them quickly. Files that are used less frequently can be kept more economically in larger storage devices. The system is designed to keep the content of a storage device at some level slightly below its capacity. If this content builds up to the point where the device might "overflow", the system automatically scans the device "table of contents" for low-activity files. Such files are then moved out of the loaded device, and are rewritten into a larger, slower-access unit. Subsequent references to the same file will again cause it to be brought back to a faster-access device. Thus, the data files are dynamically adjusted to the proper class of storage.

User Commands

The user at a terminal can request that the system take certain actions in his behalf. Each action that he can specify is a command, which in the case of a typewriter-like terminal is a one-line message. Commands are always acted upon by the command-language interpreter. This system program receives the incoming message, scans it to determine the nature of the action requested, and fetches from the system files the command program that will carry out the action.

The command system (the set of actions that the command-language interpreter can recognize) provides the capability for a program to be:

- Entered from the terminal, line by line, with immediate diagnostic feedback from the system after each line
- Cataloged, stored permanently in the system for later manipulation, then made part of the user's prestored library
- Compiled or assembled
- Executed

To the system, the user is an active terminal and is presenting a series of tasks to be performed. As long as these tasks are system commands, the system is fetching its own programs and executing them on behalf of the user. In most cases, these command programs are in one part of the user's "virtual memory", operating upon the user's program (or data file), which is in another part of the "virtual memory". System commands are not very different from user programs; they are time-shared, multiprogrammed, and page-turned as they execute. When the user initiates execution of his own program he retains complete control over it. Thus the command system also allows a program to be:

- Stopped
- Modified or displayed -- for debugging purposes

- Checkpointed (this will create a new file for later use)
- Restarted
- Terminated

The program as entered may be in a variety of source languages. Initially the system will accept FORTRAN IV and an assembly language, and will accept PL/I when it becomes available.

Beyond these basic capabilities of entering, executing, and controlling programs from terminals, the command system provides for:

1. System accounting. A user must LOGON at a terminal before any other action can be taken, and LOGOFF when his terminal operations are complete. These commands initiate time charges to the user for use of the terminal itself and of other hardware facilities as required. They also serve to identify the user to the system. If, in succeeding commands, he requests access to files, his identity will be matched against the file-protection and security data.

2. File maintenance. A user can request that his files be moved from one storage unit to another, or be removed from the system. He can also modify them, combine them, or change their protection status.

Commands can be initiated from terminals, as previously described, or they can in most cases be embedded in programs as the equivalent of subroutine calls. They can also be included as "control card" functions to specify the type of processing to be applied to batch jobs entering the system. The command functions are, in essence, the verbs of a terminal control language. All instructions to the system are intercepted by the command language interpreter.

The command system has been discussed thus far in terms of its appearance to the remote computer user. This, of course, is its primary orientation. It allows the user to direct the system in its manipulation of files and the solution to problems. More than this, however, it is the universal interface between the system and the outside world. From the machine-room console, for example, installation personnel control the partitioning and recovery functions of the system, using special commands reserved only for their use. Batch jobs entering the system from peripheral equipment or tape drives are controlled by the command system. And finally, since commands are programs in the system files, the command system itself can be augmented or modified by terminal action.

This general concepts discussion has been presented from a remote-computing point of view. It is, however, a system capable of performing batch-processing operations introduced both locally and remotely -- as well as of providing immediate response to terminals. It is expected that any given installation workload will be a blend of batch-processing work (where response is not necessarily critical) and conversational computing (where response is often basic to the operation).

PROCESSING UNIT FEATURES

Program Switching

Switching between programs may be (1) initiated by the program (as when a problem program calls the supervisor or a supervisor returns control to a problem program), or (2) caused by a condition external to the currently operating program (as when the completion of an I/O operation is signaled by an interruption, or a timer interrupt).

Rapid program switching is facilitated by the program status word (PSW). This word contains all information required for proper execution of a given program. It contains the instruction address, condition code, mask and status bits, and other fields indicating the status of the program at the time of interruption. The current PSW is automatically stored during an interruption; thus the status of the processor is preserved for subsequent inspection and resumption.

Interruption and Priority

An interruption consists of the automatic storing of the current PSW and the fetching of a new PSW. Processing resumes in the state and at the location indicated by the new PSW. To permit proper program action following an interruption, the cause of the interruption is identified and provision is made to locate the last instruction executed under control of the old PSW.

The five classes of interruption conditions are:

Input/output
Program
Supervisor call
External
Machine check

The mask bits make it possible to assign relative priorities to the various interruption sources. The processor accepts interruptions only when they come from a specific interruption source, as determined by the mask. When masked off, an interruption either remains pending or is ignored, depending on the type of interrupt. Input/output and external interruptions may be kept pending by the system mask. Four of the standard 15 program interruptions may be masked off by the program mask. The machine-check interruption may be masked by the machine-check mask. If, for example, the external

interruption has a higher priority than an I/O interruption, the mask bits can be set up so that an external interruption will come in while an I/O interruption is being processed; an I/O interruption would not be allowed to come in while an external interruption is being processed. The mask bits may also be set to cause a reverse of priority between external and I/O interruptions.

Storage Protection

The basic design of the System/360 includes a flexible memory protection system. Complete memory protection is essential for time sharing.

For the time-sharing system, the storage protection feature is augmented by fetch protection and two indicators that indicate whether a block of storage has been referenced and whether the block has been changed. Three bits have been added to the storage key for this purpose. The protection keys, however, in the PSW and channel address word (CAW) remain unchanged.

Dynamic Relocation

To explain the operation of dynamic relocation, consider a specific instruction.

Within this instruction, the logical storage address is represented by the sum of the displacement, the index register, and the base register. Before the supervisor begins processing a task, the location of a table, called a segment table, is loaded into a special register called the table register.

The segment table contains an entry for each segment called for by the task. Each segment entry contains the location of a page table. The page table contains the physical address of the 4096-byte block (page) in which the data or instruction is located. There may be more than one page entry in the page table for any one entry in the segment table.

The page tables referred to by one segment table need not be contiguous; however, the page entries of each page table must be contiguous.

The supervisor loads the location of the particular task's segment table into the table register. The logical address is broken into segment, page, and byte portions. To call the proper element from the segment table, the segment portion of the logical address is added to the segment table origin. This entry, located in the segment table, contains a page table origin which, when combined with the page portion of the logical address, selects the proper entry in the page table. The page entry contains the core storage address of a 4096-byte block which, when combined with the byte portion of the logical address, produces the core storage address of the operand.

While the logical addresses would be sequential, the operands could spill over from one page to the next. Since a new page would be referred to, a different entry in the page table would produce a different core storage block address. Consequently, the use of randomly located blocks does not affect program operation.

To speed up the operation when constant reference to a specific page or group of pages is occurring, a group of eight associative registers is added. These registers are transparent to the program, and affect only the rate at which the program runs. A register is loaded with the logical and core storage page addresses each time a new page not already in the associative register is referred to. The registers may contain up to eight entries. A high-speed parallel search is made for the segment and page in the associative register. If the desired segment and page values are found in one of the associative registers, the core storage address portion is routed to replace that which would otherwise have been supplied by the page table.

If the logical page address were not found in an associative register, the original translation operation would be performed and an entry made in an associative register that had probably not been referenced recently.

There is another register that contains the instruction counter in relocated form. Sequential instructions are referenced using this register instruction by instruction. Should a page boundary be crossed, or a branch out of this page occur, the conversion of a logical address to a core storage address is performed and the new relocated address is entered into this register.

MULTIPROCESSING

The multiprocessing capability of the system provides reliability and continuity of operation and provides as well for growth in throughput capability. It is truly a polymorphic system that can ensure a single multiprocessing configuration or a symmetric multiprocessing relationship between two or more subsystems. The power, reliability, and flexibility of the system can be increased by expanding up to four computing units and up to 2 million bytes of actual core storage. It must be possible to interconnect the system components so that they may interact with one another. It must also be possible to partition a system, that is, to operate a selected processor plus storage and certain I/O devices as an autonomous unit.

Interconnection of System Components

The simplest interconnecting structure for a series of processing units and associated storage units is the crossbar switch arrangement shown in Figure 2.

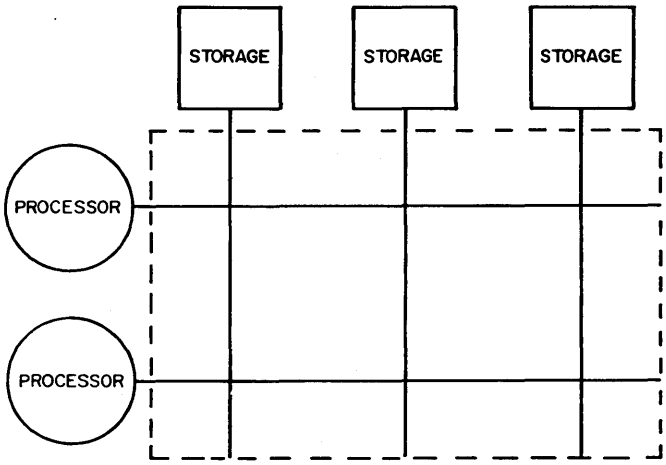


Figure 2. Crossbar interconnections of system components

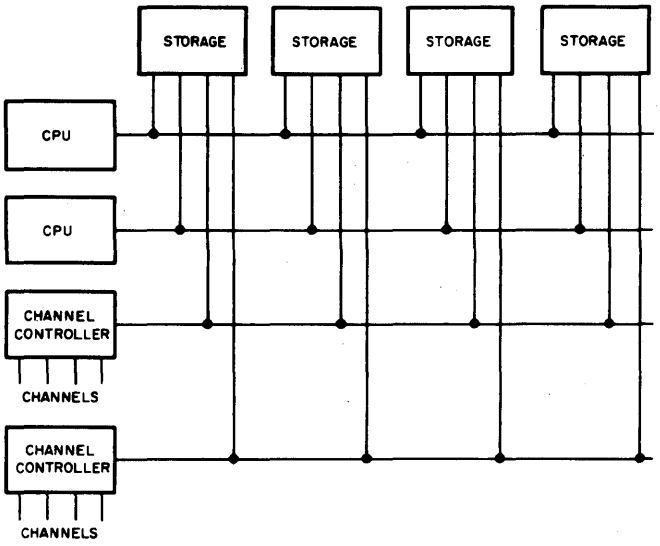


Figure 3. Distributed-switching interconnection of system components

For the time-sharing system, however, the switching arrangement is not a centralized switching point, but has been distributed among the various components that form the system (see Figure 3).

The reasons for adopting the distributed approach are improved availability and flexibility. A centralized piece of equipment represents a crucial link within the system, since its failure would cause the entire system to fail; it would be more costly to remedy this by duplicating equipment. The distributed approach is more flexible and facilitates extension of the system by addition of more processors or storage units.

The interconnecting structure shown in Figure 3 shows that the necessary switching equipment is distributed among the system components, central processor, channel controller, and core storage units. Drivers are added to the processors as required, and the equipment needed for selection of any of the processor bus systems is added to the core storage units. Instead of the single bus connection or tail of the simplex system, multiple tails are provided for a multiprocessing system. The design of the circuits that select from among the tails is such that if a storage unit should fail (including power failure), the other storage units connected to the storage bus would remain operative. When the bus control unit within a processor fails, the entire bus driven by this unit is, of course, inoperative. This bus will not, however, prevent proper functioning of the storage units with the remaining processors.

I/O channels within a multiprocessor system are flexible in the System/360 design. The channels perform the transmission function. All controlling functions for I/O devices have been placed within the control units for these devices. Thus, the channel design is general and applies to all I/O devices.

As with the processors and core storage units, a control unit can be attached to more than one channel by means of multiple tails. The switching equipment is modular, a design that avoids the problems of cost and poor reliability of centralized switches.

A channel operates concurrently with the processing unit and may be treated as an independent entity within the system. As shown in Figure 3, the channels are grouped into two sets, each provided with its own channel controller; thus there are two systems for I/O operations. Storage units are equipped with the necessary tails to accommodate the additional buses. Channels are addressable by either processor and can return their interruptions to either unit.

Sharing of external storage media can be achieved by interconnecting their channels, their control units, or the devices themselves. Special attention has been given to availability considerations, with minimal duplication of equipment.

System Partitioning

A system with sufficient elements may, in effect, be divided or partitioned into at least two systems that can operate independently as single systems. This type of partitioning can be achieved by relying on the programs in each system to refer only to those components that have been assigned for their use. This approach can be enforced by a program that supervises storage assignment and protection and assigns I/O devices. System/360 was designed for operation with such a supervisory program. Where it is desirable to experiment with new and as

yet undebugged supervisors, however, a more absolute means of partitioning may be required. Such a means is provided in the Model 67 by manual partitioning switches.

Partitioning is detected dynamically by sampling a single wire -- the availability signal. When the signal is up, the unit is available to the processor; when the signal is down, the unit is unavailable.

An availability signal is provided for each processor bus system from each storage unit within the multiprocessing system. Absolute partitioning is achieved by making these lines manually switchable. One storage array, therefore, may be available to one part of the system and unavailable to another part of the system. An invalid address indication results if the system attempts to address an array designated as not available.

Partitioning of the control units and channels is similar. The control units can be selected from the multiple interface tails; this selection can be overridden by manual switches.

Partitioning, therefore, makes it possible to operate the processors within the multisystem as single processors, as independent processors sharing common storage facilities, and as part of a multiprocessing system.

Figure 4 shows a representative multiprocessor configuration. Each circled X denotes a partition

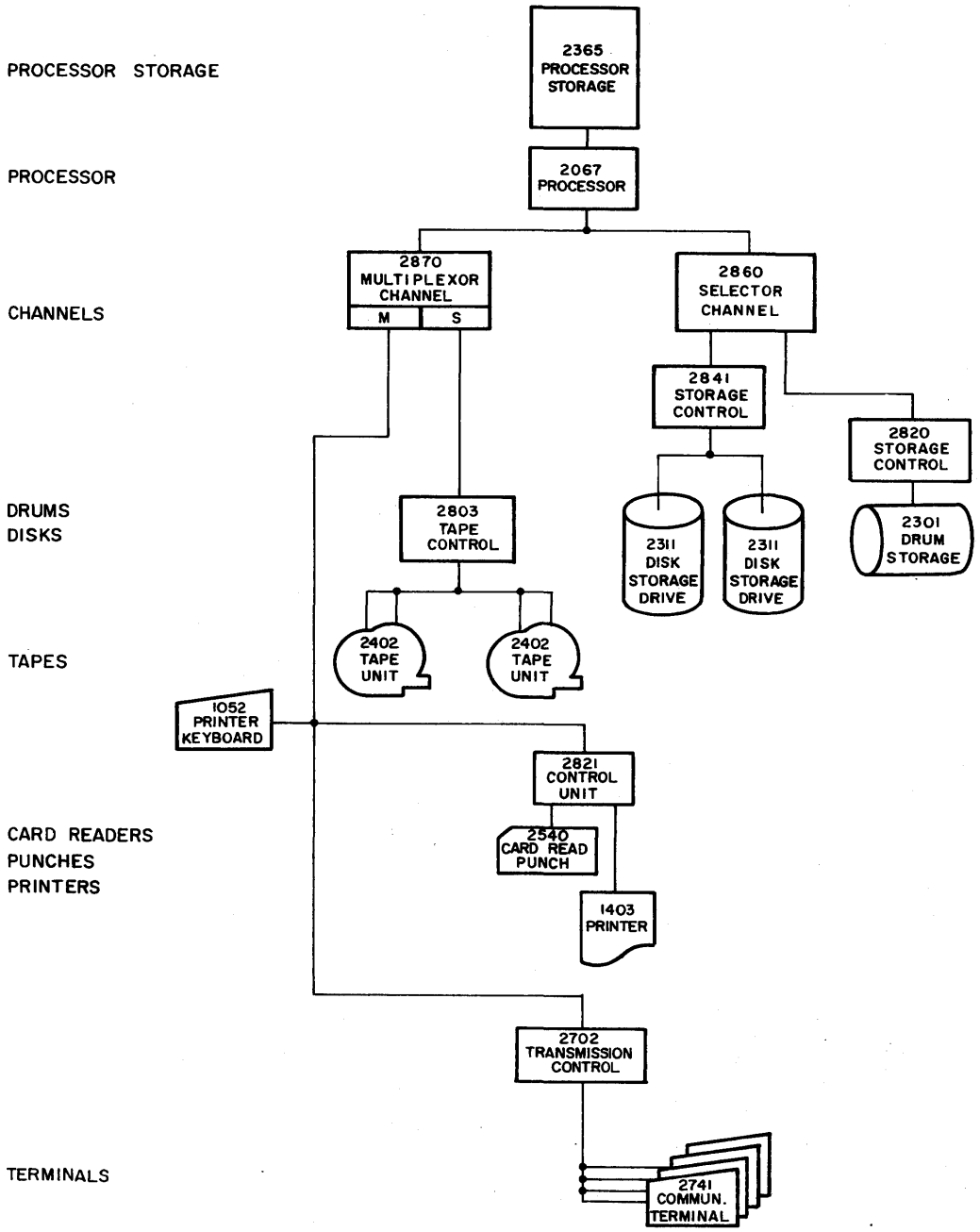


Figure 4A. Small, single-processor IBM System/360 time-sharing system

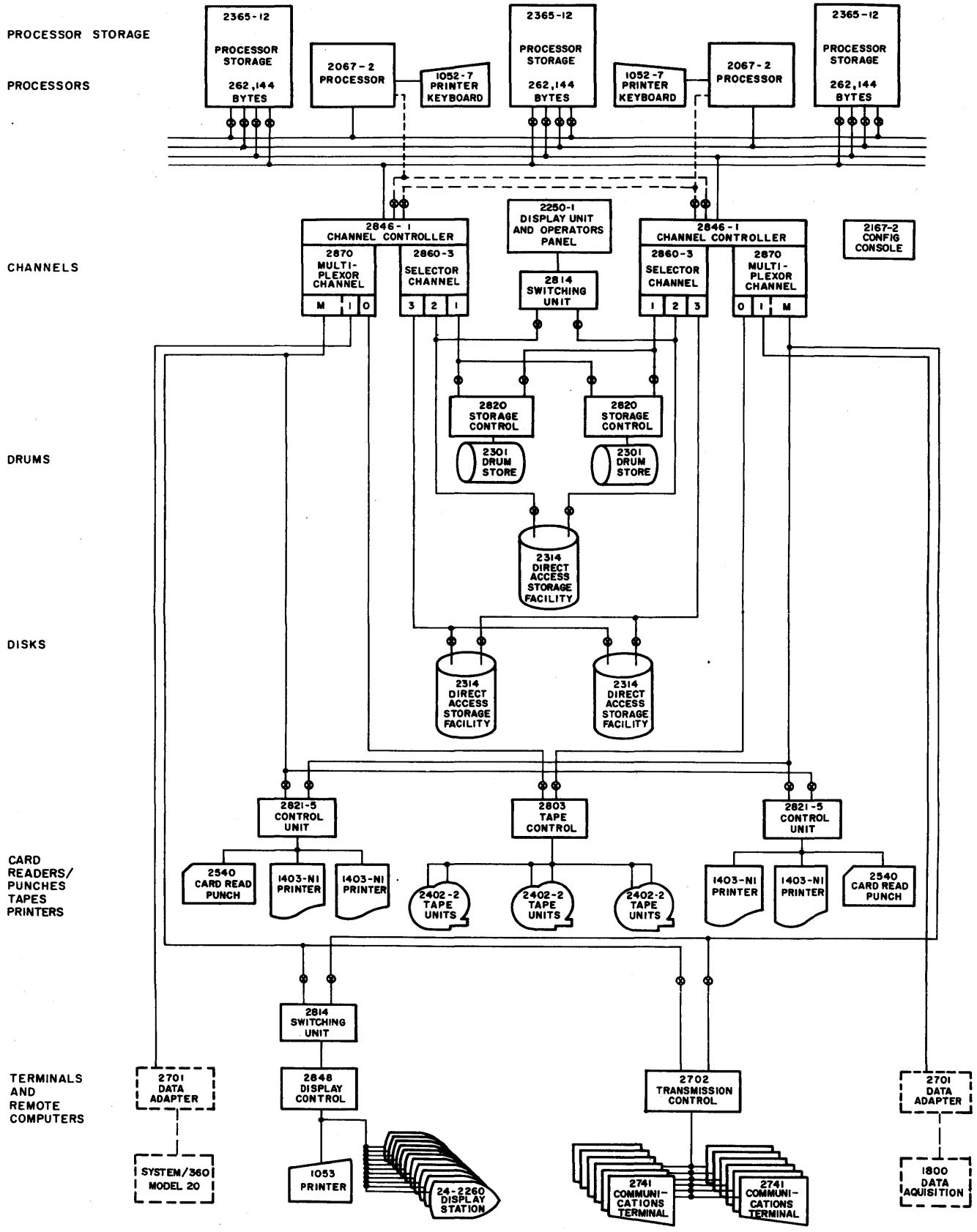


Figure 4B. Typical IBM system/360 model 67 dual-processor, time-sharing system with manual partitioning switches

point. Each storage unit can be assigned to all processors and channel controllers in any desired configuration.

Each channel controller may be controlled by any processor in the system via buses connecting the two. A partitioning switch either permits this control or causes the channel controller to ignore any attempt by this CPU to initiate an I/O function. When the processor-to-channel controller path is inactive, commands to the channels and devices attached to that channel controller are not executed and cause condition code 3 to be set in the PSW. This indicates to the processor that the channel is not operational.

The control units for the high-speed drum and disk files can have two interface connections (tails) each, thus permitting each control unit to be physically attached to two channels. The logical connection between the control units and the channels is under program control, thus providing the program with the facility to connect the control unit to any of its channels. Once a connection is established, it is preserved until the control unit is released by a command from the connected channel. Release causes the control unit to assume the neutral state in which it is available to any channel.

In the IBM 2702 Transmission Control Unit, which provides for attachment of communication lines to the multiplexor channel, the switch connecting the control unit with either of the channels is placed in the neutral position upon resetting the control unit. The first programmed selection of a communication line subsequently causes the control unit and all associated communication lines to be switched to the selecting channel. This connection is maintained until the control unit is reset or a release function is performed.

When a control unit has been disconnected from a channel by the partitioning switch, that channel does not have access to the control unit and all devices on the control unit appear to the channel to be not operational. The not-operational state of a control unit or device is indicated by setting condition code 3 in the PSW. To restore switching under program control, the control unit must be reconnected to the system by the partitioning switch.

The condition of all partitioning switches can be sensed by the Store Multiple Control instruction.

Aids to Intrasystem Communication

Communication between the processors of a multiprocessing system is supplemented with means for:

- Signaling that a message has been or is to be transmitted
- Interlocking the use of storage to prevent conflict
- Requesting intervention in case of malfunction
- Permitting initialization to attempt recovery from malfunction

Signaling

For communications through a common storage facility, a processor must be alerted when a message has been prepared for it by another processor. The extended direct-control feature and external interrupt lines of the Model 67 perform this function.

Associated with the direct-control instructions is an interface at which eight signals are made available. A signal from one processor is connected to one of the external interruption lines of another processor. By means of the Read Direct or Write Direct instruction, the program in one processor causes an external interruption in another processor.

Interlocking

When shared storage is used as a common medium for data, restart information, programs, or results that are updated by different processors, interlocks must be provided to prevent mutual interference. The interlocks are provided within the device when I/O devices are the common storage media. The operating nature of devices that can maintain only one data transfer operation to or from the storage medium at a time, provides the desired interlock.

A longer interlocking period is required during the time between the physical arm movement in disk files and the read or write command. Additional controls can therefore be supplied so that a program can reserve an access mechanism.

The requirement for interlocking is apparent in cases where each processor simultaneously attempts to update a single record. One processor could read the record to perform the updating and the other processor could immediately attempt to read the same record. Processor 1 performs the updating and writes back the record, and processor 2 updates and writes back the record. Without interlocking, a complete transaction could have been lost. With interlocking, processor 1 would have reserved the file during an update and then released it, at which time processor 2 could have performed its updating without any chance of interference.

When main storage is the means of communication, electrical interlocks are provided for the period of one storage cycle. When two processors simultaneously request access to storage, a tie-breaking priority circuit grants access to one processor, then gives the next cycle to the other processor. This simple rule prevents one processor from locking out and therefore effectively halting

another processor. I/O is always granted higher priority than a processor.

Access to storage is granted for the duration of one storage cycle, not for the duration of an entire instruction. Since, as a rule, information is processed in internal registers, a typical procedure is to fetch data from storage, process the data in registers, and place the results in storage. The storage-interlocking mechanism, unaware of the relationship between successive storage accesses, makes it possible for one processor to store results in a location after the second processor has fetched its operands from that location, but before the second processor stores the results. This possibility necessitates a programmed interlock in the use of core storage as a shared medium. This requirement becomes even stronger when one processor is to use an entire storage area before permitting a second processor to use it.

The Test and Set instruction is designed specifically for this requirement. When this instruction is performed, a bit is tested and subsequently changed to 1 without access to the particular bit by any other processor in the intervening period; the instruction is available to the problem program and supervisor program alike. By means of the protection feature, testing and setting of particular control bits can be reserved for the supervisor program. A concise description of the Test and Set instruction is included under "System Components".

Malfunction Alert

In a multiprocessing system designed for high availability, the system should be alerted when a malfunction occurs in one of the components. A malfunction signal, similar to the direct control signal, is issued as soon as a machine malfunction is detected. This signal may be transmitted to other processors in the system, using the external interruption inputs of those processors.

The extensive checking included in all System/360 equipment is useful not only in error detection but in the improvement of fault location. A high degree of checking makes it possible to recognize malfunctions on short notice and thus preserve the state of the CPU for later diagnosis. Furthermore, the detailed error information made available to the customer engineer reduces the repair time and contributes to the overall system availability.

Programmed Initialization

Each IBM System/360 processor uses permanently assigned storage locations (0-127) for program status words, channel address and status words, the timer, and initial program loading. If these locations were common, they would be shared by several processors and interference among processors would result. To provide each processor with separate assigned storage, a quantity called a prefix is used for dynamic relocation of all addresses referring to the first 4096 storage locations. In a multiprocessor system each processor is normally assigned a different prefix, and the sharing of these preferred locations is therefore avoided.

The prefix relocates all locations that can be directly addressed (using zero-index specifications) by the displacement. During program switching, such direct addressing is necessary when the supervisor must store the general purpose registers. The prefix makes this programming technique possible even if locations 0-4095 are not available to the system.

When a malfunction occurs only in storage, a system can resume operation immediately by eliminating the faulty storage unit. If the faulty storage contains the permanently assigned storage locations for the processor, new locations can be provided by introducing an alternate prefix. For this reason, a second prefix quantity is provided for a CPU as part of the System/360 multisystem feature.

Normally, the two prefix quantities relocate the preferred storage locations to different storage units; the processor therefore becomes independent of the specific storage unit for its operation.

The identity of the processor executing a program may be determined by setting apart one of the addresses in the range 0-4095 as the address of an identifying location, and loading identifying quantities in each of the corresponding physical locations. Since the physical location actually selected is determined by the prefix number, inspection of the addressed location identifies the processor and prefix currently used.

When a processor is reintroduced into a multiprocessor system, it is desirable to minimize operator action. Introduction of a new program status word and the corresponding instructions may best be performed by the still-operating part of the multiprocessor system. For this reason, means are provided for one processor to start another processor. This signaling again has been defined consistent with the signals of the direct-control circuits. Two signal inputs are provided, each of which causes an action similar to initial program loading. The choice between the two signals determines which prefix is used, and hence, the location of the permanently assigned storage addresses. In this case, the external start consists of loading an initial PSW from location 0 and performing the necessary system reset. Before the external start,

the necessary PSW's should have been established by the active part of the system.

SYSTEMS DATA FLOW

General Information

Data is transmitted throughout the system along paths whose number, width, and data-rate capabilities are designed for efficient information flow and minimum interference.

In general, paths are widest where data rates are highest. The number of paths between major elements of the system is a function of the requirement for simultaneous operation and availability.

Data is transmitted eight bytes at a time between main storage and the processors and channels. It is transmitted one byte at a time between channels and I/O control units. Parity checking throughout the system is at the byte level.

Storage Bus

Each processor and each channel controller has a separate bus connecting it to each storage unit in the system. A system having two processors and two channel controllers has four buses. Expansion facilities provide for a total of eight buses, divided among processors and channel controllers. Each bus is connectable to as many as eight storage units.

Conflicts that occur among the several buses connected to each storage unit are resolved at the storage unit. This conflict resolution adds 150 nanoseconds to storage access time. Channel controller requests for storage cycles are given priority over processor requests.

Each processor has a bus control unit (BCU) that controls the storage bus associated with it. The processor can place data in core storage selectively by bytes. To initiate a store-type reference, the processor requests a storage cycle by providing to the BCU the address of a doubleword, accompanied by one or more mark signals. There is one mark line for each of the eight bytes that can be stored in a single cycle, and presence of signals on these lines indicates which bytes are to be replaced in the storage unit. Bytes that are not replaced remain unchanged. When the BCU indicates that the addressed storage unit has been selected and is available, data is provided to the storage unit.

When the processor requests a storage cycle without any signals on the mark lines, a fetch operation is initiated. The IBM 2067 processor expects data in a fetch operation to be available a fixed time after the initiation of the request. If the storage unit does not provide the data at the anticipated instant, operation of the processor is inhibited until data is available. During fetching, storage always provides full doublewords of data.

When a storage unit is shared by multiple processors or among processors and channel controllers, multiple concurrent references to the storage unit are possible. As far as the BCU and the channel controller are concerned, execution of storage cycles is the same as in a nonshared storage unit, except that the time interval between the initiation of a storage request and the acceptance of the request by the storage unit is variable. Whenever the storage unit is busy with the execution of a request from another bus, the new request remains pending on its bus until accepted by the storage unit. The storage units execute references in a fixed priority, established among those references that are pending at the instant of sampling. When multiple requests are pending, establishment of the priority is overlapped with the execution of the preceding cycle.

In the data transfer function, the channel controller performs two types of tasks: it communicates with the storage units and it serves the requests from its channels.

Channel servicing consists of establishing priority among the channel requests and initiating storage references in response to these requests. Channel priority is established by a priority network on the basis of requests from the channels. When more than one request signal is present at time of sampling, the channel controller establishes priority among the requests in the order of ascending channel addresses. Establishment of priority is overlapped with the execution of the preceding storage cycle to make maximum use of the storage bus.

When priority has been established for a particular channel, the channel controller requests the storage address and mark signals from the channel and initiates a storage cycle. Communication between the channel controller and the storage units is identical to that between the BCU and the storage units.

Channel Data Path

The channel data path provides for the transfer of data between the channel controller and the attached channels. This path is eight bytes wide, including the associated parity bits. All data transfer between the channel controller and the attached channels takes place over a single path; the time sharing of the path is under the control of the channel controller. The definition of signals on this path, other than those required for establishing priority among channels, is the same as for signals on the storage bus.

Input/Output Interface Data Path

All communication between channels and I/O control units is via I/O interface data buses (one in each direction), which are one byte wide (eight data bits and one parity bit). The connection between channels and control units is standardized. Further information is given in IBM System/360 I/O Interface -- Channel to Control Unit OEMI (A22-6843).

Data Path Bandwidth

A channel controller can achieve a maximum data rate of 800,000 doublewords or 6,400,000 bytes per second.

Two types of channels can be attached to a channel controller: the IBM 2860 Selector Channel and the IBM 2870 Multiplexor Channel. The 2860 Selector Channel can sustain data rates of 1,300,000 bytes per second. The basic interface of the IBM 2870 Multiplexor Channel can handle an aggregate rate of 110,000 bytes per second, when no medium-speed interfaces are provided. Table 1 illustrates the data rates that can be sustained by the basic interface and the selector subchannels of the IBM 2870 Multiplexor Channel:

Table 1: Selector Subchannels

Basic	1st	2nd	3rd	4th
110	x	x	x	x
88	180	x	x	x
66	180	180	x	x
44	180	180	180	x
30	180	180	180	100

Data rates are in kilobytes per second.

Nonstandard Component Interfaces

IBM System/360 is designed to permit attachment of special as well as non-IBM components. This versatility is achieved by providing well defined interfaces for certain components of the system. Many non-IBM components, such as processors, core storage units, control units, or I/O devices, may be incorporated in the system by designing to the interface specifications.

Detailed descriptions of these interfaces will be published as Original Equipment Manufacturers Information (OEMI) manuals. These descriptions will be made available upon request.

SYSTEM COMPONENTS

The IBM System/360 is a versatile, all-purpose system that can accommodate all applications which may be encountered in a diversified computing activity. The system is unique in its capability to grow easily with increasing needs for computational capability. If the particular I/O equipment is outgrown, more or faster I/O can be added. Core storage can also easily grow in capacity. The important point, however, is that any or all of this growth can be accomplished with no changes in system programming or in problem programming.

This section briefly outlines most of the components in the System/360. Specific emphasis has been placed on those devices and features which are unique to the time-sharing, multiaccess, and multiprocessing aspects of the system.

IBM 2067 PROCESSING UNIT

General Information

The 2067 Processing Unit provides the necessary arithmetic, logical, and control functions for the System/360.

The processor contains the facilities for addressing core storage, for fetching and storing information across the storage bus to which it is connected, for arithmetic and logical processing of data, for instruction sequencing, and for initiating the communication between core storage and external devices.

Information is transmitted between the processor and core storage across a storage bus 72 bits wide. These 72 bits constitute a doubleword of eight 8-bit bytes plus eight parity bits. Byte manipulations are possible, and any number of bytes up to a maximum of eight can be transferred in a single cycle.

The processor operates with a basic internal cycle time of 200 nanoseconds. A 60-bit parallel adder handles the full-length fraction in floating-point operations. An eight-bit adder handles simultaneous exponent arithmetic in floating-point operations, and is also used in serial variable-field-length decimal arithmetic.

The processor contains 16 general registers for fixed-point binary arithmetic, address manipulations, and indexing purposes. Each has a capacity of 32 bits (one word) and may be addressed by four bits in the instruction. For certain operations, pairs of registers may be coupled to form single 64-bit registers. Four additional registers each contain a 64 data-bit doubleword. They can be used for both short (one word) or long (doubleword) floating-point arithmetic.

A capacitor read-only storage (ROS) etched on a glass substrate provides logical control for the processor. The flexibility of ROS control permits incorporation of specialized instructions or functions into the processors. The readout time of this storage is 140 nanoseconds. Sixteen planes, of 176 words each, provide an ROS capacity of 2816 hundred-bit words.

In addition to the above, the processor incorporates several features that provide dynamic address translation, protection, extended I/O control capability, communication between multiple processors, a high-resolution interval timer, and facilities for the programmed inspection of partitioning switches.

Dynamic Relocation

By its very nature, time sharing requires extremely large storage capacity. Not all information needs to be immediately available in core storage, however, since all programs and data sets are not simultaneously active. When programs or data sets or portions thereof become active, they must be quickly moved into core storage so that delays seen by the users are minimized. The dynamic relocation feature on the Model 2067 provides a means of doing this quickly and efficiently.

The dynamic relocation feature permits the use of a "virtual store" concept, whereby the total memory addressed is far larger than the actual core storage. The virtual addresses are translated to actual addresses by relocation tables that are set up and changed on a continuing basis by the supervisory program as information is moved back and forth between core storage and drums or other slower, larger-capacity storage devices.

This moving of information is done by the control program to satisfy user demands. The unit of information moved is 4096 bytes, commonly referred to as a page of information -- hence the term "page turning" to refer to this process of dynamically relocating information.

A set of eight associative registers is provided to minimize the effect that repeated references to relocation tables in core storage would have on processor performance. These registers are called associative registers since, in a translation process, a register is referenced by its contents rather than its physical location.

One other feature provided to minimize delays due to address translation is storage of the instruction address (that is, the instruction counter) in the translated or relocated form. This feature obviates

the need for instruction address translation until a branch occurs or a page boundary is crossed.

The 2067 processor uses 24-bit addressing for the time-sharing system; in this mode of operation, a program may have up to 16 segments of 256 pages each. TSS also permits an extension of this addressing scheme to 32 bits, allowing the use of up to 4096 segments of 256 pages each. The dynamic relocation extension feature permits both 24- and 32-bit addressing.

The processor recognizes the relocation mode only when it has previously been switched into the extended control mode (see "Extended Control" later in this section). In the extended control mode, bit 4 of the PSW specifies the extent of logical addressing. That is, when bit 4 is a zero, 24-bit logical addressing takes place, and when bit 4 is a one, 32-bit logical addressing takes place. Bit 5 of the PSW (in extended control mode) specifies relocation vs nonrelocation mode. When bit 5 is a one, relocation takes place, and when bit 5 is a zero, no relocation takes place. When bit 4 is a one, bit 5 must be a one; otherwise, a specification exception is recognized.

The following table summarizes the modes of operation specified by bits 4-5 of the PSW (in extended control mode):

Bit 4	Bit 5	Mode of Operation
0	0	No relocation, 24-bit address arithmetic
0	1	Relocation, 24-bit address arithmetic
1	0	Specification exception
1	1	Relocation, 32-bit address arithmetic (specification exception if 32-bit addressing option is not provided)

Address Translation (24-Bit Version)

The relocation tables used to translate a logical address into an actual address consist of "segment" tables and "page" tables. These tables are placed in main storage at the "segment table origin" and "page table origin" respectively. Each table occupies the number of storage locations specified by the respective "table length" amount.

Segment table origin is specified by the contents of bit positions 8-31 of the "table register" (control register 0). The length is always 64 bytes or one group of entries (16 entries per group, four bytes per entry). The address of the table origin must be a multiple of 64; hence, bits 26-31 of the table register must be zeros or a data exception is recognized.

Each four-byte entry in the segment table defines a page table. The first byte (bits 0-7) defines the length of the page table, and the remaining three bytes (bits 8-31) define the page table origin. The unit of length for a page table is a two-byte entry. Thus, the table is variable in multiples of two bytes. Each page table's origin is located at a byte address that is a multiple of two. Thus, bit 31 of each segment table entry that defines a page table is zero. If bit 31 is one, no translation takes place and a segment relocation exception is recognized (program interruption with interruption code 16).

Figure 5 illustrates the translation action when the 24-bit addressing mode is used. Bits 8-19 of the "logical" (or virtual) address are first compared with the corresponding bits of each associative register having bit 36 set to one. If a match is found, bits 20-31 of that register are used as bits 8-19 of the actual address and bit 37 at the associative register is set to one. Bits 20-31 of the logical address are used directly as bits 20-31 of the actual address.

If no match is found, or if no register in the associative array has bit 36 set to one, the logical address must be translated by means of the segment and page tables. This translation proceeds as indicated by the dotted lines in Figure 5. The segment field of the logical address (bits 8-11) is first added to the origin address portion of the table register (bits 8-31). (For this addition, the segment field of the logical address is aligned with bits 26-29 of the table register since the entry to be fetched is four bytes long and has a byte address that is a multiple of 5.) The quantity obtained by this addition is the address of the segment table entry.

The segment table entry is used with the page field of the logical address in much the same manner as the table register contents were used with the segment field (see Figure 5.) Bits 12-19 of the logical address are aligned with bits 23-30 of the origin portion of the segment table entry (bits 8-31), and the two quantities are added. The resultant 24-bit quantity is used as the address of a two-byte page table entry that is subsequently fetched from storage. As described earlier, if bit 31 of the segment table entry is one, a segment relocation exception is recognized. In addition, bits 12-19 of the logical address are compared with the page table length (bits 0-7 of the segment table entry), and, if the former is greater than the latter, a page relocation exception is recognized (program interruption with interruption code 17).

The two-byte table entry consists of a physical page address portion (bits 0-11) and control bits (bits 12-15). Bits 13-15 must be zeros or a specification exception is recognized and the instruction

is suppressed. Bit 12 defines the status of the page address portion of the entry. If bit 12 is zero, the page address is used as bits 8-19 of the physical address as shown in Figure 5. If bit 12 is one, a page relocation exception is recognized. Bit 12 thus serves to indicate whether the page referenced is actually available in core storage. The other three bits (bits 13-15) are reserved for future use.

The page address obtained by the translation method described above is not only used to address memory but is also loaded into an associative register along with the segment and page fields of the logical address. Thus, it is made available for future use without the need for repeating the translation process. When an associative register is so loaded, its bit positions 36 and 37 are set to one. (Selection of the registers to be loaded is under control of a usage algorithm which uses in sequence the registers with bit 37 set to zero. When bit 37 is one in all registers, this bit is reset to zero in each register.) Bit 36 is used to indicate the presence of a valid entry in the associative register. It is reset to zero each time the contents of the table register are changed.

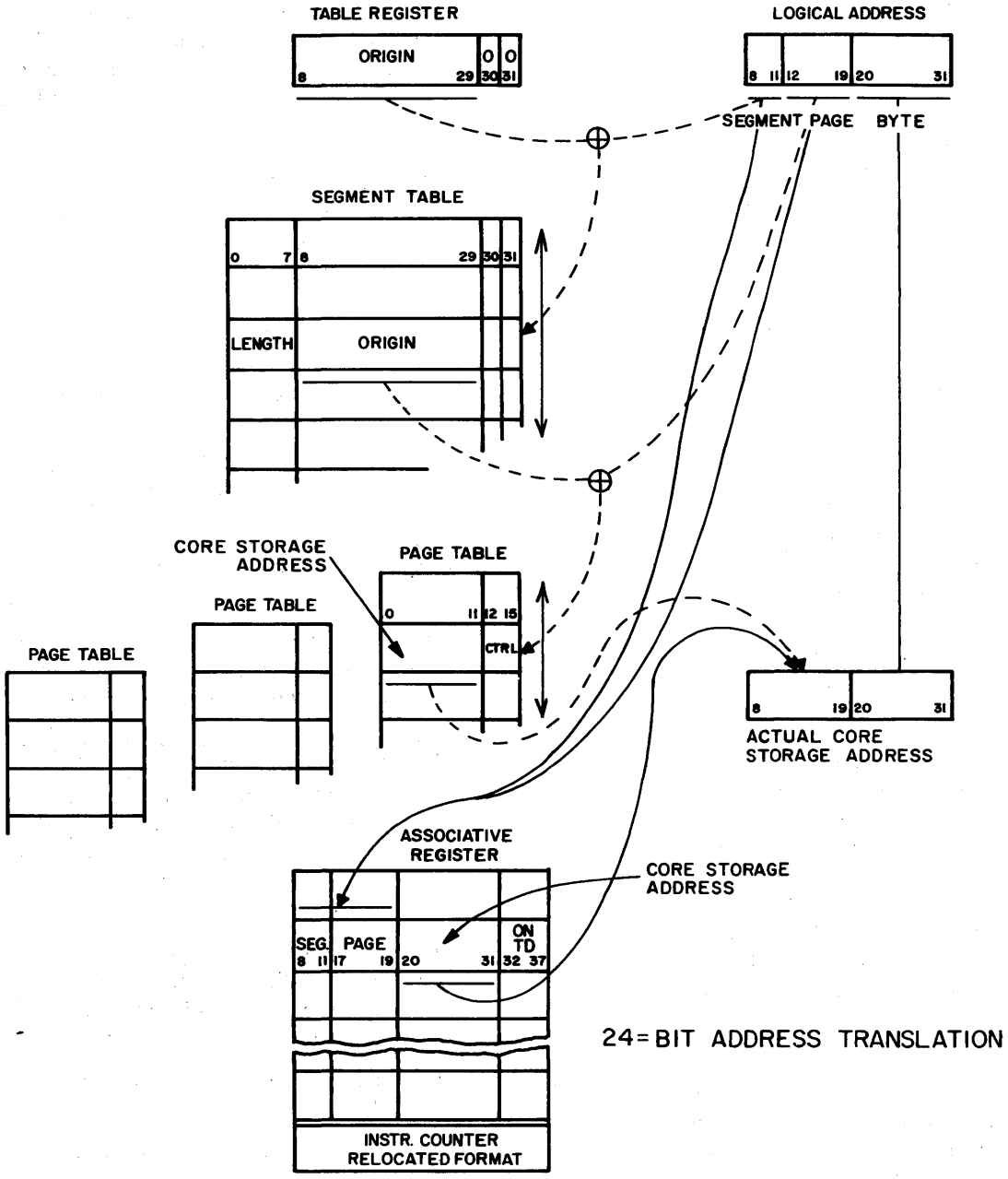


Figure 5. Simplified data flow for dynamic relocation

Relocation Mode

Relocation of addresses provided by the processor is specified by bit 5 of the PSW (in extended control mode). When the bit is a one, relocation takes place; when the bit is a zero, the logical address is used as the actual core storage address.

All main-storage locations where information is stored in the course of an operation are subject to relocation.

Addresses provided by the channels, either for fetching channel control words from main storage or for fetching data from (or storing data into) main storage, are never relocated regardless of the setting of bit 5 in the PSW.

Locations whose addresses are generated by the processor or channels for updating or interruption purposes (equipment generated addresses), such as the timer, channel status words, or PSW addresses, are not relocated via the relocation tables. However, when the program specifies these locations, they are subject to relocation as defined above.

Actual core storage addresses in the range 0-4095 (including the above-mentioned equipment-generated addresses) are relocated by means of the primary or alternate prefix, as defined in System/360 Principles of Operation (A22-6821), unless the prefix is disabled by means of the prefix deactivation switch. Consequently, the prefix is applied when the address (either the logical address when no relocation takes place or the translated address obtained via the relocation tables) falls within the range 0-4095.

When relocation is specified, the storage protection, by means of the protection keys, is still active.

Whenever access to main storage is made by the equipment for the purpose of fetching an entry from a relocation table in the course of an address translation process, storage protection is ignored; that is, the equipment acts as if the block of storage containing the relocation tables was not fetch-protected during the memory cycle in which the relocation table entry is fetched. However, if the addresses at which the relocation tables are located are generated by the program, they are subject to storage and fetch protection in the normal manner.

If the storage address, generated in the address translation process for fetching a relocation table entry, exceeds the storage capacity of the installation, an addressing exception is recognized, resulting in a program interruption (interruption code 5).

Extended Control (24-Bit Version)

PSW Format

The system can operate under the control of a PSW in the following two modes:

- Standard PSW format as defined in System/360 Principles of Operation (A22-6821)
- Extended control PSW format, as defined in this section

Switching between these two modes is under control of bit 8 of control register 6. When this bit is one, extended control PSW format is used. Upon system reset resulting from a power-on sequence, manual system reset, manual IPL, or external start (electronic IPL), this bit is set to zero. If it is changed by a Load Multiple Control instruction, the resulting PSW mode change becomes effective upon completion of that instruction. (If other bits are changed by the Load Multiple Control instruction, the exact moment when the new settings take effect is unpredictable.)

The following are bit assignments of the PSW format for extended control:

Bits	
0-4	Must be zeros. Otherwise, a specification exception is recognized whenever the PSW containing a nonzero bit in these positions is used (that is, during the fetching of the first instruction under the control of this PSW, similar to the specification exception caused by a nonzero bit in position 63 of the PSW).
5	Relocation mode bit. When this bit is a zero, no relocation takes place; when it is a one, relocation takes place.
6	I/O mask bit, controls the masking of all I/O channels. When this bit is a zero, all I/O channels are masked off; when it is a one, the masking of the individual channels is contained in control register 4-5.
7	External mask bit. When this bit is zero, all extended interruptions are masked off; when it is one, masking of external interruptions is controlled by control register 6.
8-11	Protection key (same as standard PSW format)
12-15	AMWP (same as standard PSW format)
16-17	Instruction length code
18-19	Condition code

20-23	Program mask
24-29	Must be zeros. Otherwise, a specification exception is recognized.
40-63	Logical instruction address

In addition to storing the above fields of the PSW as an "old PSW", the 16-bit interruption code field generated as a result of an interruption is stored as a halfword in storage in the following byte location.

Interruption Type	Storage byte location
External	14-15
SVC	16-17
Program	18-19
Machine check	20-21
I/O	22-23

Control Registers

A set of control register positions is provided as part of various features. Up to 16 registers of 32-bit positions each may be provided. The bit position assignments for the control registers are shown below. Some of the bit positions are assigned for the purpose of sensing the settings of manual switches, and are therefore not implemented as registers. These bit positions can only be stored into main storage, but cannot be loaded from main storage. The control registers are not part of addressable storage. They are changed by Load Multiple Control and inspected by Store Multiple Control instructions.

Bit positions of the control registers are assigned as follows.

Control Register	
0	Table register (for dynamic relocation)
1	Unassigned
2	Relocation exception address register
3	Unassigned
4-5	Extended mask registers. The bit assignment is as follows: Bit 0-63: I/O channel mask for channels 0-63
6	Bits 0-3: machine check mask extensions for channel controllers. Bit 8: extended control mode. Bit 9: configuration control. Bits 24-31: external interruption masking as defined in the following table.

Interruption Source	Bit Position*
Timer	24
Interrupt key	25
External signal 2	26
External signal 3	27
External signal 4	28
External signal 5	29
External signal 6	30
External signal 7	31

*Bit position applies to:
(1) PSW for interruption code
(2) Control register 6 for masking

7	Unassigned
8-9	States of core storage partitioning switches, one eight-bit byte for each logical processor storage unit. The bits in the byte correspond to the eight tails of the logical processor storage units, with "one" indicating that connection is established over the tail.
10	Bits 0-31: Core storage address assignment, one four-bit field for each of the maximum of eight logical processor storage units. The four-bit field contains bits 11-14 of the assigned core storage address.
11	Bits 0-15: States of channel controller partitioning switches with one four-bit field for each channel controller. The bits in the field correspond to the four tails of each channel controller, with "one" indicating that connection is established. Bits 16-31: Channel address assignment (as viewed from the processor executing the STMC instruction), one four-bit field for each of the maximum of four processors. A field containing three zeros and a one indicates that for the particular processor, only the channel controller corresponding to the bit position that is "1" is addressable and its channels are 0-6. No other bit combinations are possible in these four-bit fields.
12-13	States of control-unit partitioning switches, with at least two bit positions assigned to each control unit. "One" indicates that connection is established. The particular assignment of bit positions is presently left open.
14	Bits 0-23: Unassigned

Bits 24-27: States of direct-control partitioning switches, one bit for each processor. "One" indicates that the direct-control interface of the corresponding processor is connected to the other CPU's; zero indicates that the direct-control interface is disconnected from the other processors.

Bits 28-31: States of prefix deactivation switches, one bit for each processor.

15 Unassigned

New Instructions (24-Bit Version)

The following new instructions are included in the instruction set of the Model 67: Branch and Store, Load Real Address, Load Multiple Control, and Store Multiple Control; discussion of these new instructions follows.

	Mnemonic	Type	Code
BRANCH AND STORE	BASR	RR	OD
BRANCH AND STORE	BAS	RX	4D

The updated logical instruction address, consisting of the rightmost 24 bits of the PSW, are stored as link information in the general register specified by R_1 (in bit positions 8 to 21). Zeros are stored in bit positions 0-7 of the general register specified by R_1. Subsequently, the logical instruction address is replaced by the logical branch address.

Condition code: The code remains unchanged.

Program interruptions: None.

Programming note: The link information is stored without branching when in the RR format and the R_2 field contains zeros.

When Branch and Store is the subject instruction of Execute, the instruction length code is 2.

	Mnemonic	Type	Exceptions	Code
LOAD REAL ADDRESS	LRA	RX	M, S	B1

The translated address of the second operand is inserted in the low-order 24 bits of the general register specified by the R_1 field. The remaining bits of the general register are made zero.

The address specified by the X_2, B_2, and D_2 fields is translated through the dynamic relocation feature (regardless of whether the relocation mode bit is a zero or a one), and the translated address is inserted in bits 8-31 of the general register specified by R_1. Bits 0-7 are set to zero. The translated address is not inspected for protection or resolution.

During the address translation process, no relocation exceptions are recognized. Instead, the condition code is used to indicate successful translation or the reason for its failure. If the translation was unsuccessful, the contents of R_1 are replaced by the table entry, leaving its availability bit set to one; and the logical address that was to be translated is stored in control register 2.

	Mnemonic	Type	Exceptions	Code
LOAD MULTIPLE CONTROL	LMC	RS	M, A, S, D	B8

The set of control registers starting with the control register specified by R_3 is loaded from the locations designated by the second operand address.

The storage area from which the contents of the control registers are obtained starts at the location designated by the second operand address and continues through as many storage words as needed. The control registers are loaded in the ascending order of their addresses, starting with the control registers specified by R_1 and continuing up to and including the control register specified by R_3, with control register 0 following control register 15. The second operand remains unchanged.

If any of the bits loaded into positions 26-31 of control register 0 are ones, a data exception is recognized.

Condition code: The code remains unchanged.

Program interruptions: Privileged operation, addressing, specification, data

	Mnemonic	Type	Exceptions	Code
STORE MULTIPLE CONTROL	STMC	RS	M, P, A, S	B0

The set of control registers starting with the control register specified by R_3 stored at the locations designated by the second operand address.

The storage area where the contents of the control registers are placed starts at the location designated by the second operand address and continues through as many storage words as needed. The control words are stored in the ascending order of their addresses, starting with the control register specified by R_1 and continuing up to and including the control register specified by R_3, with control register 0 following control register 15. The control registers remain unchanged.

Condition code: The code remains unchanged.

Program interruptions: Privileged operation, protection, addressing, specification

Storage Protection Extensions

The advanced storage protection features of the IBM System/360 have been made still more useful and flexible in the IBM 2067 processor and core storage units. Although the storage protection keys are functionally a part of core storage, they are discussed here because they are logically related to the new and expanded features of the processor.

Storage protection is achieved in System/360 by dividing main storage into blocks of 2048 bytes. A four-bit key is associated with each block. When storing of data is specified by an instruction from the processor or a channel, this storage protection key is matched with another key supplied by the current or channel address word (CAW). When a mismatch is detected, storing is suspended and a processor interrupt occurs. The protected storage location remains unchanged.

The four-bit storage protection keys have been extended to seven bits. Bits 0-3 are the standard four-bit storage protection keys. Bit 4 is the fetch protection bit, bit 5 is the reference bit, and bit 6 is the change bit. The protection keys in the PSW and CAW remain unchanged at four bits in length.

Fetch Protection

When the fetch protection bit is 0, no protection against fetching by either a processor or a channel is indicated, regardless of the value of the four-bit storage protection key in the PSW or CAW, or the value of bits 0-3 of the storage key.

When the fetch protection bit is 1, the data or instructions in the corresponding storage block are protected against fetching whenever they are protected for storing.

When an instruction causes a fetch protection violation, execution of the instruction is terminated, a program interruption occurs, and a protection exception is indicated in the old PSW. The protected information is never loaded into a register or moved to another location.

Fetch protection violations caused by a channel result in a termination of data transmission; thus the protected information is never transferred to any output medium. The violation is indicated in the channel status word (CSW), which is stored as a result of the channel operation.

Locations whose addresses are generated by the processor for updating or interruption purposes, such as the timer, CAW, and PSW, are not fetch-protected. When a program specifies these locations, however, they are subject to protections.

Reference and Change

Bit 5 of the storage key (the reference bit) is set to one each time the corresponding storage block is accessed for storing or fetching by a processor or channel. Bit 6 (the change bit) is set to one each time data is stored in the corresponding storage block by a processor or channel.

The storage key is not part of addressable storage. The key is changed by the Set Storage Key instruction, and is inspected by the Insert Storage Key instruction. In these instructions, bits 0-6 of the storage key correspond to bits 24-30 of the register designated by the R_1 field.

The reference and change recording is always active. It is independent of (1) the problem supervisor, or masked state of the processor, (2) the type instruction of I/O command being executed, and (3) the manner in which the address is generated. Hence, references for updating or interruption purposes, such as the timer, CSW, or PSW locations, are included in the reference and change recording.

Test and Set

When two processors simultaneously request access to storage, access is granted for the duration of one storage cycle, not for the duration of an entire instruction. Since, as a rule, information is processed in internal registers, a typical procedure is to fetch data from storage, process the data in registers, and place the results in storage. The storage-interlocking mechanism, unaware of the relationship between successive storage accesses, makes it possible for one processor to store results in a location after the second processor has fetched its operands from that location, but before the second processor has stored the results. This possibility necessitates a programmed interlock in the use of core storage as a shared medium. This requirement becomes even stronger when one processor is to operate upon an entire storage area, before permitting a second processor to operate upon it. The Test and Set instruction was designed specifically for this requirement. When this instruction is performed, a bit is tested and later changed to 1 without access to the particular bit by any other processor in the intervening period; the instruction is available to the problem program and supervisor alike. By means of the protection feature, testing and setting of particular control bits can be reserved for the supervisory program. A concise description for the Test and Set instruction follows.

Mnemonic TS		Format SI
93	B1	D1

The leftmost bit of the byte at the first operand address is used to set the condition code, and the entire byte is set to all 1's.

The byte in storage is set to all 1's as it is fetched for the bit test. No other access to this location is permitted between the moment of fetching and the moment of storing all 1's.

Resulting condition code:
0 Leftmost bit 0
1 Leftmost bit 1
2 --
3 --

Program interruptions: Protection (termination—storage remains unchanged upon protection violation, but the condition code is unpredictable) and addressing (suppression)

High-Resolution Interval Timer

Each processor in the system is equipped with a program-resettable interval timer having 13-microsecond resolution. Actual implementation includes the use of an internal register to reduce storage interference to the level of the standard 60-cycle timer on the Model 65. Automatic interrupt occurs when the interval set is completed. The time remaining in the interval may be read under program control at any time.

IBM 2365 PROCESSOR STORAGE

The time-sharing systems utilize the IBM 2365 Processor Storage, Models 2 and 12. The specific model chosen for a system depends upon the number of processors and channel controllers that must communicate with it.

For all models the storage cycle time for eight-byte access is 750 nanoseconds. References to even-numbered and odd-numbered doublewords are interleaved within each unit for an effective storage rate as high as eight bytes every two cycles of the processor (400 nanoseconds).

Models 2 and 12 have a capacity of 262,144 (256K) bytes.

Model 12 provides the capability of communicating with multiple processors and channel controllers, and is therefore used in multiprocessor configurations. For simplex systems up to four Model 2s are available, providing a maximum of over 1 million bytes of processor storage. In multiprocessing systems up to eight Model 12s are available, providing a maximum capacity of over 2 million bytes.

In multiprocessor configuration, each storage unit can be separately partitioned for diagnostic purposes or to partition the system. System partitioning does not in itself affect the addresses assigned to the storage unit. When the processor refers to a storage location that is not available to it, the reference is suppressed and the program is alerted by means of an interruption condition.

Each processor in the system is provided with a separate bus connecting it to each storage unit in the system. Each bus can be connected to as many as eight storage units.

CHANNELS

General Information

The I/O control element permits I/O data transfer to proceed concurrently with processor operation. It provides for the attachment of a wide range of devices, from a manual keyboard to an I/O device having a data rate exceeding 1 million bytes per second.

The I/O control element consists of an IBM 2846 Channel Controller, 2870 Multiplexor Channels, and 2860 Selector Channels. A multiprocessor configuration can include four I/O control elements, each independently controllable by the processors. The I/O transmission capability of the system can be readily expanded by adding more channel controllers, more channels to a channel controller, or more subchannels on the multiplexor channel.

IBM 2846 Channel Controller

The IBM 2846 Channel Controller is contained in a single stand-alone frame and has its own power supply. It permits interconnection of multiple I/O channels to multiple processors and multiple core storage units. Seven I/O channels, four processors, and eight storage units may be interconnected by one channel controller. Four controllers can be attached to a System/360 Model 67. The I/O channels attached to the channel controller are the 2860 and 2870. Since a total of seven channels can be attached to one controller, TSS can have as many as 28 channels.

The channel controller provides for concurrent operations of all channels attached to it.

The two main functions of the 2846 controller are to:

- Control communication among attached processors and attached channels.
- Control communication and data flow among attached storage units and attached channels

The channel controller provides the operational control signals to perform the functions described above during normal program operation. The signals include those necessary to start and terminate I/O operations, perform addressing, establish processor channel and storage unit priorities, and

to respond to I/O interrupts. Signals are also provided to perform checking and maintenance operations such as channel checking, fault locating tests (FLT's), and other diagnostic functions.

In order to direct interruption conditions from the various channels to the processors, mask bits are accepted from each processor to indicate the interrupt status for each of the attached channels. When a channel signals an interruption condition, the channel controller directs the interruption to the appropriate processor as controlled by the mask bits.

The priority of the processors having access to the I/O channels is dynamically determined by the time-sharing supervisor. The priority of the I/O channels requesting storage units is assigned in the following manner: Each channel sends a "storage request" signal to the storage selection element (SSE) in the channel controller when needing a storage cycle. The SSE continually scans these lines. Upon reaching a channel requesting service, the SSE establishes priority for that channel. Once this occurs, priority will not be established again until the SSE processes the current request. At this point the scanning resumes at the highest-priority channel. This enables the SSE to respond quickly to the higher-speed devices.

Each channel controller in the system has its own unique interface with each of the storage units, therefore allowing simultaneous storage unit/channel controller operations. Conflicts arise only if two or more channel controllers and/or processors reference the same storage unit at the same time. If a conflict does arise, it is resolved by the storage unit.

In order to select the proper storage unit, the SSE must take into account:

- The value of each processor's current prefix
- The state of the "floating address" switches, whereby any storage unit can be assigned any address increment, if the feature is installed
- The state of the partitioning switches if the feature is installed

Using the above information, the SSE matches the decoded address to the proper physical storage unit and performs the gating of control and data lines.

IBM 2860 Selector Channel

The 2860 provides high-speed data transfer to or from I/O devices. It is capable of handling data rates up to 1,300,000 bytes per second. Each selector channel attaches up to eight I/O control units and can address as many as 256 I/O devices. Only one I/O device per selector channel can transmit data at any given time; other devices attached to the channel can, meanwhile, perform operations that do not involve data transfer, such as positioning a disk access mechanism.

Up to seven selector channels can be attached to each channel controller. All channels can operate concurrently if the aggregate data rate does not exceed the capacity of the channel controller.

IBM 2870 Multiplexor Channel

The 2870 provides for economical operation of multiple low-speed or medium-speed devices, neither of which justifies independent channel equipment. The multiplexor channel achieves economy of operation by sharing equipment among concurrent operations. The channel dwells on any one operation only for the time required to respond to a request for service from a device.

An I/O device on the multiplexor channel can operate in either the multiplex or burst mode. In the multiplex mode a single I/O interface can be time-shared by multiple low-speed I/O devices. In this mode the device remains connected to the channel only for the time required to accept or present a byte of data or to exchange status or control information. When this sequence is completed, the device disconnects and another device can be serviced. In the burst mode the device remains logically connected to the channel for the duration of a sequence of signals, thus monopolizing the I/O interface.

The logical entity capable of sustaining an independent I/O operation on the multiplexor channel is referred to as a "subchannel". Except for addressing and the control of interruption, each subchannel on the multiplexor channel appears to the program as an independent channel. Each subchannel is capable of executing its own chain of commands and of preserving the information associated with an I/O operation.

The 2870 is equipped with two types of subchannel and two types of interface for connected I/O devices. For the purposes of addressing and masking interruption conditions, the 2870 is considered a single channel.

The basic interface on the 2870 can accommodate operations in either the multiplex or burst mode. It provides for attaching up to eight control units and addressing up to 192 I/O devices. The basic interface can be associated with up to 192 subchannels, each of which is related to a unique I/O device.

A medium-speed interface can accommodate eight control units and provides for addressing up to 16 devices on the interface. It is associated with a single subchannel and, hence, can sustain only one operation at a time. Whenever the program addresses a device on that medium-speed interface,

the channel refers to the common subchannel. When the subchannel is already involved in an operation, the channel causes the busy condition to be signaled. The medium-speed interface is designed to operate only in the burst mode.

The 2870 can have one basic multiplexor interface and up to four medium-speed interfaces. The basic interface can handle an aggregate rate of 110,000 bytes per second when no medium-speed interfaces are provided. Table 1 (refer back to "Systems Data Flow") illustrates the data rates that can be sustained by the basic interface and the selector subchannels of the IBM 2870 multiplexor channel.

I/O CONTROL UNITS

The following is a brief description of the control units that may be attached to a time-sharing system. Where appropriate, control units may be equipped with a two-channel switch that permits the control unit, under program control, to be available to either of two channels. (Refer to the section entitled "Programming Systems" to determine the support being provided for each unit.)

IBM 2803 Tape Control

The 2803 Tape Control unit is a stand-alone unit capable of controlling up to eight tape drives in any combination of IBM 2400 series tape units.

The seven-track compatibility feature permits any attached drive with a seven-track read/write head to read or write tape in seven-track format compatible with tape generated on an IBM 729 or 7330 Magnetic Tape Unit.

In addition, a 2803 may address up to 16 tape drives with two IBM 2816 Switching Units. The 2816 provides switching between two to four 2803s and IBM 2400 series tape units. Switching is on a per-record basis under program control. The ability to manually isolate any tape drive or drives from the control of one or more tape controls is provided.

IBM 2820 Storage Control

The 2820 Storage Control unit provides the capability for controlling up to four IBM 2301 Drum Storage units for a total data capacity of up to 16,360,000 bytes. Data is transferred to and from a selector channel at a rate of 1,200,000 bytes per second. The 2820 provides file protection for each attached drum. The ability to protect a logical file is provided by the combination of commands in the unit and by checks within the control program servicing the file system.

Automatic data checking on reading is provided by means of a pattern of 16 check bits that are recorded on the drum after each count, key, and data area. Checking is accomplished by comparing the recorded check bits with bits generated while reading.

The 2820 is program-compatible with the IBM 2841 Storage Control unit. File organization and format are under program control, allowing each data and key field to be individually variable in length.

The two-channel switch feature provides facilities for programmed switching of the 2820 to either of two channels, only one of which can at any time be communicating with the 2820. The two channels can be on the same channel controller or on different channel controllers.

IBM 2821 Control Unit

The 2821 Control Unit provides a control and buffer storage unit for card readers, punches, and printers. The 2821 Model 1 controls one IBM 2540 Card Read Punch and one IBM 1403 Printer, and the 2821 Model 2 controls a 1403 Printer.

IBM 2822 Paper Tape Reader Control

The 2822 Paper Tape Reader Control unit provides status and data information from the IBM 2671 Paper Tape Reader to the processing unit. In addition, the 2822 checks for parity and signals end of record and end of tape. The unit uses an available control unit position on a system selector channel or a subchannel on the 2870.

IBM 2841 Storage Control

The 2841 Storage Control unit provides a means for attaching random access storage devices to the various models of the System/360. The IBM storage devices that can be attached to the 2841 are:

2302 disk storage, Models 3 and 4
2311 disk storage drive
2321 data cell drive, Model 1
7320 drum storage

The 2302 Model 3 has two access mechanisms; the 2302 Model 4 has four. The 2311 and 2321 each have one access mechanism. The 7320, although a drum unit, is considered to have one access mechanism. The 2841 can handle a maximum of eight access mechanisms in any combination of the above-named units.

The 2841 performs the following functions:

- Interprets and executes commands
- Translates data as it moves to and from the serial-by-bit storage units to the parallel-by-bit interface

- Checks the validity of the information as it is transmitted to and from the storage unit
- Furnishes status information to the system

The basic unit of information recorded on storage devices attached to the 2841 is eight bits (one byte).

As each byte of information is transferred from core storage to the file, the parity bit associated with the byte is removed by the 2841. This results in a substantial improvement in storage capacity. When data is returned to the system, the control unit assembles the serial-by-bit file data into the eight-bit bytes complete with the necessary parity bit.

The validity of the information recorded by the 2841 is checked automatically by appending a string of 16 bits to the end of each count, key, and data area. This type of checking is called cyclic code checking. During a write operation, an arithmetic operation is performed on the data as it is being written. The remainder generated by this operation forms the 16-bit check code, which is appended to the data train. During reading, the data train and its appended remainder are again operated upon arithmetically. If there are no errors, the remainder consists of all zeros.

Information written by the 2841 can be verified by executing the appropriate read command (for example, read-data, read-key and data, read-count key and data). Execution of any read command causes the contents of the count, key, and data areas to be operated upon arithmetically. The read command used for verification can be given in such a way that no core storage is required for this operation.

Single-record or multiple-record verification is possible. Multiple-record verification is accomplished by arranging channel commands in a chain. Once the verify operation is initiated, verification continues until the command chain is exhausted or until the end-of-cylinder is reached.

The ability to protect logical files is provided by the combination of commands in the 2841 and by checks within the control programs servicing the file system. Optional features of the 2841 are as follows:

- A scan function permits searching through random access storage for a specific record or condition.
- An additional storage feature enables a 2841 to control and address eight additional IBM 2302 access mechanisms (four more modules), thus doubling the total online capacity.
- The record overflow feature provides greater utilization of the available storage capacity by allowing a record to overflow from one track to another.

I/O DEVICES

Operator's Console

The IBM 1052 Printer-Keyboard operating with or without the IBM 2150 Console forms the operator's console. By using this console, operator-to-program and program-to-operator communication such as program checking, program correction, and job logging can be performed. Specifically, the operator's console, under the control of a supervisor program, provides facilities by which:

- The operator can enter data into the time-sharing system
- The system can give printed output
- The system can give an audible alarm to the operator
- The operator can present an attention signal to the system

Functionally, the operator's console consists of a 1052 Printer-Keyboard as the console I/O device and a control unit that communicates between the I/O channel and the printer-keyboard. The control unit is housed in the processor or the 2150.

In addition to containing the control unit for the 1052, the 2150 console has provisions for mounting two remote operator control panels (ROCP's). The ROCP, with the exception of the emergency pull switch, duplicates the control in the operator control section of the System/360 system panel. The purpose of this ROCP is to enable the operator to control the power-on, power-off, and other conditions of the CPU associated with the system panel. There is no connection between the ROCP and the console control unit, other than that they share the same physical box.

IBM 1403 Printer

The IBM 1403 Printer is controlled, buffered, and attached to the System/360 channel by the 2821 Control Unit. The 1403 Printer, Model N1, prints 132 characters per line at a rated speed of 1100 lines per minute (lpm). Equally important is the dual-speed carriage, which permits skipping at about 75 inches per second on skips over eight lines.

The 1403 Model N1 uses an IBM 1416 Interchangeable Train Cartridge, which enables quick and easy changing of type fonts, styles, or character arrangements.

A significant new feature, the Universal Character Set, provides for printing any set of 240 characters in any desired sequence on the 1416. Six new trains are provided as standard for optimizing various applications. One train, which includes all alphameric characters and three special characters, will provide printing rates of 1400 lpm maximum, 1158 lpm minimum, and a nominal rate of 1250 lpm.

IBM 2301 Drum Storage

The 2301 Drum Storage provides direct-access storage of approximately 4,000,000 bytes at a data rate of 1.2 megabytes per second. The data rate is achieved by accessing four bits of information in parallel. Data is recorded on 800 tracks, divided into 200 addressable groups of four tracks read in parallel. Average access time is 8.6 milliseconds (ms). Data records can be of variable length.

Additional 2301s may be attached to the System/360 Model 67. Thus, transfer rates in multiples of 1,200,000 bytes per second are attainable.

IBM 2302 Disk Storage

The 2302 Disk Storage is a large-scale, high-speed direct access storage device for the System/360. The 2302 Model 4 has two storage modules with a total storage capacity of 224,280,000 bytes. Each storage module has two independent access mechanisms, each of which can access approximately 56 million bytes. The cylinder concept is used with comb-type access mechanisms containing vertically aligned heads for each disk surface.

Access times vary from 50 ms for a minimum track-to-track access, to 180 ms maximum for access to any track.

The high-speed sequential reading and writing rate of 156,000 bytes per second permits efficient sequential processing in addition to random processing.

IBM 2311 Disk Storage Drive

The 2311 Disk Storage Drive provides random access storage for 7,250,000 eight-bit bytes in each disk pack. In the packed decimal mode, the capacity is 14,500,000 numeric characters. Eight disk storage drives can be attached to each storage control unit.

In addition to this online capacity of 58,000,000 alphameric or 116,000,000 numeric characters per control unit, virtually unlimited data storage capacity is possible because the operator can exchange disk packs on a drive within one minute.

The 2311 provides a data rate of 156,000 bytes per second or 312,000 digits per second. Average access time is 85 ms, with track-to-track access time of 30 ms.

IBM 2314 Direct Access Storage Facility

The 2314 Direct Access Storage Facility consists of eight independent modules, each storing up to 25,870,000 eight-bit bytes or 51.75 million digits in packed decimal mode in an IBM 2316 Disk Pack. The eight removable and interchangeable disk packs provide a total of 207 million bytes of online storage and virtually unlimited offline storage. The average access is 75 ms, and the maximum access is 140 ms. File Scan, which performs a comparison on selected bytes of file information, and Record Overflow, for greater utilization of storage, are standard features. Enhanced system reliability and performance is achieved by providing a ninth "spare" drive for use if one of the eight normally addressed drives becomes inoperable. Data is transmitted at the rate of 312,000 bytes per second. The 2314 is equipped with an integrated control unit having characteristics similar to those of the 2841.

IBM 2321 Data Cell Drive

The 2321 Data Cell Drive provides very large-capacity direct-access storage for System/360. Each 2321 accommodates up to ten removable and interchangeable data cells. Each data cell contains 200 strips, which are the basic recording medium.

The 2321 has a storage capacity of 400 million bytes or 800 million decimal digits. Up to eight 2321s may be attached to a single IBM 2841. Thus, one 2841 may control over 3 billion bytes of information.

The cylinder concept is used for the 2321, as it is for the 2311. The 2321 may store variable record lengths of up to 2000 bytes.

Average access time for selection of a strip ranges from 175 to 600 ms; average rotational delay, once a strip is on the drum, is 25 ms; and access time to another cylinder averages 95 ms.

Reading and writing is done at the rate of 55,000 bytes per second.

IBM 2400 Series Magnetic Tape Units

The 2400 series magnetic tape units use a two-gap, nine-track (eight data tracks and one check track) read-write head, with the first gap used for writing and the second for reading. The two-gap head allows automatic error checking of the tape while it is being written. As an optional feature, a seven-track head is available. The seven-track head allows the 2400 tape unit to read or write BCD tape characters at densities of 200, 556, or 800 bits per inch. This feature permits reading and writing of tape compatible with current IBM 729 tape.

The 2400 tape units use the non-return-to-zero (NRZI) method of recording data on tape. In the NRZI system of recording information, tape is continuously saturated in either the positive or negative direction. A change in saturation polarity is called a one, and no change is called a zero.

The 2400 tape units read or write in true binary form, or in the System/360 eight-bit code. However, the 2400 tape units can read or write BCD data if a seven-track feature is installed. To increase reliability, the bit positions experiencing the most frequent reversals of the polarity of saturation are arranged on the center track of the tape. The ability to read backward is a standard feature on all 2400 tape units. Recording density is 800 bytes per inch.

Error correction on a reread of a record containing one or more errors in the nine-track format is a standard feature. Another standard feature is tape-in column load, whereby tape automatically enters the tape columns after a reel is mounted and threaded and the load key is pressed. Each drive has a quick-release latch that facilities mounting and removing tape reels.

The 2400 tape units are available as single tape units (IBM 2401) similar in appearance to the IBM 729 magnetic tape units, as two units housed in a common frame (IBM 2402), or as a single tape unit combined with a tape control unit (IBM 2403). The 2400 tape units are available in three models, providing for data rates of 30,000, 60,000 and 90,000 bytes per second.

It should be noted that these data rates are given as bytes per second. The System/360 has the ability to pack two decimal digits in an eight-bit byte. Thus, if one had a 90,000-byte-per-second tape drive transferring entirely numeric data, the effective transfer rate would be 180,000 decimal digits per second.

The 2400 Model 3 tape drive operates at a tape speed of 112.5 inches per second, and a density of 800 bits per inch. The nominal interrecord gap is .016 inch, with a nominal gap time of 5.3 ms. Rewind time is one minute, and rewind unload time is 1.1 minutes.

IBM 2540 Card Read Punch

The 2540 Card Read Punch reads cards at a rate of 1000 per minute, and punches cards at a rate of 300 per minute. The read and punch sections are separate entities, except for one common pocket out of the five; reading and punching can take place simultaneously.

IBM 2671 Paper Tape Reader

The 2671 Paper Tape Reader has the capability of reading five-, six-, seven-, or eight-track codes. The tape can be 11/16 inch (five-track telegraphic code), 7/8 inch (six- and seven-track codes), or one inch (eight-track code). The type of tape used in the 2671 is chad tape. Up to 1000 characters per second can be read from the chad tape.

Optional spooling features are available that provide for center roll or reel feeding, and reel rewinding. Buttons and switches are provided that can be used to set particular end-of-record codes, tape codes and widths, parity, and deletion recognition. Lights are also provided that signal the operator concerning the status of the reader. The 2671 requires an IBM 2822 Paper Tape Reader Control. The 2671 can be placed on the 2822 as a table-top reader.

GRAPHIC AND DISPLAY TERMINALS

IBM 2840 Display Control

The 2840 Display Control unit performs the functions of communicating with the CPU channel via the System/360 interface and provides buffer storage, regeneration, and character generation for the 2250 Display Unit, Model 2, and the 2280 Film Recorder and 2282 Film Recorder/Scanner. Connection of the 2840 can be either through a multiplexor or selector channel.

The basic 2840 can initially accommodate two displays. An additional feature is attached to accommodate two more display units. Three features may be added to bring the number of attached consoles to a total of eight when the application makes this desirable.

IBM 2848 Display Control

The 2848 Display Control unit provides the interface control, character generator, buffer storage, and timing and control logic for the 2260 Display Station. The interface control provides the communication path between the data processing system and the 2260 display consoles. The data being transmitted along this communication path is translated by the character generator into the appropriate display characters. The buffer storage holds the data in the normal System/360 eight-bit-byte mode for cathode ray tube regeneration. This 2848 attaches to a System/360 selector or multiplexor channel. It is also possible to attach this unit remotely through a 1200- or 2400-baud communication lines and the proper subsets to an IBM 2701 Data Adapter Unit, which in turn is attached to the System/360 via either the selector or multiplexor channel.

IBM 2250 Display Unit

The 2250 Display Unit provides System/360 with the capability of visually displaying tables, graphs, charts, and alphameric characters. The display is made within a 12 x 12-inch area on the face of a 21-inch cathode ray tube. Data can be accepted from

the processing unit at a rate of 238,000 bytes per second for the 2250 Model 1. Over 1 million display points can be addressed by X and Y coordinates on each display.

This I/O unit provides broad man-machine communication capability and offers increased speed and flexibility for dynamic monitoring of computer operations, including current status of active system users and all queue lengths.

The 2250 can be used as a system operator console or a programmer console. In either case, it can be used for data retrieval and updating of monitor records from storage.

The 2250 Model 1 has a self-contained control and is used in applications where a single console is required. The 2250 Model 2 is used in combination with the 2840 for multiple displays. Up to eight 2250s may be attached to a 2840.

The following optional features are available for the 2250:

- Light pen. This is a pen-like device used by the operator to identify to the program a particular point or character on the display screen. The light pen can be used in conjunction with a keyboard and appropriate programming to rearrange, delete, highlight, or edit information.
- Programmed function keyboard. This is a 32-key general purpose keyboard. Interchangeable overlays define the key functions to the operator.
- Alphameric keyboard. This is a typewriter keyboard for alphameric data entry. This keyboard permits the entry of letters, numbers, and symbols into the display unit.
- Character generator. The character generator translates a byte into analog signals for tracing alphameric characters and symbols on the face of the tube.
- Absolute vectors. These enable the drawing of a straight line between any two points on the 2250 display.
- Operator control panels. Up to two operator control panels may be duplicated at the 2250 operator's console.
- Buffer. The buffer, which is internal to Model 1, provides the display unit with a choice of 4096 of 8192 bytes of internal storage. Additional buffer capacity is available for the Model 2 through the 2840 display control.

IBM 2260 Display Station

The 2260 Display Station is a compact cathode ray tube (CRT) display similar in technology to a television monitor. An optional keyboard feature provides for man-machine communication at the display station. This keyboard can be either numeric or alphameric. The 2260 can display up to 960 characters on the face of the CRT, arranged as twelve lines of 80 characters each. A maximum of eight displays of this type can be attached to a single 2840 Display Control. By reducing the total number of display characters per CRT from 960 to 480, it is possible to attach up to 16 display stations to a 2848 Display Control; a further reduction of 240 characters permits the attachment of 24 display stations.

The initial condition of the display station is a clear CRT except for a cursor in the upper left corner of the display. As the operator enters a message via the keyboard, each character enters the buffer in the 2848 associated with the display. The buffer then regenerates a flicker-free image on the CRT, permitting the operator to verify the message as it is being composed. At the completion of the message, the ENTER key is depressed and the CPU is signaled so that the message can be read out of the buffer and processed. As each character is entered by the operator, the cursor moves over one character position on the screen. Replies from the CPU are sent to the buffer, starting at the last cursor position and advancing the cursor to the end of the message. Thus, messages to and from the CPU may be intermixed and displayed as contiguous lines of text on the CRT.

Characters may be erased from the buffer by means of backspacing the cursor. An optional non-destructive cursor feature permits rapid movement of the cursor horizontally and vertically to any character position on the screen without erasing data.

An optional line-addressing feature permits CPU messages to be addressed specifically to any line on the CRT screen.

An optional IBM 1053 Printer adapter unit may be attached to the 2848, permitting the contents of a display station buffer to be printed as hard copy. Messages from the CPU may also be directed to the 1053.

IBM 2280 Film Recorder

The 2280 Film Recorder provides a high-speed output capability for recording System/360 output, both graphic and alphameric, on microfilm. Film recording is done with a CRT electron beam. A computer program directs the CRT beam motion, turning it off and on to trace the desired image in a serial manner on the film. The character generator in the 2840 control unit provides appropriate signals to the recorder/CRT for up to 40,000-character-per-second film exposure of standard alphameric and special characters. These standard alphamerics can be recorded on the film in any one of three

sizes, all under program control; the current capacity is 150 lines, with up to 204 characters to a line.

In addition to the alphamerics, straight lines may be drawn from any point to any other point of the 4096 x 4096 reference coordinate grid. The drawing speed is a function of the line length. The basic line drawing speed for line lengths up to one quarter of the maximum frame size is 102 microseconds. These lines can be drawn either by an absolute or relative addressing of points on the reference grid.

An exposed image can be developed and projected on the rear projection screen within 48 seconds after exposure. The film at the projection station can be advanced or backspaced manually. It is also possible for the operator to bypass the internal film processor.

IBM 2282 Film Recorder/Scanner

The 2282 Film Recorder/Scanner combines the functions of the 2280 Film Recorder and the 2281 Film Scanner into one unit. The recorder portion of the 2282 has therefore been included with the description of the 2280. The 2282 has a single film transport; therefore, only one function, recording or scanning, can operate at a time.

The scanner portion is a high-speed input device for digitizing negative microfilm images such as drawings, charts, maps, and graphs for direct input to System/360. Images are scanned by moving a CRT electron beam across the face of the film in a pattern specified by a computer program. A digital reading occurs when light transmission through the film meets or exceeds the program-selected threshold sensitivity. Any one of 63 levels of threshold sensitivity can be program-selected. Scan vectors can be drawn from any point to any other point of the 4096 x 4096 coordinate grid. As in the case of the recorder, this scan speed is a function of the line length. The basic line-scanning speed, for line lengths up to one quarter of the maximum frame size, is 102 microseconds. For small area scanning, a scan stroke mode is provided. These scan strokes are short vectors drawn at a 20-microsecond rate.

REMOTE TRANSMISSION

Introduction

One of the basic functions of the System/360 TSS is to provide easy access to all the facilities of the central data processing system, especially its computing power and large core storage.

The units described in this section provide remote users with this capability via either private or common-carrier transmission systems.

Of particular importance in this section is the description of remote low-speed conversational consoles or terminals, such as the IBM 1050 and IBM 2741 communication terminals, which will satisfy the I/O computing requirements of many research scientists. Also, these devices will supply many administrative departments with the ability to remotely access or maintain a central data base.

IBM 2701 Data Adapter Unit

The 2701 Data Adapter Unit greatly expands the I/O capabilities of System/360. The 2701 provides for the connection and control of the information flow of a variety of remote and local external devices with System/360.

IBM 1013 Card Transmission Terminal at 50 to 400 cards per minute

IBM 7702 Magnetic Tape Transmission Terminal at 250 to 300 characters per second

IBM 7711 Data Communication Unit at 250 to 28,000 characters per second

IBM 7740 Communication Control System

*IBM System/360s with similarly equipped 2701s

*IBM System/360 Model 20 with communication adapter feature

*IBM 1800 Data Acquisition and Control System

The 2701 can be attached to either the multiplexor or selector channel of the System/360.

IBM 2702 Transmission Control

The 2702 Transmission Control unit allows attachment of multiple communication circuits to System/360. It directs and controls information flow between the system and a variety of remote communication terminals, over private and commercial common-carrier transmission systems. Some of the terminals and systems that may be attached are:

IBM 1030 Data Collection System

*IBM 1050 Data Communication System

IBM 1070 Process Communication System

*IBM 2741 Communication Terminal

AT&T 83B2 selective calling stations

Western Union Plan 115A outstations

*Model 35 Teletype terminal

*TSS will provide the necessary program control to facilitate transmitting to and from this remote device.

The 2702 is modular and flexible in line capacity, transmission code, and speed. As many as 15 half-duplex lines, operating at speeds of up to 180 bits per second, may be attached to a basic 2702. Additional line capacity and higher speeds of operation can be attained by adding optional features. Line adapters may be of different types, allowing a mixture of terminal types of various speeds to be connected to one 2702.

Optional terminal controls and line adapters can be specified to match system requirements. Also available are:

- Speed extension features, which increase the line speed capability to 600 bits per second on all 15 lines of the basic 2702.
- 31-line expansion feature, which increases the line attachment capacity of the 2702 to 31 half-duplex lines at speeds up to 200 bits per second. Inclusion of this feature excludes the speed extension features.
- Automatic call feature, which provides (with automatic call adapters) automatic capabilities for up to eight common-carrier switched telephone or 150-bit-per-second teletypewriter (TWX) attachments.

IBM 1050 Data Communication System

The 1050 Data Communication System provides a low-cost, conversation-oriented terminal. It is a modular system that permits a variety of configurations according to specific application requirements. Each system can include as many as one keyboard, two printers, two punches (paper tape and/or card punch) and two readers (paper tape and/or card punch reader).

The system's outstanding features are:

- Reading, punching, and printing at 14.8 characters per second.
- Parity and longitudinal data checking during transmission.
- Input -- punched cards, paper tape, or keyboard.
- Output -- printed or punched on cards or paper tape.
- Ability to create punched output in an offline mode for later transmission to the central processor in the online mode.

The 1050 may consist of the following units:

1051 Control Unit

1052 Printer-Keyboard for printed output or keyed input

1053 Printer for receive-only capability

1054 Paper Tape Reader

1055 Paper Tape Punch

1056 Card Reader

1057 Card Punch

The minimum terminal would be a 1051 Control Unit, 1052 Printer-Keyboard, and an RPQ break feature. Input to the central processor would be through the keyboard similar to that of a typewriter. Output from the central processor would be in the form of printed copy on the printer at 14.8 characters per second.

IBM 1070 Process Communication System

The 1070 Process Communication System is a teleprocessing system designed specifically to meet requirements for a two-way data communication between remote process locations and a central data processing facility. It provides real-time, online control and data collection when attached to the System/360.

The system is highly modular in design and permits a variety of configurations to meet specific application requirements. Each IBM 1071 Terminal Control unit can have up to 300 points of I/O. This I/O can consist of analog, contact sense, decimal, and BCD input as well as contact operate, decimal, and alphameric output. Random and sequential scanning is possible, and all data transmission is completely checked.

Depending on the model of the 1071 used, the system can send the received data at 14.8 or 66.6 characters per second. This is equivalent to 5 or 21 analog conversions per second. Each system may include manual input units, display units, and output pointers as well as process I/O signals.

IBM 2741 Communication Terminal

The 2741 Communications Terminal looks and acts much like an IBM SELECTRIC ® Typewriter. The only terminal controls located in the keyboard area are the on/off switch and the Attention key. The keyboard is that of the standard SELECTRIC Typewriter 01 or 1052 arrangement with a selection of printing elements. Code and systems compatibility provides for transmission to System/360 via 2702s. The interrupt feature, permitting transmission from the processor to be halted at the operator's convenience, is required for time-sharing systems use.

REMOTE COMPUTER SYSTEMS OR HIGH-VOLUME TERMINALS

Introduction

As indicated in the previous section, low-speed conversational consoles meet the needs of many research scientists to remotely access the central facility.

There are, however, many research projects or administrative applications that require high-volume input and output from the central computer. Also, there are projects that require a small independent computer system on the project location, for example, controlling or gathering data from an experiment on a real-time basis.

It is important, therefore, that smaller computing systems be able to communicate directly with the System/360 TSS when an application requires larger memory, additional I/O facilities, or data from a centrally located pool.

The time-sharing supervisor will provide the necessary program control to facilitate transmitting to and from the remote systems on high-volume terminals.

IBM 1800 Data Acquisition and Control System

The 1800 Data Acquisition and Control System is designed to handle a wide variety of real-time applications, including process control, high-speed data acquisition, and direct digital control. The 1800 consists of a family of real-time I/O devices including an analog input, analog output, contact sense, and contact operate, as well as data processing I/O devices including graph plotters, line printers, magnetic tapes, random access files, and card and paper tape input and output.

The 1800 processor-controller may be attached directly to System/360 processors via the System/360 selector and multiplexor channels and the 1800 data channels. Also, the 1800 can be remotely attached via the 2701 Data Adapter Unit, utilizing common-carrier facilities.

The specific technique for attaching the 1800 system to the time-sharing system is to attach one 2701 to the 1800, and one 2701 to the time-sharing system. Data is then transferred from one 2701 to the other 2701 via the synchronous communication lines and the common-carrier facilities. The data is transferred between processors (2701 to 2701) at a rate of up to 40.8 kilobauds per second or 5100 bytes per second. A data channel, an I/O channel adapter, and a 2701 attachment to the I/O channel adapter are needed to attach a 2701 to an 1800 system (RPQ reference numbers C08041 and C08037).

System/360 Model 20

A synchronous transmit/receive (STR) feature is available on the processor of the System/360 Model 20. This feature permits data to be transmitted across customer or common-carrier facilities at a rate of 300 characters per second. In addition, an RPQ is available that increases the transmission rate to 5100 characters per second for transmission over 40.8-kilobaud service. It is also possible with a second RPQ to select either 300 or 5100 characters per second as an operating speed.

Other System/360s

IBM System/360s equipped with 2701s can be connected from a remote location to the central facility. The IBM 2701 permits data to be transmitted between the two systems via private or common-carrier communication lines. The remote System/360 could operate under Operating System/360 and perform jobs autonomously, as well as communicate with the central facility by either sending or receiving data as programs. Thus, the autonomous remote system is able to access a central data base and is able to make use of the central system's larger I/O devices and expanded memory.

GENERAL DESCRIPTION

Many of the hardware characteristics of the System/360 have been designed with multiprogramming in mind. Storage protection to prevent interaction between different jobs residing in core at the same time is an example. Privileged instructions, such as those that control I/O operations and manipulate storage protection keys, are another. The interval timer and supervisor states are further examples. All of these system facilities are used to facilitate multiprogramming. Multiprogramming, in turn, is used to achieve maximum throughput from a given system.

Maximum systems throughput does not necessarily provide maximum service to each individual user of the system. Even in an efficient batch-processing shop, the time span between submission of a job and the return of the results is such that users generally work on more than one job at a time. It is much easier for the computer, however, to switch rapidly from one task to another than it is for the human using the computer to switch from one task to another.

To provide efficient service to all users, the System/360 TSS allows users easy and open access to the central data processing facility. Multiple remote consoles provide many simultaneous users with a direct means of monitoring and controlling the computers that are servicing their needs. These users at remote consoles can, if they desire, limit themselves to the batch-processing environment. Experience has indicated, however, that users will take advantage of the broader range of services that can be part of a time-sharing system. These services include online debugging, a mode of operation that can be very useful to the programmer but anathema to the machine room supervisor in a batch-processing system. The inefficiencies of system utilization during console debugging largely disappear in a time-sharing environment.

System/360 TSS, designed to operate in this type of multiple console environment, takes advantage of systems features specifically designed for time sharing. These features include multitailed storage units, multiple central processing units (CPU's), dynamic relocation devices, high-speed multiplexor channels, dual I/O paths, store and fetch protection, and system partitioning capabilities.

TSS includes:

- A time-sharing supervisor with a time-slicing capability
- A command language for communication between user and system in both conversational and batch modes
- Data management and cataloging facilities
- A mnemonic assembly language compiler with macro capability -- batch and conversational syntax analysis modes
- A FORTRAN IV compiler -- batch and conversational syntax analysis modes
- A Programming Language/One (PL/I) compiler -- batch and conversational syntax analysis modes
- COBOL compiler -- batch mode
- A sort/merge program, oriented to the disk file
- Linkage Editor and Dynamic Loader -- batch and conversational syntax analysis modes
- Program checkout system -- batch and conversational modes
- A library of mathematical and utility programs, open-ended so that user-developed programs and routines can be added easily.

SUPERVISOR

The supervisor manages the system, resides in core storage, is nonrelocatable, and will be used in parallel by each CPU. The supervisor handles all interrupts: asynchronous interrupts caused by I/O devices, hardware malfunction, and program checks; synchronous interrupts arising from the Supervisor Call (SVC) instruction, which allows programs to request service from the system.

COMMAND SYSTEM

The command system is the interface between user and system. It enables users to enter, manipulate, and control the running of programs; and it enables operators to control the operation of the system.

DATA MANAGEMENT

Data management facilities control input/output devices and provide device-independent operation for system programs and problem programs. The command system interfaces with the data management system through macro instructions. Both the data management system and the command system are relocatable, and required routines are loaded into the user's virtual memory.

GROWTH

System/360 TSS, like the System/360 itself, is designed to make growth and change logical and nondestructive. The supervisor is designed to handle multiple processors, multiple core modules,

and multiple channels and I/O devices. The logical partitioning capability in the supervisor can be extended in the opposite manner so that additional system components can be added in the future to take care of heavier system loads.

INTERFACES

The interfaces between the various modules of the System/360 TSS will be defined so that modifications and additions can be easily made to the system. IBM-supplied programs, such as the compilers, utility programs, etc., are treated no differently by the supervisor than user-supplied programs. This permits easy additions to the program libraries by both IBM and the user.

THE COMMAND LANGUAGE

TSS is a powerful tool that makes available a wide variety of program services. The user calls for these services through commands that he directs to the system through a terminal consisting of a keyboard, a printer, and perhaps a card reader or paper tape reader. These commands call into action a wide variety of systems programs. The user can enter and construct programs, debug these programs in conversation with the system, and request execution of these programs using specified data sets. Several commands provide for manipulation of data sets to permit additions, deletions, or modifications, to make duplicate copies, and to move them from and to I/O units as required.

The user is in constant communication with the system and is aware of the functions being performed by it. The system informs the user of action it has taken, of additional information required, and of mistakes in the request.

System Control Functions

The command language includes facilities that can be requested by the system operator, who monitors the performance of the entire system. The system operator may be directed to mount or demount volumes on tape drives, or direct access devices, as required by problem programs being run by different users. The system operator is also responsible for removing failing equipment from the system when hardware errors of certain types have been detected.

Although one terminal is sufficient, several terminals can be used for system control functions. For example, one terminal might be used to monitor the system operation and perform the functions of system startup and shutdown, and of attaching and detaching devices. A second terminal might be used to communicate instructions to an operator whose specific responsibility is mounting and demounting tape volumes as required by problem programs. A third terminal might be used for direct access devices.

Interpretive Execution

Interpretive execution means that when a command is entered into the system it is analyzed and the programs and services requested are executed immediately. When a command is issued, the system locates it in a list of permissible commands. When found, the parameters are analyzed and provided to the program, which performs the action required by the command. All of the functions performed by a command are completed before a new command is requested. After a command has been interpreted and executed, it is forgotten by the system and the next command is requested. The results of any action taken by the command are retained in the system, however, in the sense that data sets may have been loaded or modified, programs executed, etc.

Commands may be issued by a user at a terminal or they may be issued internally as prestored sequences. In either case, the command is interpretively executed before the next command is accepted.

System Communication

Communication within the system consists of four principal types of messages. The system operator may request that a specific message be directed to a specific user. Also, the system operator can direct a message to all users simultaneously, through a broadcast technique. In both of these types of message there is no restriction on the content of the message other than the acceptable characters for the terminals involved. Another form of communication is messages to the system operator to perform specific functions, such as mounting and demounting volumes on I/O devices. The final form of communication is from the system operator in response to a specifically requested action. For example, when the operator has mounted a tape on a specific drive, he must inform the task so that the task can continue.

System Input and System Output

As far as a user at a particular terminal is concerned, he has full access to the system and its services without regard to any other user who may be simultaneously using the system. When a user initially logs on at a terminal, a task is established

to service him for the duration of his stay at the terminal. The input expected from the user at the terminal consists of commands that specify which programs are to be executed. The source of these input commands is defined to be SYSIN. Similarly, messages sent from the task to the user who is directing or commanding it are sent via SYSOUT. In a conversational mode of operation, SYSIN is the keyboard or possibly a card reader or paper tape reader; SYSOUT is the printer, associated with the terminal that the user is using.

Conversational vs Nonconversational Tasks

In a conversational mode, the user gives commands to the system and analyzes the responses before issuing additional commands. The user can direct the system as he proceeds and may change his tactics in solving the problem or modify his approach as he examines the results of his action. By conversing in this manner, the system can be used as a powerful tool to aid in the solution of the user's problem.

In many situations, procedures can be defined ahead of time. There is no variation as execution of the problem proceeds. In this case, a user can establish a sequence of commands and store these commands in the system as a data set. The user can then establish a nonconversational task to execute these commands in the background. The SYSIN associated with this nonconversational task is not a terminal but a data set stored on a direct access device or tape or some other storage medium internal to the system. When a nonconversational task has been established, commands are taken one at a time from this data set to direct the program execution required. Similarly, SYSOUT for a nonconversational task is not a terminal but a data set that may eventually be printed on a system's printer. The originating user can examine the output at his convenience to determine how the execution of his nonconversational task proceeded.

Nonconversational tasks would include such things as compilation of large programs, execution of lengthy object programs, or manipulation of data sets in such operations as file maintenance or updating procedures.

Data Management

The command language provides the user with access to all the data management facilities of the time-sharing system: automatic program loading and linkage, the catalog system, control of direct access storage allocation, and a variety of data access methods. Commands may be issued to edit, update, or delete a data set, establish sharing procedures for cataloged data sets, request an I/O device, or resume running of an interrupted program.

The catalog is a list of the names and locations of all the data sets in the system that are accessible to the user. To access a data set, the user must give a name to it and use this name to recall it when necessary. One of the parameters that must be supplied in most of the commands available to the user is the name of the data set associated with the operations performed by that command.

There are three attributes of data sets with respect to a user:

1. Private. The first attribute indicates a private data set that is cataloged under a user's identification and is accessible only to him.
2. System. A second indicates systems programs, such as compilers, which are public and can be accessed by any user.
3. Shared. In addition, the system permits sharing of data sets. An individual user can specify other individuals who may have access to his data sets. The owner of the data set can restrict the sharing users' access as follows:
    a. Unlimited access to the data set
    b. READ-only access
    c. READ and WRITE access but erasing or replacing entries is prohibited.

Another service provides for program libraries. A program library is a partitioned data set whose members are object program modules. When a user joins the system, two libraries are cataloged for him: a system library containing standard programs, and a new user library containing his own programs. A program library list is created with entries for the user library and the system library.

The user may add a job library to the list by issuing a command that places this newest library at the top of the current list. As many job libraries as desired may be subsequently added. The list is used to store a new object module and to determine which libraries are to be searched, and in what sequence, to find a needed program module.

"Line" Data Set Attribute

Commands are available to create and manipulate "line" data sets in different ways. A "line" data set is composed of records that are fixed in length. Although the length of the record is fixed for an individual "line" data set, the length of the records may vary from one "line" data set to another.

Each record of a "line" data set represents a single line of information. The record is composed of two parts, the first being a line number and the second being the line of information itself. The

line number is an index by which the user can identify and reference a specific line within the data set.

When the user manipulates the data set, he can refer to a specific line or group of lines in such operations as deletion, insertion, or replacement. Similarly, he can print selected lines of the data set by specifying in a command the appropriate line numbers.

Commands entered by a user at a terminal can themselves be a "line" data set.

Prompting, Confirmation, Responses, and Diagnostics

A conversational task may require that considerable information be communicated to the user at the terminal in order that he may be informed of the status of the system, or of additional information required for performance of the requested action.

Each command contains two types of parameters: required and optional. Required parameters are those that must be supplied for the execution of that command. When a user gives the command he may not always specify all of the required parameters. In this case the system "prompts" him by printing a message requesting that he supply the specific parameters required. The user need not know precisely the format and order in which all of the parameters must be supplied with the command. If he has forgotten any of the parameters for a particular command, the system simply asks that he supply these parameters by sending him the appropriate prompting message.

An extension of prompting results in diagnostic messages in certain error conditions. For example, a value for a required parameter may be incorrect and not acceptable to the system. In this case, the user has not forgotten the parameter but instead has made a mistake. The system prints a diagnostic message to request that he supply the correct parameter. This form of prompting teaches the user how to use the system.

Optional parameters are those that the user may specify if he so desires. If he does not specify them, the system automatically selects a predefined value called a "default" condition. The system simply assumes the default value if the user has not supplied a value.

Confirmation is provided by the system, if the user requests it, in order to inform the user of action taken. Thus, a positive response is given to the user for each of the commands that he enters. This response may be a message to the effect that the command has been accepted or executed. However, when the user has specified that he wishes confirmation, the system prompts the user for optional parameters for which he did not specify a value.

In certain situations the system responds to the user whether he has requested confirmation or not. For example, when a user logs on, the system prints the time and date at which he logs on. When a user is removed from the system, some disposition must be made of his data sets. In this case the system prints the name of each data set, and requests instructions regarding the disposition to be made of each data set. In this situation a conversation between the user and the system exists in which the system asks the user what action must be taken.

In order that the system may inform the user that it is ready to accept a new command or a new statement, a special character is printed at the beginning of each line.

Expansion of Basic Command Language

Even though a basic set of commands is provided with the system, an authorized user may wish to add new commands. The program implementing a new command must be written to conform to the interface requirements of the command language system.

The name of the command must be put in the verb table, which is used to identify commands and to link to command programs. Once a user adds commands, he may at some point want to remove a command. To do this, the entry for the particular command must be removed from the verb table. Furthermore, the command program should be removed.

LANGUAGE PROCESSORS

The language processors have the following characteristics:

- They must either reside in core storage for substantial time periods or else be continually swapped from auxiliary storage. To achieve this, larger compilers are segmented into independent phases.
- The compilers are reentrant so that numerous users may utilize them. Work areas are associated with the source programs and never shared among users.
- The compilers use only direct access storage for scratch purposes: no sequential storage devices (such as tapes) are used.
- A common output format is produced by the assembler and compilers, and is structured to be further processed by the link editor and/or dynamic loader. Executable programs may be composed of sections of programs independently compiled or assembled.
- Special output options assist in checking out programs.

- Most language processors operate in both conversational syntax analysis and background modes.

TSS/360 Assembler

A relocatable reentrant assembler with marco capability will be available with TSS. Like the other language translators, it is treated logically as any problem program and, therefore, has all of the monitor facilities available to it. It is available in both conversational syntax analysis and background modes.

The assembler program provides auxiliary functions that assist the programmer in checking and documenting programs, in controlling address assignment, in sectioning a program, in data and symbol definition, in generating macro instructions, and in controlling the assembly program itself. Mnemonic codes specifying these functions are provided in the language.

The address translation feature of the IBM 2067 Processing Unit eliminates the need for overlays in the design of the assembler itself. Except for this facility, the characteristics of the assembler are similar to the Operating System/360 assembler. Where differences between TSS and OS/360 are required, they will be defined in a future SRL. (See IBM Operating System/360 Assembler Language, C28-6514.)

FORTRAN IV Compiler

A FORTRAN IV compiler is also available with TSS. It is a relocatable, reentrant routine designed in accordance with the conventions and requirements for systems programs in the System/360 TSS environment. It is treated logically as a problem program and has all the facilities of the supervisor available to it. The FORTRAN IV compiler may be used in either conversational syntax analysis mode or background mode.

Inputs to the compiler are source programs written in the FORTRAN language. The language is compatible with the FORTRAN IV (H-level) language supported under OS/360 and described in IBM Operating System/360 FORTRAN IV (C28-6515).

The compiler organization and information flow are designed for fast, efficient processing. In addition to the production of executable programs, the compiler detects and gives notification of source program errors and produces a variety of documentation describing the object program.

Programming Language/One (PL/I) Compiler

A reentrant Programming Language/One (PL/I) compiler is being implemented for TSS. It is available in conversational syntax analysis mode or nonconversational mode. The PL/I language combines the computational ability of FORTRAN with character handling and logical statement manipulations to permit the solving of problems of great diversity and scope. Several applications for which PL/I is especially suited are real-time command and control, systems programming, and problems of management science. The PL/I language is compatible at the source level with the H-level support under OS/360, as documented in IBM Operating System/360 PL/I: Language Specifications (C28-6571).

Output will be the same form as that produced by the assembler and other language compilers, and will be suitable for further processing by the linkage editor and/or the dynamic loader.

COBOL Compiler

A COBOL compiler is also available under TSS. It is nonconversational and operates in the background mode only. Output will be in the same form as that produced by the assembler, and will be suitable for further processing by the linkage editor and/or the dynamic loader.

The COBOL language is compatible at the source level with the F-level support under OS/360, with the exception of the SORT verb. See IBM Operating System/360 COBOL Language (C28-6516) for a description of the language.

Sort/Merge

A sort/merge program oriented to the disk drives will be provided for operation in a time-sharing environment. Its design goal is to minimize requirements on memory and I/O channels consistent with efficient operation.

In a time-sharing environment the complete facilities of the computing system are not available for storage and channel optimization; therefore, a high-volume sort will probably not run as efficiently as in a batch-processing environment. The requirement for extensive sorting can be minimized by the use of indexed sequential and partitioned data organizations supported under TSS.

Use of the SORT/MERGE program requires one additional disk drive over the minimum required for operation of TSS.

DATA MANAGEMENT

Introduction

The management of data in a time-sharing system poses many problems that either do not exist or can be ignored with less severe consequences in a more conventional batch-processing system. All data in the entire installation must be available to each terminal user within a "reasonable" time span; this capability should not be restricted by the data management system. Any restrictions on the unlimited accessibility of data should be imposed only by the users themselves, for security reasons for example. The size of the data sets may require large amounts of storage online and accessible within a matter of milliseconds. IBM supplies this magnitude of storage in a wide range of devices having large capacity and various degrees of accessibility.

Generally speaking, the cost of online storage devices varies inversely with the time required to gain access to data stored within them. The problem facing the designers of a time-sharing system is: How can the necessary secondary storage capacity commensurate with the required accessibility be provided at the lowest possible cost? This question must be asked by the designers of any data processing system, but the question is vastly more significant in a time-sharing system.

If the total data requirements could be determined, and if the activity could be accurately measured, the system designer's job would be fairly easy. This is almost never the case.

Secondary Storage Allocation

Data storage within the time-sharing system is organized into a hierarchy that might be described as constituting a triangle of variable dimensions (Figure 6).

Storage media closer to the base of the triangle store data at a lower unit cost but require longer access time. Storage media closer to the apex permit fast access at a higher cost per unit. The total area of the triangle corresponds to the total data capacity of the system. Response time can be improved by increasing the number and capacity of the devices with faster accessibility. The design of the data management system within the time-sharing monitor permits this flexibility without requiring recompilation of programs or reformatting of data.

Data Flow

Data flow through the system is from any auxiliary storage device into core and, if necessary, back

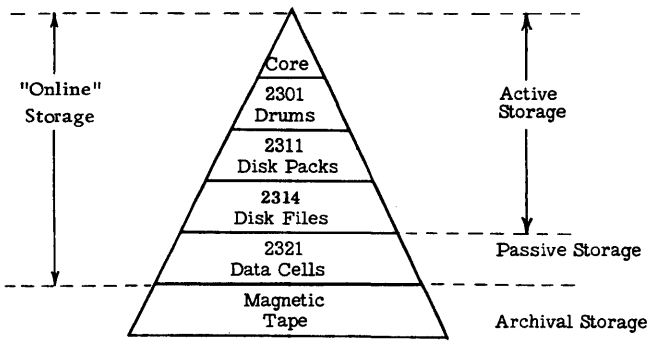


Figure 6. Storage allocation

into auxiliary storage as processing is completed. The data management routines within the time-sharing monitor maintain statistics on activity of each data set and assign them to storage levels within the hierarchy on the basis of activity. As a data set is used frequently, it will move upward within the storage hierarchy to devices of faster accessibility, and thus "closer" to core storage. As activity diminishes, it will tend to drift lower in the hierarchy to devices of less cost per unit of storage capacity.

The method of data management and flow has several advantages:

- The monitor achieves an automatic balance between the activity of data and the storage level to which the data is allocated.
- The bulk of the data and programs is kept in devices of lowest cost per unit of storage capacity, without compromising response time.
- The system capacity and response can be matched to the user's requirements without program modification.
- The ultimate in "fail-soft" capability can be achieved at the lowest cost. Duplexing of secondary storage is kept to the minimum commensurate with total capacity and response time.

The entire system philosophy of program and data set manipulation is based on the page, the basic unit of data in the system. A page may contain either a program or data. Pages are ordinarily the units extracted from cataloged locations within secondary storage and loaded into core. In a time-sharing system, they are then copied into a paging area on drum or, occasionally, on disk. The page flow then is from drum to core and back as determined by the task-scheduling and core-allocation routines within the monitor.

Each program or data set resident in the system is cataloged in an index maintained on one or more cylinders of a 2314 or 2311 disk file. Information concerning the size and location of the data set is

contained within the index, thus enabling the system to gain access to the data set itself, or any page within it.

When a page is brought into core storage, its availability is recorded in the appropriate page table entry.

Programs and data sets are introduced into the system in several ways. Certain tasks may create them, as is the case with compilers. Users may create them by requesting the system to save, under a new name, a data set with which they have been working. Data may enter the system from a wide variety of sources:

Keyboard input
Low-speed and high-speed card readers
Paper tape readers
Tape units
Disk packs
Satellite computers, local or remote
Remote data links

Data, by whatever means it arrives, must be cataloged, and storage of the required size must be assigned. Control information may govern the device type into which it is loaded, but not the specific I/O device. It need not remain in the device into which it was originally loaded. Depending on its activity, it may rise within the hierarchy of storage to a device with faster access speeds, or, if comparatively inactive, sink to a device of slower accessibility.

Lacking any other specification, it will be loaded into a 2314 or 2311 disk file. It may be loaded with control information specifying that it is to be kept in the load location, or it may be left to the system's regulatory ability to locate it in the device most suited to its usage and system storage capacity.

There are two levels at which data flow can be considered within the time-sharing system.

The first is the flow of pages from a cataloged storage location within the storage hierarchy, into working core storage in response to calls. Modified pages flow back to the cataloged secondary storage.

The second is a vertical movement between devices within different levels of the storage hierarchy. The system maintains the density of each storage device and brings programs and data sets into a level of storage accessibility commensurate with their activity. The most active of these will be found on the nonpaging drums and the least active on the 2321 data cells.

As additional programs and data sets are introduced into the system, provision must be made for a periodic purging of the lowest or most passive level of online storage to some type of archival storage. The archival storage may be tape, removable disk packs, or removable data cells. The purged programs and data sets are those of lowest activity in the system. The system keeps a record of the purged items and the tape reels, disk packs, or cells they are on, so that they can be reintroduced as necessary. Calls to purged programs and data sets are communicated to the operator's console so that the proper tape reels, disk packs, or cells can be identified.

One of the most intriguing concepts of data management that become feasible in a time-sharing environment is that of "percolate and trickle". "Percolate" implies the bringing up to higher-activity devices the more active data sets, while "trickle" implies the reverse ability of moving data sets down to the lowest storage device identified with a data set activity.

Vertical flow is expected to trigger automatically at preset intervals if there is idle time on any CPU and when core space and channel time are available. If any device has a density above an installation-designated "normal" level, a task will be created to reduce the density. The activity of each program or data set can be determined and, when an imbalance is indicated, a task can be invoked by the system to schedule an interchange from device to device, as space permits, interchanging higher-rate sets from devices lower in the hierarchy with lower-rate sets from devices higher in the hierarchy.

The object of this data flow activity is twofold: to use the various I/O devices in the most efficient manner, and to have files located as close in the hierarchy to core storage as their usage requires and storage capacity permits.

An additional advantage of this dynamic method of storage allocation is the minimizing of the cost of fail-soft capability. Ideally, the failure of any I/O device in the system should not be noticed by the terminal user. Duplexing of all I/O devices provides the ultimate in fail-soft capability. However, full duplexing of large numbers of fast-access storage devices may be prohibitively expensive. This automatic method of keeping only the high-activity files in the fast-access devices tends to reduce the total cost of secondary storage, therefore making full duplexing more attractive.

The basic storage may be duplexed very economically on low-cost, direct access devices such as the 2321. Only the active files are now located in the fast access 2314 or 2311. On the basis of the application characteristics and the installation's desires for rapid response, only a small percentage of the system's storage capacity may be on disk, and it may be quite feasible to duplex these files from an economic standpoint.

IDENTIFYING AND LOCATING DATA

Whenever a user indicates that a new data set is to be created and placed on auxiliary storage, he must

give the data set a name. The name is used when the data is to be retrieved.

In some cases, the name assigned to a data set must be qualified to avoid ambiguity. For example, the qualified names COLOR.CHERRY and TREE.CHERRY describe two different data sets having the simple name CHERRY.

A standard unit of auxiliary storage is called a volume. A volume may be:

A reel of tape
A disk pack
A data cell
A drum

A direct access volume (every one of the above except the tape reel) has a volume label in a standard location. The label specifies the location of a volume table of contents. Each data set stored on the volume has its name, location, organization, and other control information stored in the table of contents. (Similar information is stored in labels of data sets stored on tape.) Thus, if the name of a data set and the volume on which it is stored are made known to the control program, a complete description of the data set, including its location on the volume, can be retrieved. Following this, the data itself can be retrieved, or new data can be added to the data set.

However, keeping track of the volume on which a particular data set resides is a burden, and often a source of error. A provision for cataloging data sets allows the system to do this for the user.

A cataloged data set can be located by the control program, if given only its name. The catalog consists of a series of indexes stored on direct access devices. Each qualifier of a data set name corresponds to one of the indexes in the series. For example, the data set TREE.FRUIT.APPLE is found by searching an index to obtain the location of the index named TREE. The TREE index is searched to find the location of the index named FRUIT. Lastly, this index is searched for APPLE to find the identification of the volume containing the required data set.

The highest level of index in the catalog is transparent to the user. This results from the fact that the system automatically concatenates the user identification to the left of every data set name provided by the user. This highest-level index is referred to as the master index. Using the user identification as the highest index level ensures that each index below the master index can be identified with a user. The collection of indexes below a user identification is called a user catalog. The fully qualified data set name may occupy 44 characters (including periods), of which the user specifies a maximum of 35 characters.

By use of the catalog, collections of data sets that are related by a common external name and the time sequence in which they were cataloged (that is, their generation) can be identified, and are called generation data groups. Thus LAB.PAYROLL (0) refers to the most recent data set of the group, LAB.PAYROLL (-1) refers to the second most recent data set, and so on. In applications that, for example, regularly use the two most recent generations of a group to produce a new generation, the same collection of data set names can be repeatedly used -- with no requirement to know or keep track of the serial numbers of the volumes used.

ORGANIZING DATA

System/360 TSS data sets can be organized in five ways:

- Sequential. This is the familiar tape-like structure, in which records are placed in sequence. Thus, given one record, the "next" record is uniquely determined. The sequential organization is used for all magnetic tapes, and may be selected for direct access devices. Punched tape, punched cards, and printed output are considered to be sequentially organized.
- Indexed sequential. Records are arranged in logical sequence (according to a key that is part of every record) on the tracks of a direct access device. In addition, a separate index or set of indexes maintained by the system gives the location of certain principal records. This permits direct as well as sequential access to any record. Records may be added to and deleted from the data set as required, with appropriate updating of the index.
- Partitioned. This structure has characteristics of both the sequential and indexed sequential organizations. Independent groups of sequentially organized data, each called a member, are in direct access storage. Each member has a simple name stored in a directory that is part of the data set and contains the location of that member's starting point. An example of partitioned data set use is the storage of programs; as a result, partitioned data sets are often referred to as libraries.
- Telecommunications. This organization deals exclusively with data going to or coming from remote online terminals. Such data (messages) may be processed directly from main storage or from queues in direct access storage.
- Graphics. This organization is oriented to data going to or coming from online graphic display units.

STORING AND RETRIEVING DATA

Data management includes the following access methods to simplify storing and retrieving data:

- Sequential access methods. Basic Sequential Access Method (BSAM) and Queued Sequential Access Method (QSAM) are compatible at the source language level with OS/360. The interchange of data between OS/360 and TSS/360 is facilitated by this compatibility.

  BSAM furnishes device control without automatic buffering and blocking. I/O operations are scheduled at the time they are requested. Characteristically, the macro instructions READ and WRITE retrieve and store entire physical blocks of data.

  QSAM is designed to furnish a full range of buffering and blocking facilities with maximum programming simplicity. It applies only to organizations with sequential characteristics. The macro instructions GET and PUT are used for retrieval and storage of logical records. Records may be fixed length (blocked or unblocked), variable length (blocked or unblocked) or unspecified.
- Terminal access method. TAM provides the necessary communication to be established with a terminal. Routines are included to cause the proper action to be taken upon the termination of any channel program operating a low-speed terminal device.
- Virtual access method. VAM is specially designed for TSS/360, and it effects the intermediate input/output of data by interfacing with the paging supervisor. VAM data sets are limited to direct access devices. Although physical blocks are in units of pages, the logical record length can be as long as 2^{20} bytes, and records will be packed into and across "pages" as required. VAM allows the user to treat a direct access data set, or any portion of it, as part of his virtual memory. Three data set organization methods are available under VAM: sequential, indexed sequential, and partitioned. Record formats are fixed length, variable length, or unspecified.
- Graphic access method. GAM provides software services for the 2250, 2280, 2281, and 2282. It generates graphic orders for the control of these graphic devices, facilitates data handling both in main storage and in the graphic device buffer, accomplishes I/O control functions, and controls the dispatching of asynchronous light pen, alphameric keyboard, program function keyboard, attention and error interrupts. Embodied in GAM is a graphic interrupt supervisor, which queues and distributes graphic interrupt conditions. GAM is also responsible for the management of graphic device buffer storage.

SYSTEM FACILITIES FOR PROGRAM CONSTRUCTION

Virtual Memory Concept

The 24-bit addressing capability of the System/360 permits twelve-bit base addressing (4096 base addresses) and twelve-bit byte addressing (4096 byte addresses) -- a maximum of 16 million addressable bytes. In programming, however, the user is generally restricted to addresses that represent physical storage on his machine. Such a program cannot address 16 million contiguous bytes directly, but must be structured as a series of overlays.

TSS/360 permits the concept of a "virtual memory" whose size is the maximum addressing capability of the computer system. For the System/360 Model 67, which has been modified for 32-bit addressing, the capacity of virtual memory is over 4 billion bytes.

The time-sharing supervisor has the job of assigning active programs to whatever physical storage is available. Automatic relocation techniques are used to associate the logical program (after processing by the dynamic loader) with physical locations consistent with the efficient operation of the system. This physical fragmentation need never concern the programmer. He may write his program as if contiguous bytes of storage were available for each assembly.

Basic Functions

Processing by the language processors involves the translation of source statements into machine language, the assignment of virtual memory locations to instructions and other elements of the program, and the performance of auxiliary functions designated by the programmer. Output of the language processors is the object program module, a machine language equivalent of the source program. The program furnishes a printed listing of the source statements and object program statements and additional information useful to the programmer in analyzing his program, such as error indications. The object program is in the format required by the linkage and loading components of TSS/360.

The language processors use virtual memory for the allocation of working storage. Thus, language elements customarily limited by the capacity of internal tables are, in effect, without limits in TSS.

Program Sectioning and Linking

It is often convenient, or necessary, to write a large program in sections. The sections may be separately assembled or compiled, then combined subsequently into one object program. The language translators provide facilities for creating multi-sectioned programs and symbolically referencing separately compiled programs or program sections.

The concept of program sectioning is a consideration at coding time, compile time, and load time. To the programmer, a program is a logical unit. He may want to divide it into sections called control sections; if so, he writes it in such a way that control passes properly from one section to another regardless of the relative physical position of the sections in storage. A control section is a block of coding whose virtual memory location assignments can be adjusted, independently of other coding, at linkage or load time without altering or impairing the operating logic of the program. A control section is normally identified by the CSECT, PSECT, or COM assembler instructions. An unsectioned program is treated as a single control section. To the dynamic loader, there are no programs, only control sections that must be fashioned into an object program. The linkage editor may be used to combine separately produced modules, if desired, before dynamic loading.

The output of a language translation is an object program module, consisting of one or more control sections and a control dictionary. The control dictionary contains information that the linkage editor and the loader need in order to complete cross-referencing between control sections. The linkage editor and the loader can take control sections from various assemblies and compilations and combine them properly with the help of the corresponding control dictionaries. Successful combination of separately produced control sections depends on the techniques used to provide symbolic linkages between the control sections.

Regardless of the degree to which his program is sectioned, the programmer still knows the elements that make up his virtual memory, because he has described them symbolically. He cannot, however, make any assumptions about the position or ordering of control sections, since their virtual memory location assignments may have been adjusted by the linkage editor and/or the loader, and their physical storage addresses may be constantly changing within the time-sharing environment.

Control Sections

CSECT -- Control Section

The CSECT identifies the beginning of a control section. If a symbol names the CSECT instruction, the symbol is established as the name of the control section; otherwise the section is considered to be unnamed. All statements following the CSECT are assembled as part of that control section until a statement identifying a different control section is encountered. The text of each control section starts on a page boundary, and a virtual memory page table is constructed as the text is produced.

PSECT -- Prototype Control Section

Within TSS a single copy of a commonly used reentrant routine appears to have different virtual memory location assignments to different users, although its actual disposition in storage remains unchanged. When control is transferred to a reentrant routine, the calling program must supply an address constant that reflects the virtual memory assignments of the calling program in order that the reentrant routine may obtain data storage unique to the user.

This would ordinarily imply that a program which calls a reentrant routine be knowledgeable about all address constants that might be required within the hierarchy of reentrant programs. To minimize this clerical burden, a prototype control section is defined for use by reentrant programs to simplify the handling of address constants and working storage.

On linkage to the reentrant routine, a copy of the contents of the prototype section is made and assigned virtual memory locations within the domain of the calling program.

In reentrant program all working storage and address constants are assembled within a prototype section; the user need not know any of the internal requirements of the routine he calls.

Communication of prototype section information is accomplished through the use of the R-type address constant. This supplies the location of the control section that is required by the reentrant program for working storage and address constants unique to the calling program. Use of the R-type address constant causes the loader to supply the location of the prototype section as a function of the entry point name.

COM -- Common Control Sections

The COM assembler instruction identifies and reserves common areas of storage that may be referred to by independent assemblies or compilations that have been linked and/or loaded for execution as one overall program.

One blank common section and any number of named common control sections can be designated in an assembly or compilation.

No instructions or constants are assembled in the "blank" common control section. Data can be

placed there only through execution of the program. Instructions and constants, however, can be assembled in "named" common control sections. The rules governing the final structure of common control sections are described in the linkage editor section.

Attributes of Control Sections

To facilitate dynamic linkage and loading within TSS, it is often necessary to indicate that certain attributes are characteristic of the data or instructions within a control section. One or more of the following operands may be used in CSECT, PSECT, or COM statements to indicate which attributes are to be assigned to the section:

- PUBLIC - Indicates that the section contains shared, public data or instructions.
- READ-ONLY - Indicates that the section contains instructions or data which are never modified.
- VARIABLE - Indicates that the length of the section may vary during program execution.
- REENTRANT - Indicates that the section contains instructions that may be reexecuted at any point through interruption procedures.

Attributes may be specified singly or in combination, where meaningful.

Symbolic Linkages

Symbols may be defined in one program and referred to in another, thus effecting symbolic linkages between independently assembled programs. The linkages can be effected only if the language processor is able to provide information about the linkage symbols to the dynamic loader that resolves these linkage references at load time. The assembler or compiler places the necessary information in the control dictionary on the basis of the linkage symbols identified by the ENTRY and EXTRN statements. Note that these symbolic linkages describe linkages between independently compiled control sections.

In the program where the linkage symbol is defined (that is, used as a name), it must also be identified to the language processor by means of an ENTRY statement. It is identified as a symbol that names an entry point, which means that another program will use that symbol in order to effect a branch operation or a data reference. This information is placed in the control dictionary.

Similarly, the program that uses a symbol defined in some other program must identify it by an EXTRN statement. It is identified as an externally defined symbol (that is, a symbol defined in another program) that is used to effect linkage to the point of definition. This kind of information is also in the control dictionary.

USE OF THE LANGUAGE PROCESSORS

Invoking the Language Processor

The desired language processor is called by a Language Processor Control (LPC) program, which functions as intermediary between the language processor and the remainder of the system. Its principal duties are to invoke the appropriate program, receive source language statements from the terminal (or other system input device), and route diagnostic messages from the language processor to the terminal (or other system output device). Thus the Assembler FORTRAN IV or PL/I may be used in either conversational or nonconversational mode.

Source statements for the processors are contained in a line data set with each item sequentially numbered. Such a data set may be cataloged from an external source by a service routine, or may be entered directly at a terminal keyboard. It is the function of LPC to supply line-image items to the language processor, one at a time, upon request. The source language line itself remains in the virtual memory assigned to the LPC for the duration of the translation process. Note that both the LPC program and the language processor are part of the user's virtual memory.

The sequence order of the lines is maintained through a series of address linkages that permit the accommodation of any additions, deletions, or changes that the terminal user may introduce. In supplying source lines to the language processor, LPC follows the address linkages to ensure that the processor receives the statements in sequential order. This order forms the basis for all subsequent processing.

Output from the Assembler

The machine-language output produced by the Assembler is distributed between a text module and a control module; together they make up the object program module. The text module contains machine language instructions; the virtual memory pages in the text module will ultimately map onto pages in the target virtual memory of the assembled program.

The control module contains information about external symbol definitions, external symbol references, and the relocation properties of the text, all organized by control section.

The listing module contains a line-image listing suitable for printing on an external device.

Two optional facilities are available. One of these prepares a cross-reference listing from information generated for this purpose. The other collects information from the internal symbol and using-register tables and constructs an output data set suitable for later use by the program checkout system. This module contains a symbol table for all pertinent internal symbols, in addition to information about the ranges of virtual locations over which specified using-registers are effective. Preparation of this module and the cross-reference listing is controlled by user-supplied parameters.

Output from the FORTRAN IV Compiler

The FORTRAN IV compiler is designed to produce compact, efficient object programs in a form suitable for loading and executing by the TSS/360 system. All object program modules (including "main" programs) are relocatable, reentrant subroutines, which may optionally be processed by the linkage editor or loaded directly through the facilities of the dynamic loader. The output module is organized into several control sections, each of which has a dictionary part and an optional text part. The text of each control section starts on a page boundary, and a virtual memory page table is constructed as the text is edited.

Each output module contains at least two control sections. A prototype control section (PSECT) is always present. Other control sections, their presence determined by the categories of source statements, may be:

1. Code control section (CSECT)
2. Control section for blank COMMON
3. A number of control sections representing named COMMON areas

The PSECT is a fixed length, prototype control section required by reentrant routines that contains all the address constants in the module. The dictionary part includes external references and definitions as well as relocation information for address constants.

The CSECT is a fixed length, read-only, reentrant control section. Its text part consists of the object code, including numeric and alphameric constants. The single exception is a module that represents a BLOCK DATA subprogram, where text may be generated. Named COMMON sections are fixed length, but blank COMMON is a variable length control section.

Documentation of Object Programs

A program list data set is generated whenever the listing is requested. This listing is a representation, in Assembler-like form, of storage classes within the object program module.

The user may also request optional output in the form of a memory map, a symbol table, or a cross-reference symbol table. These will appear on the same data set generated when the object program listing is specified.

These optional outputs may provide static information useful in analyzing the source program. For dynamic information regarding the flow of the program, the user has access to FORTRAN facilities and may also request the services of the program checkout system, described elsewhere in this section of the manual.

LINKAGE EDITOR

The linkage editor is a service program, optionally used in association with the language translators.

The object program output of a single compilation or assembly is called a program module or simply a module. Normally, this single module is input directly to the dynamic linkage loader, and all other modules necessary to form a total program are dynamically linked together. However, the user may find it desirable to process one or more program modules through the linkage editor before employing the loader.

The linkage editor performs the following functions as specified by the user:

1. Two or more program modules, created by the same or different language processors, may be statically linked together to form one program module. A statically linked program requires somewhat less loader processing time and makes more efficient use of virtual memory (at the cost of linkage editor processing time).
2. Control sections within modules may be replaced, deleted, or renamed.
3. Entry names and external references may be renamed.
4. Entry names may be deleted.
5. A new module entry point (transfer point) may be defined.
6. The attributes of control sections may be changed.
7. Two or more control sections may be combined into a single control section.

The output module from the linkage editor may be executed or may be cataloged for later use. (Also, the output module may later be processed again as input to the linkage editor.)

Processing by the linkage editor is governed by statements arriving from a remote terminal device (conversational mode) or the primary input device (nonconversational mode). Users at the terminals may correct control statement errors detected by the linkage editor.

Subroutines from a user library or the system library can be included in the output module at a specified point in the program. However, if it is desirable that this type of routine be included dynamically at execution time, a LIBRARY statement to the linkage editor creates tables for the dynamic loader, facilitating the loading of these routines at execution time. Thus, user libraries need not contain standard system subroutines, and if the routine is sharable, it is possible that the copy linked to is already in core storage.

Output from the linkage editor is the same as the output from any language processor and is in a form processable by the dynamic loader. The output module consists of two parts: the dictionary and the text. The dictionary contains all the information necessary to process and load the text, including information about all control sections within the module, relocation information, and the initial entry point to the program. Also supplied is information pertaining to user library requirements and the total blank COMMON requirements of the output module. If desired, the output module may be cataloged for continued usage or may optionally be link-edited with additional program modules.

DYNAMIC LOADER

The dynamic loader allocates storage and links external symbols among various separately assembled or compiled program modules on a dynamic basis. A module is only linked into the user's executing program when a need for it becomes apparent.

The dynamic loader consists of three independent routines:

1. The explicit linkage routine is called by the user through a Supervisor Call (SVC). The user provides a name as input to the routine. The loader finds the module containing that name either as the module name or external definition. If the module has not already been processed, information about the module is placed in the loader tables, virtual memory addresses are assigned to its various control sections, and page tables are constructed for the module. If any relocation is necessary, the pages are marked "unavailable" and "unprocessed by loader".

2. The implicit linkage routine is called when a "page unavailable" interrupt occurs because of a user program reference to a page for which the loader has constructed a page table entry but has not performed the relocation of address constants. This routine relocates the address constants in the referenced page. External references to new modules from the page cause the loader to "link" them in a manner similar to that described above for an explicit linkage call.

3. The explicit unlinkage routine, a facility that is made available for the unlinking of modules, is entered through an SVC, which causes the loader to delete all table entries for the module specified by name.

Control sections are allocated virtual memory in the following way: Standard control sections of specified length are allocated a fixed number of pages. To handle facilities like blank COMMON, where the length depends on the longest length declared in any module already loaded, a variable number of pages are made available as required. Prototype control sections are handled similarly to standard control sections. However, when the prototype section is associated with a shared public routine, its external page table is saved, and a fresh copy made for each user of the routine.

The dynamic loader is one of a subset of service routines initialized into the user's virtual memory at LOGON time. It is available for use by other system routines and by the user, in either conversational or nonconversational mode. With this facility the user need not use the linkage editor's services between creation of program modules by a language translator and execution of the job. This is especially appealing when a given run of a program may cause only one of a number of independent routines to be required. That particular routine may then be explicitly called at execution time, eliminating the requirement for linking the unused routines.

PROGRAM CHECKOUT SYSTEM

The program checkout system is an extension of the command language system. The command language recognizes program checkout statements and passes them on to the appropriate subroutines for interpretation and response. These routines are reentrant (and are part of the user's virtual memory). Under PCS the user can display and change the value of a variable, alter the normal sequence of program execution, terminate execution, dump, etc. These facilities are available in conversational mode as well as nonconversational mode.

The complete services of the program checkout system are available after the user program is loaded. After the command for loading is recognized by the command language, any PCS statement may be input to set up debugging conditions before control is transferred to the program. During execution of the program, the user may dynamically interact with his program by depressing the attention mechanism, which causes the object program to stop and terminal communication to revert to command mode. The user may input program checkout and/or command statements as required, and then cause the program to resume execution.

At termination of the object program, the terminal is again returned to command mode, and again, program checkout statements or commands are recognized. Program checkout operations, including a restart of the program from a specified entry point, can be continued as desired.

The utility of the program checkout system is maximized when a complete symbolic description of the object program, expressed in source language symbols, is available. This is possible by retaining the Internal Symbol Dictionary provided optionally by the Assembler and FORTRAN compiler. If this dictionary is not available, the user is restricted to referencing hardware registers and external symbols in his debugging statements.

SYSTEM DESIGN CONSIDERATIONS

Supervisor Functions

The time-sharing supervisor controls the execution of jobs entering the system and the hardware environment in which they operate. Many features of TSS have been specially designed to facilitate multiprogramming, time sharing, and multiprocessing. These special features are fully exploited by the time-sharing supervisor to increase system throughput, minimize response time, and efficiently utilize available system components.

Basically, it can be said that the supervisor's function is to respond to interrupts. These interrupts must then be sorted as to type and function and a program must be executed to respond to them.

When an interrupt occurs, it is processed before control is returned to the interrupted program. Since the System/360 has facilities for masking or allowing different types of interrupts, however, and differentiates among five different classes of interrupts, each interrupt routine is normally disabled only for an interrupt of its own class. If a second interrupt of the same class occurs before processing of the first is complete, the second (and all later interrupts) will be stacked. As one is completed, another is removed from the stack and processed.

It is possible to define levels of priority between interrupts and allocate portions of storage to certain interrupt routines in such a way as to ensure that specified terminal actions receive the quickest possible response.

The supervisor programs that respond to interrupts can be classified into four general categories:

- Processor control -- the scheduling of processor time to user programs and supervisor functions
- Storage control -- the allocation of both primary and secondary storage space, including address translation and storage protection
- I/O control -- the assignment of I/O devices to both user programs and supervisor functions, and the scheduling of data paths between storage and I/O devices
- System control -- the logical control of all of the above functions together with operator communication, partitioning, and recovery functions

All of these supervisor programs use status data organized into tables or indices that are continuously sampled, changed, and recorded by the control programs.

Task Status Index

The most important index in the supervisor is the task status index. (TSI). There is a separate TSI for each unit of work or task in the system. TSI's may be either dormant or active. Dormant TSI's are in this state when there is no user currently at the particular terminal, or when the terminal has an exceptionally long wait. Active TSI's contain status information on jobs currently in some state of execution.

Because of the magnitude of information that it must contain, the TSI is split into two sections. The resident TSI contains the user-oriented information, which must be in core at all times. The rest of the information is kept in an extended TSI and is eligible to be swapped out of core along with its associated user programs and data.

Multiprogramming

The basic purpose of a time-sharing system is to provide rapid and complete service to a number of users concurrently. One of the chief functions of the time-sharing monitor is the rapid assignment of the processor facilities on a rotating basis to all active users in the system. Ideally this commutating of processor facilities is so rapid that it appears to each user as though he has all the facilities of the system at his complete disposal, and that he is the sole user of the entire system.

The large storage capacity of System/360, both in terms of physical storage capacity and direct addressing capability, is an obvious aid to efficient time sharing. The more users' programs that can be held in directly addressable core, the less system time required in transferring to them, and the more immediate the response.

When the number of users and the size of their jobs expand, the capacity of directly addressable core is exceeded. The monitor must then resort to paging or storing portions of momentarily inactive users' programs and data on a secondary storage device to make room for the loading of program and data for a currently active user. The high data rate

(1.2 megabytes per second) and the low access time of the IBM 2301 drum make it particularly desirable for this purpose.

This access time, while extremely low for a secondary storage device, is nonetheless several orders of magnitude greater than the time to access core storage. Efficient utilization of the processor requires the supervisor to multiprogram among a subset of active users' programs that are already resident in core during the time a requested non-resident program segment is retrieved. Therefore, the time-sharing supervisor fully supports multiprogramming. Advanced methods of storage allocation, storage utilization, and statistical data gathering techniques are used to this end.

Multiprocessing

The time-sharing supervisor also supports multiprocessing. The addition of more processors, as well as multiple data paths between many I/O devices and multiple core storage units, requires an added dimension of control within the supervisor. This added control capability is of a very general nature so that additional processors, channels, I/O devices, and primary storage units can be easily added to the system.

Dynamic Program Linkage

The dynamic linkage of programs and subroutines is permitted by the time-sharing supervisor. Implementation of this powerful programming aid is greatly simplified by the address translation feature of the IBM 2067 Processing Unit. This allows the dynamic loading of program or data with minimal virtual address modification and facilitates shared use of program and data segments. The supervisor identifies a program module by name and resolves symbolic linkages on call. Subsequent references to the program are fast and direct. The net result is an ability to dynamically call programs and subroutines and, once loaded, have them tightly and efficiently linked.

Scheduling

The scheduling of jobs in a time-sharing system is somewhat more complex than in a simple batch-processing system. Many good arguments can be advanced for various scheduling algorithms. Ultimately, only continued usage of the time-sharing system in the user's own environment will determine what scheduling algorithm is best adapted to the particular user. For this reason the scheduling routines in the time-sharing monitor are both flexible and replaceable. Experience is likely to dictate modifications.

The scheduling algorithm to be supplied by IBM is planned as follows:

Assume that there are no interrupts to be processed. Core storage contains a chain of TSI's representing the current set of tasks that are attemping to time-share the system. The simple, straightforward approach would be to assign each task a quantum of time on a processor. Scheduling then reduces to a simple commutating around the TSI chain, allotting each TSI its full quantum of time. This would work well if all users' programs could be kept in core at one time and if each user could effectively use the processor for his full quantum.

Actually, however, this ideal situation is rarely encountered. Users may request I/O operations during their time-slice (quantum) and then await their completion. Core storage limitations may force the supervisor to roll out portions of core to make room for another user's program and data (page turning). A task may terminate. Efficient utilization of the processor dictates that some of these conditions terminate execution of a task before expiration of its time-slice. Page turning and its associated delays in gaining access to secondary storage force the supervisor into a situation of multiprogramming among a small subset of TSI's. To provide better orderly procession around the TSI chain (and thereby, reasonably proportionate distribution of system capabilities among TSI's), a commutator is defined, marking the TSI that should be executed next if at all possible. When this TSI's time-slice ends (for any reason), the commutator moves on to the next TSI.

Since multiprogramming within the time-sharing supervisor is required to properly utilize the system, a dispatcher is defined. Whenever a processor needs work, it goes to the dispatcher. The dispatcher then looks at the TSI currently pointed to by the commutator. If the TSI is in a ready state, it is dispatched to the currently available processor for execution; if not, the dispatcher searches through the TSI chain for the next ready TSI. This task (which is not pointed to by the commutator) is executed until an interrupt occurs.

Since the dispatcher always starts looking for work at the commutator, the general effect will be a procession of the commutator around the TSI chain, with the dispatcher frequently scampering ahead of the commutator and then returning in an attempt to keep the processor busy.

The system may be regarded conceptually as a set of facilities to which tasks are queued for service. As a task is processed by the system, it moves from facility to facility.

Thus, the active list of TSI's is a task queue for the CPU facility. In a multiprocessed configuration, this facility may consist of several CPU's. (To the

time-sharing monitor, this presents no serious problem; an interlock mechanism, using Test and Set instructions, prevents a task from being given control by two or more CPU's simultaneously.)

In a given time-slice when a task program reaches a "page wait", it is necessary to enter a task in a queue in order to obtain an area of core storage that will contain the page currently residing on a storage medium (2301, 2311, 2314). When this request for core storage has been serviced, the task is moved to a queue on the appropriate paging device to read the page into core. After the page read service is completed, the task is moved back to the CPU facility queue.

It is quite possible, and probably desirable, to maintain more than one TSI chain. Many tasks may be completed in a short quantum of time. Other tasks may require much longer execution periods. To best service the varied requirements of terminal users, it may be desirable to allot frequent time-slices of short duration to the shorter-duration users, while allotting less frequent slices of longer duration to the users of larger amounts of processor time. The commutator can then work its way around the short-duration chain with less frequent excursions to a chain or chains of slower-duration users.

The quanta of time allotted to longer- or shorter-duration TSI's is an installation option, as is the number of TSI chains. It can be varied by the user to tailor the system response as required. Indeed, the whole scheduling algorithm can be replaced by one that best serves the user's needs.

Storage Allocation

The supervisor permanently occupies about 100K bytes of physical core and runs nonrelocated. The balance of memory serves as a pool of 4096-byte blocks (called pages) for those tasks currently being multiprogrammed.

An index within the supervisor shows the current allocation of each block of core storage. Every block is either free or assigned to an actual job. Since every job has known priorities and core requirements, the algorithms for primary (core) and secondary (drum and disk) storage allocation and swapping can be designed to meet specific user requirements.

When a new user is logged on to the system, he is given a minimal set of "system service routines" in his otherwise empty virtual memory, so that he can strike up an effective conversation. This set includes:

1. The command language interpreter to interpret users' commands
2. The dynamic loader to load requested programs (and, if required, to unload them)
3. Catalog services, to locate data sets
4. Virtual memory allocation, for assigning space in the task's virtual memory to programs and data

As programs and data are dynamically called from secondary storage, the supervisor must allocate space for them within core and on the swapping drum. A request may have come from the user at a terminal, from a user's program, or through the address translation process finding a page that is not in core. The supervisor then must choose a block in primary storage to receive the block from secondary storage. The chosen primary block may be empty, either because it has not been allocated or because the task to which it had previously been allocated has terminated. Alternatively, it may contain valid data. If so, one of two conditions may occur: the contents may have been changed since the block was last fetched from secondary storage, or they may not have changed. If the contents have changed, the block must be swapped out to the drum. If not, the block may be overwritten with impunity, since a copy of it is available from secondary storage. Determination of usage and change can be made from the reference and change bits in the storage protection key.

The first choice of the allocation routine is from a list of free blocks. The second choice is from a list of blocks currently assigned to tasks that are for the moment inactive (for example, waiting for a user response). The third choice is from a set of blocks currently assigned to tasks having a lower priority than that of the task making the storage claim.

If all core storage blocks are assigned to tasks of higher priority than the task making the claim, allocation is deferred until a higher-priority task terminates or is removed from the top of the queue for other reasons.

If the claiming task is the highest-priority task in the queue and storage is still not available, it will need to overwrite itself. In this case, blocks will be assigned which have not been recently used and which do not contain the currently executed instructions. Program index blocks and I/O blocks currently queued for service will not be overwritten.

Normally, the demands of core storage can be satisfied by the first two choices made by the allocation routine. Severe partitioning, together with high terminal activity can, of course, combine to force the allocation routine into further choices and lengthen the time devoted to allocation.

Reentrant Coding

Efficient utilization of primary storage is further enhanced by the supervisor's use and management of

reentrant code. Blocks of reentrant code are so indicated in the storage dictionary. Additional copies, with their attendant waste of core storage, are no longer required. Since reentrant blocks are also read only, they do not have to be written out on the drum when page swapping occurs, thus reducing system overhead.

Statistical Data Gathering

The statistical data-gathering capabilities of the time-sharing supervisor include both job accounting and system performance information. Job accounting information includes such items as name, project number, subproject number, number of times the job entered and exited from the system, elapsed time, total processor time, and space utilized. System statistics are gathered on an optional performance basis, since the data produced is so voluminous. These statistics include data on utilization of system components and data paths. It is anticipated that this type of statistical data gathering will be done on a less frequent basis to optimize the storage allocation and job scheduling algorithms.

From an accounting viewpoint, a table of statistics is established for the user at "log on" time and transferred at "log off" to a master set of accumulated statistics for the particular user within the charge number under which he is working. The user may command the system to display his accumulated charge.

Static statistics are also maintained for each data set cataloged in the system -- for example, length of data set and length of time during which the data set is cataloged. These statistics are accumulated under the charge number of the creating user.

Standard I/O Retry Routines

When a machine error is determined by CPU hardware, a machine check interrupt occurs in that CPU and this same signal is broadcast to all other CPU's in the system, which receive such indications as malfunction (external) interrupts.

The original CPU is put in "wait state", with interrupts masked, thus preventing it from disrupting the total system. One of the other CPU's in the system accepts the associated malfunction alert, the others going into the wait state. It is the function of the active CPU via the "recovery nucleus" to identify the failing unit in order to remove it from the active system.

When a less disastrous fault occurs in the system, such as failure to read a record correctly from a storage device, the time-sharing supervisor invokes a standard retry routine. This routine repeats the operation a predetermined number of times to attempt to read the record. If this retry routine fails to read the record correctly, it reports this information to the time-sharing supervisor. The supervisor logs this information and then calls for a system error analysis program. This program, which is treated by the supervisor as a user program, decides which units are to be eliminated from the resource table in the supervisor.

The decision as to which unit or units to drop from the resource table is made by examining the recorded error environment information, then determining the logical partitioning that would have the least impact on system performance. For example, when a fault occurs in a unit that has two data paths, the system error analysis program analyzes the fault to determine whether one of the data paths or the unit itself should be eliminated from the resource table. The program does not eliminate an operational unit from a resource table if there is at least one data path to that unit.

Messages are sent to the operator when a data path is eliminated from the resource table, but no maintenance action is taken until all data paths to the unit are out or until the customer engineer and operator decide that maintenance is required. At this time, the customer engineer calls for the diagnostic monitor and begins the diagnostic procedures.

Partitioning

The logical control of the system configuration is the responsibility of the time-sharing supervisor. System components are allocated by the supervisor to the processing of tasks as they are required. When a task is completed, the components that were assigned to that task are returned to the supervisor for use in the processing of other tasks.

In addition to this logical assignment of system components, the supervisor also considers physical partitioning. A component may be physically detached from the supervisor by means of switches, in order that the component may become part of another system or be made available for maintenance. Regardless of the type of partitioning required, the supervisor maintains control so that system operation is not disrupted. To accomplish this, the supervisor keeps track of the condition of the system components.

At any time, the machine room operator can enter a privileged command that will add or delete a unit from the device index. When equipment is leaving a partition, the supervisor examines the circumstances with regard to potential conflict or interferences. The supervisor must not, for example, release the core that is taking processor interrupts, until an alternate area has been initialized.

In general, when a device is to be subtracted from a list, the supervisor must arrange to withdraw from that unit.

PERFORMANCE ANALYSIS

In nonrelocation mode the Model 67 behaves like a standard System/360 and will run any programs that will run on a standard System/360 having the same storage and I/O complement available on the Model 67 (or that portion of a dual system that is partitioned to operate as a non-time-shared system). System performance in nonrelocation mode will approximate that of a Model 65 degraded only by storage priority and cable-length delays present in the particular system.

A quantitative measure of systems performance is always a function of the application environment within which the system is operating. There are several factors which must be given consideration when the system is operating in relocate mode under TSS.

In conventional operating systems a certain amount of time is lost because of inability to completely overlap I/O and computing. While the same situation may exist in a system operating with a page-oriented virtual memory concept, the probability of having a program ready to run is higher — hence a greater potential for overlapping I/O and computing.

A time-shared system spends a certain fraction of its time in the function of page turning, which is nonproductive systems time. The analogous function in conventional operating systems is that of programmed overlay. Overlaying, while it is potentially more efficient, suffers from the fact that it imposes an additional chore on the programmer; in addition, it is frequently inefficient because of insufficient detailed knowledge of how the program will actually run.

The performance improvement to be gained from the functional characteristics of the system must be estimated for each individual environment and are not included in this section.

Specific hardware performance factors, on the other hand, tend to be less application-dependent. Relocation timing and shared storage interference and delays can be estimated with reasonable precision without specific knowledge of the environment. These timings are included, but, unless balanced by the associated functional performance improvements, do not present an accurate appraisal of the IBM Model 67 Time Sharing System.

RELOCATION TIMING

The basic processor operates at a five-megacycle (200-nanosecond cycle) rate. When the relocation action is active for each memory reference, the clock is stopped for 150 nanoseconds to allow for the associative compare. When the relocation action is active and no valid associative compare occurs, the processor clock remains blocked until the 16-bit page table entry is fetched from main storage and loaded into one of the associative registers. The length of time the processor clock is blocked depends on the storage speed. Using the IBM 2365 Processor Storage (750 nanoseconds), the total elapsed time is 2.1 microseconds.

An IBM 7090 program has been written that interpretively executes 7090 problem programs. Each effective address generated by the problem program is explicitly computed and examined in a model associative register (AR)/storage address list facility. Various parameters are possible -- for example, the number of words in the AR, the size of the block, and the replacement algorithm. The program counts the number of times the model AR's must be loaded and the number of references made. It makes two additional measurements: (1) the total number of blocks claimed by the problem program, and (2) the number of blocks actually used by the problem program. Among many replacement algorithms tested is one similar to that implemented in the System/360 hardware. A variety of significantly different programs were processed repeatedly, with variation in the parameters.

All the following remarks must be considered when applying the results to System/360 operation.

The problem program and the simulation program must fit in the 32K 7090 store. The simulator program is approximately 4K words long, so that no problem program claiming more than 28K words of store would be run. In particular, no program equivalent to a compiler is processed.

The ratio of AR loads to AR references is called activity. The accuracy of the reference count is important, but can be extrapolated only to System/360 counts. In the load count, the simulator program does not retain the physical store instruction counter location in a separate register. It is treated like other references so that, in fact, there are only seven registers available for data coverage. Moreover, the reference count includes all instruction fetches. The data fetches reflected in the results are IBM 7094 fetches, and 7094 code produces significantly more data fetches than System/360 code. It is felt that these two factors are inaccurate in approximately the same proportion, and tend to balance out.

At any instant in time, a processor may be in one of three states:

1. Running in a relocation mode
2. Running out of the relocation mode
3. Idle (awaiting I/O, no work, etc.)

The estimated overhead applies only to state 1.

It is here that processor work is done for users. There should be a disproportionate and favorable decrease in time spent in states 2 and 3 above, which, to a great degree, reflect system overhead and swap time.

With an eight-register AR using the indicated replacement algorithm, and block size of 4096 bytes, the activity was approximately zero (see Figure 7 which shows activity versus block size). However, an activity of 5% is assumed in the following calculation, since program analysis was restricted to programs of less than 28K and there will probably be a tendency to utilize more shared routines in time-sharing problem programs.

A scientific mix of instructions whose run time on the Model 67 in unrelocated mode is 140 microseconds required 110 storage references. Of the 110 storage references, 25 were branches, 50 were operand fetches, and 35 were other types.

The following calculation provides us with the run time of this mix on the Model 67:

$$140 + [75] \; [.15 + (.05)\; 2.1] = 160 \text{ microseconds}$$

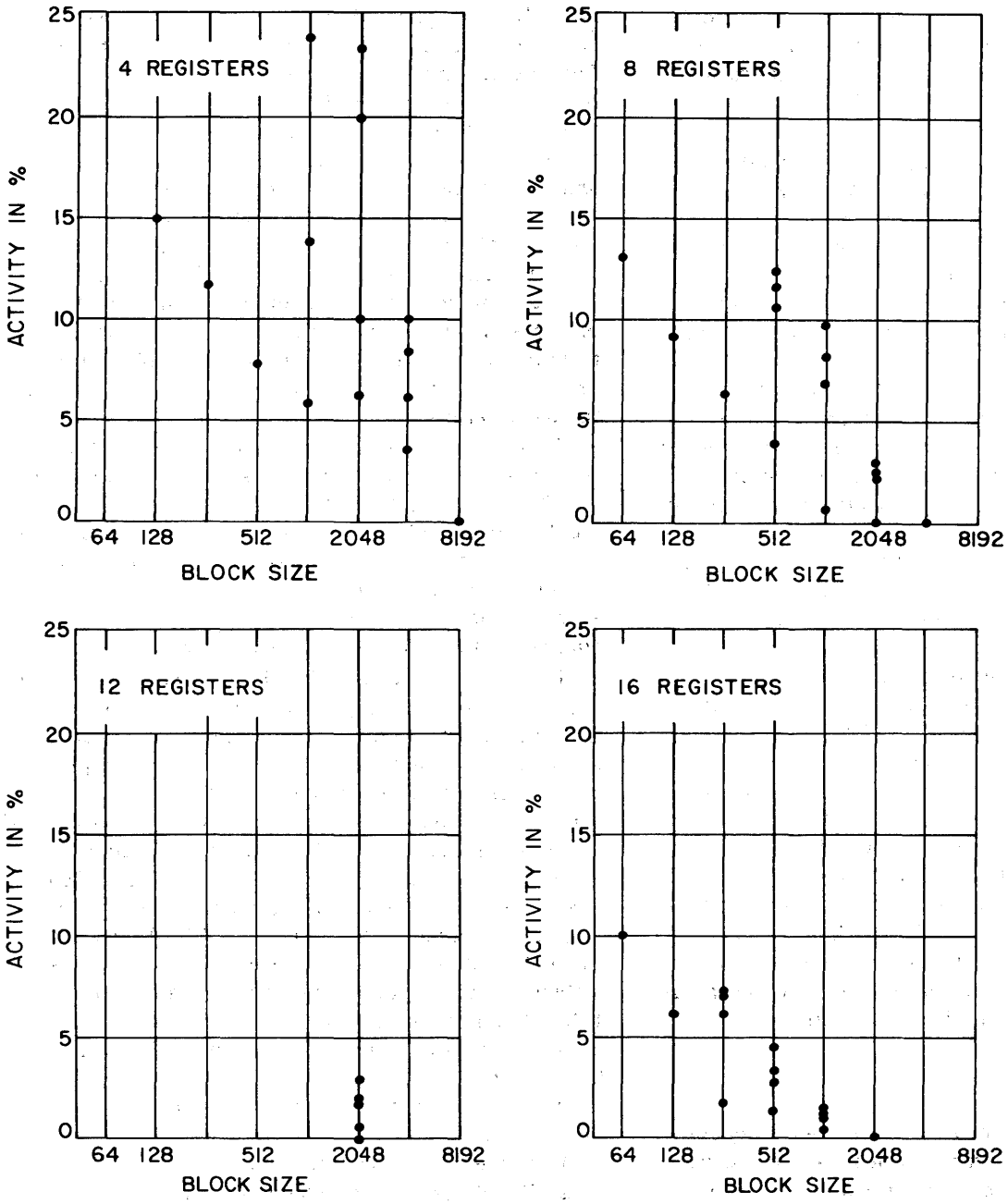


Figure 7. Activity vs block size (in bytes) for 4, 8, 12, and 16 registers

The resulting run time has been extended by about 14%. However, if the activity is zero, the run time is increased by only 8%.

SHARED STORAGE INTERFERENCE AND DELAYS

This section deals with the effect on internal processor performance of those interferences and delays characteristic of multiprocessor shared multiple storage configurations.

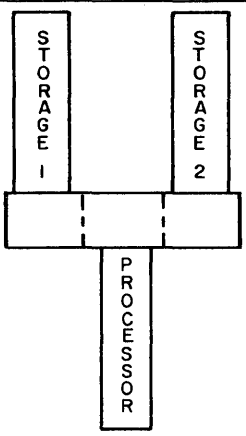
A timing diagram was generated that shows the storage reference demand made by a Model 67 when executing what is considered a typical scientific mix of instructions. From the timing diagram a cycle demand distribution was obtained and used in generating the cycle demand for the simulated processors modeled in GPSS III.

A simulation run was made with only one processor executing the instruction mix and without any cable or priority determination delay. This run was used as the base run to which all other modeled configurations are compared. System degradation is defined as 1 minus system efficiency, where system efficiency equals the ratio of job run time in the base system to job run time in a processor of the system to be compared. Job run time was chosen to be that time required for the CPU to obtain 20,000 memory cycles executing the instruction mix.

For each processor in the model, one transaction is generated which flows through the model at a rate that is a function of its unrestricted (desired) memory cycle demand and the delay encountered because of interference by memory demands from other devices, as well as the cable and priority scan delay. All memories in the model have an equal chance of being selected, regardless of the device making the request or the number of other devices contending for the storage unit at that time.

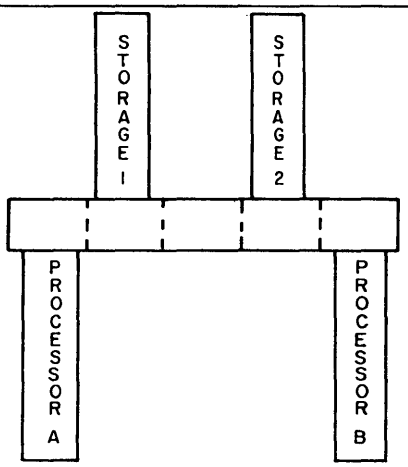
The resulting degradation under these circumstances should correspond to the real situation where no attempt has been made to reduce memory bus contention. It is obvious that the contention will be reduced with the addition of memory units, and some reduction should be realized by careful consideration of core mapping. Routines that reside permanently should be distributed in such a way as to evenly distribute the demand between physical storage units.

The following charts show the various storage-processor configurations as well as tables containing data related to those configurations, showing the increase in storage access time caused by priority determination and cable delay. Simulation results related to the configurations are presented, showing the relationship between the I/O data rate and the system degradation. This system degradation results from the priority determination and cable delay as well as from conflicting requests for memory cycles between CPU's and between I/O units and CPU's.



Simulation Results

I/O Rate in Megabytes	System Degradation
0	0
.8	1.1%
1.6	3.2%

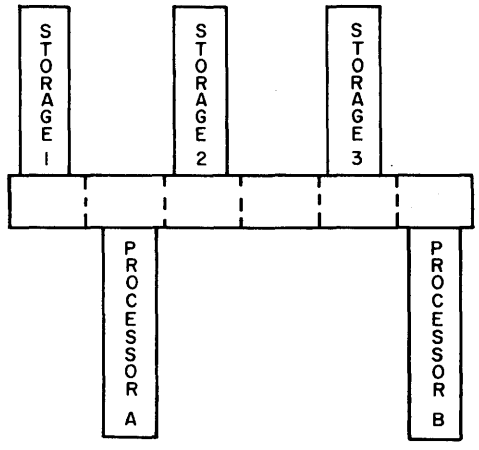


Storage access time increase in nanoseconds caused by priority determination and cable length:

	Storage Unit 1	Storage Unit 2
Processor A	150	200
Processor B	200	150

Simulation Results (including priority determination and cable length)

I/O Rate in Megabytes	System Degradation
0 0	8.8%
1.6	9.7%
3.2	10.8%

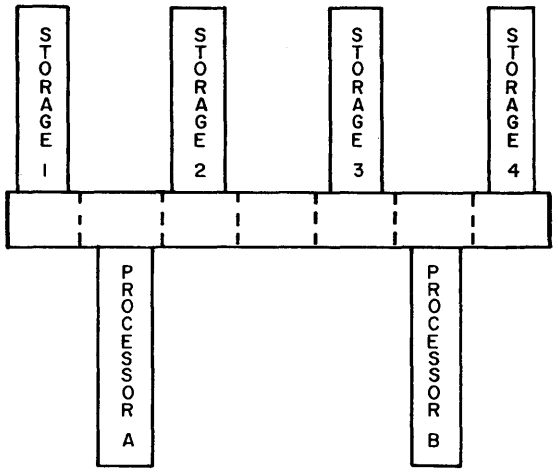


Storage access time increase in nanoseconds caused by priority determination and cable length

	Storage Unit 1	Storage Unit 2	Storage Unit 3
Processor A	150	150	200
Processor B	250	200	150

Simulation Results (including priority determination and cable length)

I/O Rate in Megabytes	System Degradation
0	9.1%
1.6	10.6%
3.2	11.9%



Storage access time increase in nanoseconds caused by priority determination and cable length

	Storage Unit 1	Storage Unit 2	Storage Unit 3	Storage Unit 4
Processor A	150	150	200	250
Processor B	250	200	150	150

Simulation Results (including priority determination and cable length)

I/O Rate in Megabytes	System Degradation
0	10.8%
1.6	12.7%
3.2	14.5%

RELIABILITY AND MAINTAINABILITY

INTRODUCTION

Two major factors considered in the design of System/360 were reliability and maintainability, which includes the ability to prevent, detect, and correct hardware malfunctions. The maintainability plan developed during the conceptual phase of System/360 and incorporated into the design provides operating and maintenance personnel with improved and expanded facilities. These facilities include extensive checking, automatic interrupts, log-out, and maintenance programs. Thus, isolation time (the time needed to identify a malfunctioning component assembly, such as a logic card) has been significantly reduced.

Throughout the engineering, manufacture, assembly, and test of the System/360, there is a continuing effort to achieve the highest standards of quality. The basic elements, Solid Logic Technology (SLT) modules, are designed and manufactured by IBM to exacting standards. The glass-encapsulated solid-state components are mounted on ceramic substrates and then sealed to form a circuit module that is impervious to contamination. This reliable module forms the base for System/360 and its reliability and maintainability.

A reliability and maintainability design review is incorporated in the design program to minimize reliability and maintainability problems caused by improper use of circuit and manufacturing ground rules. IBM provides this review function through "automated logic design" (ALD) techniques, which include:

- Verifying logic and circuit rules
- Generating etched-board layouts
- Generating logic diagrams directly from layout inputs
- Preparing maintenance diagrams that reflect actual wiring
- Providing engineering-level control

In addition, all IBM systems must be reviewed before and during development by a competent IBM specialist in reliability and maintainability, human factors engineering, industrial design, acoustics, etc., for agreement with IBM-recommended practice and policy.

IBM Product Test reviews the objectives and specifications of all IBM systems for measurability, and then actually evaluates initial systems relative to these specifications.

IBM Quality Control exercises in-process and final-review control of all manufactured systems, thus providing a continuing check that the system conforms to its specifications.

Furthermore, the outputs generated during the process of correlating information received from design, manufacturing, testing, and Field Engineering sources are utilized to improve reliability and maintainability. In this way a rapid and accurate exchange of information brings to each System/360 user the benefit of knowledge gained in all areas of the IBM Corporation.

An important function in any system capable of expansion is to maintain an effective maintenance concept throughout its growth. As it increases in size and complexity, the system must be capable of being maintained without increasing the operator's maintenance workload. For example, in a complex system, each piece of equipment should contain sufficient checking circuitry to enable a program to identify a malfunctioning data path, rather than depend totally upon the operator to identify such a data path. Also, the testing capabilities must be able to minimize the effect of maintenance upon the remainder of the system. Therefore, a decision must be made whether to provide initially for independent maintenance (computer use is excluded) or dependent maintenance (the computer is always required). A single approach is insufficient when minimum mean time to restore and impact upon the system must be considered. Therefore, IBM utilizes a combination of the two approaches: (1) fault isolation and unit checkout, utilizing the independent unit-testing capabilities, and (2) computer-controlled maintenance programs for isolating the more difficult malfunctions.

FACILITIES

Extensive Checking

Movement of data through the system is checked on a byte basis through parity checking. Checks for control failures are also made at strategic locations. Other modes of checking used in the system are comparisons and detection of invalid operations, addressing, data formats, specifications, and program rules. The details of an equipment malfunction are made available for analysis by a log-out process, which is a combination of hardware and a program. The log-out to core storage for the CPU is by means of circuitry, whereas the I/O units transfer the log-out information to storage upon demand by means of sense bytes. After the log-out is placed in core storage, the CPU can be programmed to relocate the log-out for eventual analysis and/or editing. Thus, as a by-product, the error environment and related statistics can be made available to maintenance personnel in printout or display form.

The purpose of this extensive checking is to provide an indication when data is inadvertently altered because of a machine malfunction or illegal operation, and to reduce downtime by helping to identify the malfunctioning data path. The machine-check (error) indication can be utilized as the basis for programmed-task retry or switchover in place of extensive redundant programming. Thus, the traditional problem with unchecked machines--that of not being able to recognize when a machine malfunction invalidated a task--is largely, if not completely, eliminated.

Automatic Interrupts

An automatic program-controlled interrupt system causes a hardware branch to a predetermined location for five types of interrupts: I/O, program check, machine check, monitor call, and external. Status indicators further define causes of interrupts within each type. Undesired interrupts can be masked out. The interruption system classifies errors as machine or program-check so that errors and causes of malfunction can be separated. The automatic machine-check interrupts and I/O malfunction checks also provide the initiating impetus for log-out and programmed-recovery procedures.

Log-Out

The log-out facility enables pertinent unit-environment conditions to be recorded for eventual output to a printer and/or a maintenance-history tape. The advantages of this concept allow a program to provide:

- A record of intermittent malfunctions that can lead to recognition of a degrading component (the analysis can be performed offline)
- A detailed record of malfunctions, thus virtually eliminating the need for maintenance personnel to depend on the operator for symptom information
- The details of an equipment malfunction, thus facilitating reconstruction of the error environment
- Data to support the process of analysis and localization of a malfunction

In addition, an increase in "uptime" (availability) will be realized by minimizing the necessity for maintenance personnel to gain direct access to the equipment for occurrences of an intermittent (transient) type malfunction.

MAINTENANCE PROGRAMMING

The primary method of localizing system malfunctions is through the use of diagnostic computer programs. The utilization of diagnostic programming ensures thorough and rapid fault localization with minimum reliance on the individual. Since these programs exercise the system units in nonfunctional ways and create abnormal conditions, they operate in a subsystem that has been partitioned using the configuration console.

Maintenance programs in this system fall into three categories:

- Online Test System — operating under TSS
- Multiprogrammed programs, which operate under control of a Diagnostic Monitor (DM)
- Hard-core programs, which diagnose failures in critical areas of the processor or system

The Online Test System operates as a series of problem programs under the Time Sharing System supervisor enabling maintenance of devices while continuing system service to users not directly affected by the device under test. The following devices are supported by the Online Test System:

2702 with 1052, 1051, and 2741 attached
2840 with 2250 Model 2 attached
2848 with 2260 attached
2701 with parallel data adapter, serial data adapter, and Type III adapter
2841 with 2311 attached
2314 disk system

Monitor-controlled maintenance programs usually test control units, I/O devices, and certain aspects of the terminals in the system. Other areas that they may test are storage if there are multiple storage units, and even channels and the processor if the program can conform to the restrictions imposed by the monitor. They perform checks of timing, error-checking circuitry, and functional operation. Running time of these programs is appropriate to the mode of testing and the unit involved.

Hard-core programs normally do not operate with a monitor, because they are not able to adhere to restrictions a monitor imposes. There will be cases where more than hard-core units are required to be tested in this mode, they are considered the exception rather than the rule. The types of programs operated or the units tested under these conditions are such that the processor cannot be reliably used to operate a monitor program and obtain necessary malfunction diagnosis.

The actual decisions as to the mode in which maintenance programs and maintenance actions will take place depend on the type of malfunction and the joint agreement of both the system operator and IBM Field Engineering.

Diagnostic Monitor (DM)

The diagnostic programs for a unit are run under control of a DM program. The DM provides a number of facilities that can be used by all of the device programs. Among these are program and data loading, interruption handling, code conversion, and communication to and from maintenance personnel or other operational monitor programs.

The diagnostic programs for a device communicate with the DM, using a well defined set of programming conventions called the "diagnostic interface". Thus, the diagnostic library can be augmented as needed. New programs can become a part of the standard library, provided that they conform to the specifications of the diagnostic interface. Another advantage here is that the DM exerts a normalizing influence that screens individual diagnostic program differences from the view of the user. Output messages are produced in standard formats.

Programmed Error Recovery

A set of program elements, which are an integral part of the TSM, is provided to utilize the failure-detection circuitry in the system. The purpose of these programs is to provide identification of the suspect unit and a complete, detailed, chronological system history of equipment malfunctions. This information would be made available to maintenance personnel to permit a detailed analysis of troubles before a subsystem is taken over for repair. It would also be used as the source of detailed and accurate symptom information on intermittent failures.

The programmed error recovery package consists of a set of modular elements, only one of which resides permanently in main storage. When a recordable incident occurs, the resident element causes the other necessary elements to be fetched into main storage. These other elements occupy part of the area on the peripheral storage device that is used to hold other operation monitor-program components. This area on the peripheral storage device is called "systems residence".

The programmed error recovery element that is permanently resident in main storage receives control of a CPU from two sources: the recovery nucleus and the I/O supervisor of the TSM.

Recovery Nucleus

In a multiple-processor system, a part of the resident supervisor is called the "recovery nucleus" (in fact, there is one copy in main storage for each active CPU in the system). Its primary function is to respond to machine checks and to locate a nonfailing CPU plus some nonfailing main storage, so that the TSM can continue operation. The processor that registered the machine check is then turned over to a program that records the failure information and retries the operation if feasible.

In the case of machine-check interruptions, the appropriate program element gets control after completion of the logging out of the processor status into main storage. The program adds other data to this logged-out information in order to complete the record of the error environment. The program then causes this record to be added to previous such records in an area reserved for this purpose on systems residence. The recovery nucleus then calls in the necessary service routines to restart the TSM supervisor. This may be an unsuccessful attempt to reread magnetic tape or an equipment-detected failure in the I/O channel device.

Utility programs are provided to process the environment records and to edit them into a form that is convenient for maintenance personnel.

BUILT-IN DIAGNOSTICS AND CHECKING FEATURES

In addition to the maintenance programs and the programmed error recovery package, other diagnostic provisions are made for most units. These provisions vary from manual controls and indicators for insertions, control, and observation of test data to built-in diagnostics initiated from the test panel. The remaining units utilize their control units or external testers for offline diagnostic purposes.

POWER AND THERMAL MALFUNCTIONS

Malfunctions in the power and thermal area are identified by indicators as to type and location. Manual intervention by maintenance personnel is required to correct such conditions.

PACKAGING

The components are packaged for ease of accessibility and replacement; in addition, functional packaging that simplifies troubleshooting is incorporated. Examples of functional packaging are:

- One complete register byte on a logic card
- One complete nine-bit parity-checking circuit on a logic card
- Two complete adder positions on a logic card

ADDRESS TRANSLATION (32-BIT VERSION)

The relocation tables used to translate a logical address into a physical address consist of "segment" tables and "page" tables. These tables are placed in main storage at the "segment table origin" and "page table origin" respectively. Each table occupies the number of storage locations specified by the respective "table length" amount.

Segment table origin and segment table length are specified by the contents of the "table register" (control register 0). The length is specified by bits 0-7 of this register, the origin by bits 8-31. The unit of length for the segment table is a group of entries (16 entries per group, four bytes per entry). Thus, the table length is variable in multiples of 64 bytes. Further, the address of the table origin must be a multiple of 64; hence, bits 26-31 of the table register must be zeros or a data exception is recognized.

In the 32-bit addressing mode, a segment table length of up to 256 groups of entries (pages) is possible. The actual table length is one more than the quantity specified by bits 0-7 of the table register. In the 24-bit addressing mode, the segment table length is always one group of entries, and bits 0-7 of the table register must be zero.

Each four-byte entry in the segment table defines a page table. The first byte (bits 0-7) defines the length of the page table; the remaining three bytes (bits 8-31) define the page table origin. The unit of length for a page table is a two-byte entry. Thus, the table is variable in multiples of two bytes. Each page table's origin is located at a byte address that is a multiple of two. Thus, bit 31 of each segment table entry that defines a page table is zero. If bit 31 is one, no translation takes place and a segment relocation exception is recognized (program interruption with interruption code 16).

Figure 8 illustrates the translation action when the 32-bit addressing mode is used. Bits 0-19 of the "logical" (or virtual) address are first compared with the corresponding bits of each associative register having bit 36 set to one. If a match is found, bits 20-31 of that register are used as bits 8-19 of the physical address, and bit 37 at the associative register is set to one. Bits 20-31 of the logical address are used directly as bits 20-31 of the physical address.

If no match is found, or if no register in the associative array has bit 36 set to one, the logical address must be translated by means of the segment and page tables. This translation proceeds as indicated by the dotted lines in Figure 8. The segment field of the logical address (bits 0-11) is first added to the origin address portion of the table register (bits 8-31). (For this addition the segment field of the logical address is aligned with bits 18-29 of the table register, since the entry to be fetched is four bytes long and has a byte address which is a multiple of four.) As part of the process of fetching the segment table entry, bits 0-7 of the logical address are compared with the segment table length (bits 0-7 of the table register). If the latter is less than the former, a segment relocation exception is recognized (program interruption with interruption code 16). The quantity obtained by this addition is the address of the segment table entry.

The segment table entry is used with the page field of the logical address in much the same manner as the table register contents were used with the segment field (see Figure 8). Bits 12-19 of the logical address are aligned with bits 23-30 of the origin portion of the segment table entry (bits 8-31) and the two quantities are added. The resultant 24-bit quantity is used as the address of a two-byte page table entry, which is subsequently fetched from storage. As described earlier, if bit 31 of the segment table entry is one, a segment relocation exception is recognized. In addition, bits 12-19 of the logical address are compared with the page table length (bits 0-7 of the segment table entry), and, if the former is greater than the latter, a page relocation exception is recognized (program interruption with interruption code 17).

The two-byte table entry consists of a physical page address portion (bits 0-11) and control bits (bits 12-15). Bits 13-15 must be zeros or a specification exception is recognized and the instruction is suppressed. Bit 12 defines the status of the page address portion of the entry. If bit 12 is zero, the page address is used as bits 8-19 of the physical address as shown in Figure 8. If bit 12 is one, a page relocation exception is recognized. Bit 12 thus serves to indicate whether the page referenced is actually available in core storage. The other three bits (bits 13-15) are reserved for future use.

The actual page address obtained by the translation method described above is not only used to address memory but is also loaded into an associative register along with the segment and page fields of the logical address. Thus, it is made available for future use without the need for repeating the translation process. When an associative register is so loaded, its bit positions 36 and 37 are set to one. (Selection of the registers to be loaded is under control of a usage algorithm that uses in sequence the registers with bit 37 set to zero.

When bit 37 is one in all registers, this bit is reset to zero in each register.) Bit 36 is used to indicate the presence of a valid entry in the associative register. It is reset to zero each time the contents of the table register are changed.

The translation process described above applies in all respects when 24-bit addressing is used, with the provision that bits 0-7 of the logical address are always zero.

RELOCATION MODE

Relocation of addresses provided by the processor is specified by bit 5 of the PSW (in extended control mode). When the bit is a one, relocation takes place; when the bit is a zero, the logical address is used as the actual address.

All main storage locations where information is stored in the course of an operation are subject to relocation.

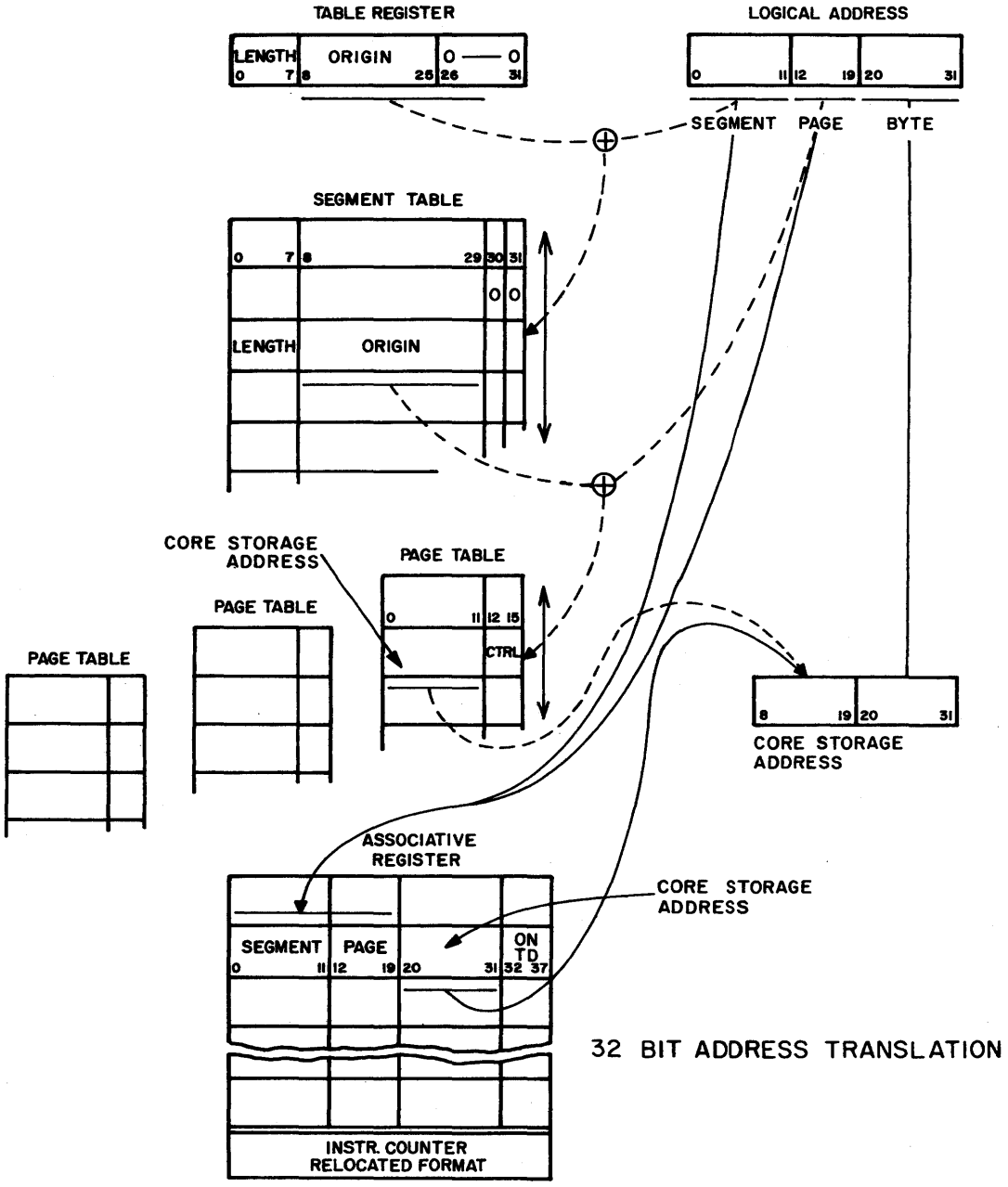


Figure 8. Simplified data flow for dynamic relocation

Addresses provided by the channels, either for fetching channel control words from main storage or for fetching data from (or storing data into) main storage, are never relocated regardless of the setting of bit 5 in the PSW.

Locations whose addresses are generated by the processor or channels for updating or interruption purposes (equipment-generated addresses), such as the timer, channel status words, or PSW addresses, are not relocated via the relocation tables. However, when the program specifies these locations, they are subject to relocation as defined above.

Core storage addresses in the range 0-4095 (including the above-mentioned equipment-generated addresses) are relocated by means of the primary or alternate prefix, as defined in System/360 Principles of Operation (A22-6821), unless the prefix is disabled by means of the prefix deactivation switch. Consequently, the prefix is applied when the address (either the logical address when no relocation takes place or the translated address obtained via the relocation tables) falls within the range 0-4095.

When relocation is specified, the storage protection, by means of the protection keys, is still active.

Whenever access to main storage is made by the equipment for the purpose of fetching an entry from a relocation table in the course of an address translation process, storage protection is ignored; that is, the equipment acts as if the block of storage containing the relocation tables was not fetch-protected during the memory cycle in which the relocation table entry is fetched. However, if the addresses at which the relocation tables are located are generated by the program, they are subject to storage and fetch protection in the normal manner.

If the storage address, generated in the address translation process for fetching a relocation table entry, exceeds the storage capacity of the installation, an addressing exception is recognized, resulting in a program interruption (interruption code 5).

EXTENDED CONTROL (32-BIT VERSION)

PSW Format

The system can operate under the control of a PSW in the following two modes:

- Standard PSW format as defined in System/360 Principles of Operation (A22-6821)
- Extended control PSW format, as defined in this section

Switching between these two modes is under control of bit 8 of control register 6. When this bit is one, extended control PSW format is used. Upon system reset resulting from a power-on sequence, manual system reset, manual IPL, or external start (electronic IPL), this bit is set to zero. If it is changed by a Load Multiple Control instruction, the resulting PSW mode change becomes effective upon completion of that instruction. (If other bits are changed by the Load Multiple Control instruction, the exact moment when the new settings take effect is unpredictable.)

The following are bit assignments of the PSW format for extended control:

Bits	
0-3	Must be zeros. Otherwise, a specification exception is recognized whenever the PSW containing a nonzero bit in these positions is used (that is, during the fetching of the first instruction under the control of this PSW, similar to the specification exception caused by a nonzero bit in position 63 of the PSW).
4	24/32-bit virtual addressing mode bits. When this bit is a zero, 24-bit address arithmetic applies. When this bit is a one, 32-bit address arithmetic applies.
5	Relocation mode bit. When this bit is a zero, no relocation takes place. When this bit is a one, relocation takes place.
6	I/O mask bit, controls the masking of all I/O channels. When this bit is a zero, all I/O channels are masked off. When this bit is a one, the masking of the individual channels is contained in control register 4-5.
7	External mask bit. When this bit is zero, all extended interruptions are masked off. When this bit is one, masking of external interruptions is controlled by control register 6.
8-11	Protection key (same as standard PSW format)
12-15	AMWP (same as standard PSW format)
16-17	Instruction length code
18-19	Condition code
20-23	Program mask
24-31	Must be zeros. Otherwise, a specification exception is recognized.
32-63	Logical instruction address (Note: When bit 4 is zero, bits 32-39 must be zero or a specification exception is recognized.)

In addition to storing the above fields of the PSW as an "old PSW", the 16-bit interruption code field generated as a result of an interruption is stored as a halfword in storage in the following byte location.

Interruption Type	Storage byte location
External	14-15
SVC	16-17
Program	18-19
Machine check	20-21
I/O	22-23

Control Registers

A set of control register positions is provided as part of various features. Up to 16 registers of 32-bit positions each may be provided. The bit position assignments for the control registers are shown below. Some of the bit positions are assigned for the purpose of sensing the settings of manual switches, and are therefore not implemented as registers. These bit positions can only be stored into main storage, but cannot be loaded from main storage. The control registers are not part of addressable storage. They are changed by Load Multiple Control and inspected by Store Multiple Control instructions.

Bit positions of the control registers are assigned as follows.

Control Register	
0	Table register (for dynamic relocation)
1	Unassigned
2	Relocation exception address register
3	Unassigned
4-5	Extended mask registers. The bit assignment is as follows: Bit 0-63: I/O channel mask for channels 0-63
6	Bits 0-3: machine check mask extensions for channel controllers. Bit 8: extended control mode. Bit 9: configuration control. Bits 24-31: external interruption masking as defined in the following table.

Interruption Source	Bit Position*
Timer	24
Interrupt key	25
External signal 2	26
External signal 3	27
External signal 4	28
External signal 5	29
External signal 6	30
External signal 7	31

*Bit position applies to:
(1) PSW for interruption code
(2) Control register 6 for masking

7	Unassigned
8-9	States of core storage partitioning switches, one eight-bit byte for each logical processor storage unit. The bits in the byte correspond to the eight tails of the logical processor storage units, with "one" indicating that connection is established over the tail.
10	Bits 0-31: Core storage address assignment, one four-bit field for each of the maximum of eight logical processor storage units. The four-bit field contains bits 11-14 of the assigned core storage address.
11	Bits 0-15: States of channel controller partitioning switches with one four-bit field for each channel controller. The bits in the field correspond to the four tails of each channel controller, with "one" indicating that connection is established. Bits 16-31: Channel address assignment (as viewed from the processor executing the STMC instruction), one four-bit field for each of the maximum of four processors. A field containing three zeros and a one indicates that for the particular processor, only the channel controller corresponding to the bit position that is "1" is addressable and its channels are 0-6. No other bit combinations are possible in these four-bit fields.
12-13	States of control-unit partitioning switches, with at least two bit positions assigned to each control unit. "One" indicates that connection is established. The particular assignment of bit positions is presently left open.

14 Bits 0-23: Unassigned

Bits 24-27: States of direct-control partitioning switches, one bit for each processor. "One" indicates that the direct control interface of the corresponding processor is connected to the other CPU's; zero indicates that the direct-control interface is disconnected from the other processors.

Bits 28-31: States of prefix deactivation switches, one bit for each processor.

15 Unassigned

NEW INSTRUCTIONS (32-BIT VERSION)

The following new instructions are included in the instruction set of the Model 67: Branch and Store, Load Real Address, Load Multiple Control, and Store Multiple Control; discussion of these new instructions follows.

	Mnenomic	Type	Code
BRANCH AND STORE	BASR	RR	0D
BRANCH AND STORE	BAS	RX	4D

In the 32-bit addressing mode, the updated logical instruction address is stored as link information in the general register specified by R_1 (in bit positions 0 to 7, and 8 to 31 respectively). Subsequently, the logical instruction address is replaced by the logical branch address.

The branch address is determined before the link information is stored. The instruction length code is 1 or 2, depending on the format of the Branch and Store.

In the 24-bit addressing mode, zeros are stored in bit positions 0-7 of the general register specified by R_1.

Condition code: The code remains unchanged.

Program interruptions: None.

Programming note: The link information is stored without branching when in the RR format and the R_2 field contains zeros.

When Branch and Store is the subject instruction of Execute, the instruction-length code is 2.

	Mnemonic	Type	Exceptions	Code
LOAD REAL ADDRESS	LRA	RX	M,S	B1

The translated address of the second operand is inserted in the low-order 24 bits of the general register specified by the R_1 field. The remaining bits of the general register are made zero.

The address specified by the X_2, B_2, and D_2 fields is translated through the dynamic relocation features (regardless of whether the relocation mode bit is a zero or a one), and the translated address is inserted in bits 8-31 of the general register specified by R_1. Bits 0-7 are set to zero. The translated address is not inspected for protection or resolution.

During the address translation process, no relocation exceptions are recognized. Instead, the condition code is used to indicate successful translation or the reason for its failure. If the translation was unsuccessful, the contents of R_1 are replaced by the table entry, leaving its availability bit set to one; and the logical address that was to be translated is stored in control register 2.

	Mnemonic	Type	Exceptions	Code
LOAD MULTIPLE CONTROL	LMC	RS	M,A, S, D	B8

The set of control registers starting with the control register specified by R_3 is loaded from the locations designated by the second operand address.

The storage area from which the contents of the control registers are obtained starts at the location designated by the second operand address and continues through as many storage words as needed. The control registers are loaded in the ascending order of their addresses, starting with the control registers specified by R_1 and continuing up to and including the control register specified by R_3, with control register 0 following control register 15. The second operand remains unchanged.

If any of the bits loaded into positions 26-31 of control register 0 are ones, a data exception is recognized.

Condition code: The code remains unchanged.

Program interruptions: Privileged operation, addressing, specification, data

	Mnemonic	Type	Exceptions	Code
STORE MULTIPLE CONTROL	STMC	RS	M,P, A, S	B0

The set of control registers starting with the control register specified by R_3 stored at the locations designated by the second operand address.

The storage area where the contents of the control registers are placed starts at the location designated by the second operand address and continues through as many storage words as needed. The control words are stored in the ascending order of their addresses, starting with the control register specified by R_1 and continuing up to and including the control register specified by R_3, with control register 0 following control register 15. The control register remains unchanged.

Condition code:	The code remains unchanged.
Program interruptions:	Privileged operation, protection, addressing, specification

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York 10601