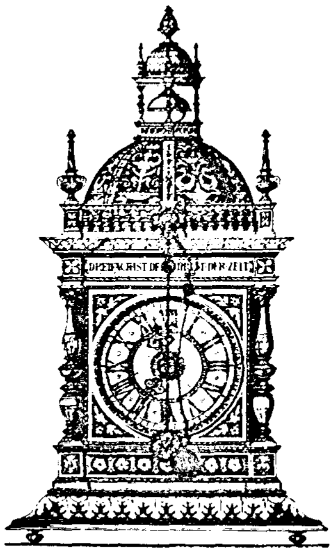
5

1-68
Part 1

Fundamentals of Time Shared Computers

by C. Gordon Bell

Associate Professor of Computer Science and Electrical Engineering, Carnegie Institute of Technology. Formerly manager of Computer Design, Digital Equipment Corporation, Maynard, Massachusetts.



"Time-sharing" is discussed generally in this article to cover any application of a computer system that has simultaneous users. The discussion defines general purpose time-sharing so as to include special purpose time-sharing, "real time", and "on line" systems as a subset. "Graceful Creation", or the "boot strapping" of a system, is described in which newly created individual user procedures are immediately available to the whole community of users, and the system expands in an open-ended fashion because many users contribute to the formation.

Although the discussion is separated into hardware, operating system software, and user components, a sharp delineation does not exist in reality. After the basic system is specified, it is the philosophy of the author that the system should be formed in a time-shared environment (including the construction of the operating system software). Few resrictive features or functions should be "built-in", but instead, be optionally available through the library or common files.

The underlying design criteria should be: flexibility, modularity, simplicity of module intercommunication, and open endedness.

The basic objectives of time-sharing are to increase user and/or overall computer system productivity. Present general computational systems are an extension of special, shared, multiprogrammed systems centered around special applications (e.g., process control, command and control, information inquiry, etc.). As such, time sharing is another technique that makes the computer a more general tool.

All future computers will have at least some basic hardware for a form of time-shared usage. These systems forms will run the gamut from dedicated systems with a permanent user, through general systems with varying number of users, to a network of shared computers.

The article discusses only the basic structure of the system, with emphasis on the hardware, because of space limitations. For example, the issue of scheduling jobs is discussed only superficially by listing the system variables on which scheduling depends, together with a common scheduling algorithm.

INTRODUCTION

Time-Sharing is the simultaneous shared use of a computer system by independent users expecting short or appropriate (or apparently instantaneous) responses, within the limits of the request and system, to computational demand stimuli.

Time sharing provides a level of service to a user who could only previously have had the service by owning his own computer. The sharing is based on the principle that there is enough capacity in a computer for multiple users, assuming: the proper ordering of requests; the user consoles are active only a small fraction of the time; and a console is being used for input or output, in which case, another user can be processed on an overlapping basis during the input or output.

TIME-SHARING SYSTEM COMPONENTS

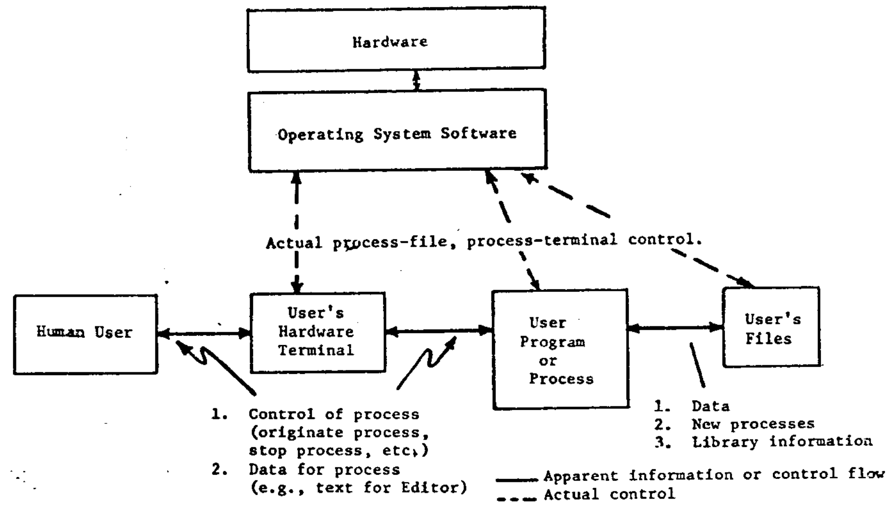
The system components (see Figure 1) include the *operating system software,* the *hardware,* and the *user.*

The Operating System Software

The Operating System Software is responsible for the allocation of resources among users and the efficient management of the resources. In addition, it manages all common software procedures (or program library), such as translators, management of files or data bases, editing programs, etc. The system provides logical abilities, such as message switching among user terminals.

The Hardware

The hardware enacts the procedures required by either the user or the operating system, and provides the physical components which make a logical and physical implementation possible. The hardware components are: processors, primary memories, peripherals (terminals and file memories), control and switches.



Hardware
Operating System Software
Actual process-file, process-terminal control.
Human User
User's Hardware Terminal
User Program or Process
User's Files
1. Control of process (originate process, stop process, etc.)
2. Data for process (e.g., text for Editor)
1. Data
2. New processes
3. Library information
Apparent information or control flow
Actual control

Fig. 2. User's apparent system.

The User's Apparent System

The User's Apparent System includes the terminals, files, and a process as shown in Figure 2.

The *terminals* provide a node for a communication link between the system and user for the control of the user process and transmission of data. Terminals are at the computer's periphery and include devices like typewriters, printers, cathode ray tube displays, audio output response units, etc.

The *files* or *data base* retain the user's information while in the system. This information includes both his dormant processes or programs, or, in general, all the data he wishes the system to retain.

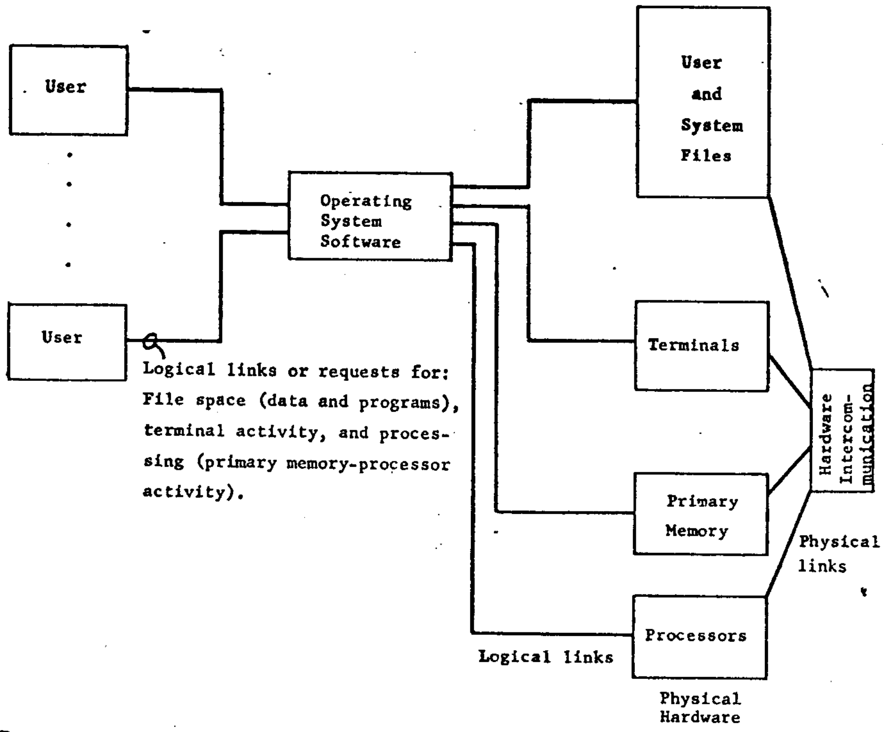
The *user process* or *user procedure* or program directs the system for his file, terminal, and processing activity.

TIME-SHARING CRITERIA

Time-shared computers' basic criteria are: being shared among multiple users; providing independence among the users; and providing nearly "instantaneous" service to its simultaneous users (within the limits of their requests).

Independence Criteria

For each system component the relationship among users may vary over a range from dependence (the simultaneous attempt of a group to solve a single problem) to independence (no user affects another user). A completely independent system would require the system to perform as though each user were the sole user.



User
User
Operating System Software
User and System Files
Terminals
Primary Memory
Processors
Hardware Intercommunication
Logical links or requests for: File space (data and programs), terminal activity, and processing (primary memory-processor activity).
Physical links
Logical links
Physical Hardware

Fig. 1 Logical organization of time-shared computer components.

TABLE 1. CAPACITY REQUIREMENTS FOR TIME-SHARING SYSTEM APPLICATIONS

Specialized System Service, or Application	Primary Memory for Process (in bits)	Primary Memory for User Data (in bits)	Processing Capacity/ User (in operations*/ interaction†)	File Organization and Size (10^6-10^9 bits)	Direct Terminals
Desk calculator	very small	very small ($<10^3$)	very small ($>10^4$)	none	typewriter, input keyboard, strip printer, scopes, audio output, or special console.
Stock quotation	small	small ($<10^4$)	very small ($>10^4$)	one (small-medium)	see above, stock ticker tape or transactions input, telephone.
Airline reservations	medium	small ($>10^4$)	small ($>10^5$)	approx. 6 (medium-large)	special consoles, typewriters, scopes.
On line banking	medium	small ($>10^4$)	small ($>10^5$)	approx. 10 (medium-large)	see above, special bank teller consoles.
General conversational computational languages (JOSS, CULLER-FRIED System)	medium	small-very large (10^3-10^5)	small-large unbounded (10^4-$>10^8$)	multiple files per user, with few file types (medium-large)	typewriter, printer, scope, plotter. (Culler-Fried consists of scope, keyboard, and tablet.)
Specialized computer aided design, engineering, problem solving languages (COGO, etc.)	medium-large	small-very large (10^3-10^5)	small-very large (10^4-$>10^8$)	see above	see above
Process control	medium-large	medium ($>10^5$)	small-very large (10^4-$>10^8$)	few (small)	physical quantity transducers, general user terminals.
Text editing (Administrative Terminal Service)	medium	small ($>10^4$)	small (10^4-10^5)	multiple single purpose files/user. (medium)	typewriter, printer, scope.
On line information retrieval of periodical headings, bibliographies, keywords, abstracts	medium-large	medium ($>10^5$)	medium (10^5-10^7)	one (very large)	see above. telephone (dial in, audio out)

*assumes a fairly sophisticated processor and instruction set
†maximum interaction intervals for user requests are $\simeq$ 10 sec.

File independence, for example, is controlled by associating information with the file concerning the file's users, and uses to which the file may be put. Such file directory data provides system capability to cover a wide range of applications concerning private and public data bases. In fact, systems could be categorized by the organization of their data bases. Table 1 presents some special purpose systems which are ordered approximately in terms of the filing demands. For example, a file containing a teaching program may be universally available, while a program for monitoring the teaching program or for grading the users may not.

Process or program independence (and dependence) is the most expensive hardware aspect of user independence. One program cannot affect nor destroy another; on the other hand, a mechanism for making procedures available to the community's members is necessary.

Instantaneous Criteria

The instantaneous nature of a time-sharing system includes both direct terminals for the users and rapid response to user demands. That is, users are "on line" and served in "real time". An *on line* computer is one which provides terminals which allow users to directly communicate with it by a single, simple action, (e.g., like pressing a typewriter key or looking at a display). The system is never farther away than the nearest terminal. A *conversational program* is an on line program which allows a user to directly communicate or "converse" with it in terms of requests and acknowledgement dialogues at an appropriately rapid rate.

A *real time* system is one which has the ability to execute a required process or program in an "acceptable" period of time as governed by the extra computer process requesting computation power. All systems are real time if they are acceptably fast: e.g., overnight for payroll calculation might be acceptable!

Normally, we associate "real time" with a mechanical process in which a computer is constrained by a mechanism, e.g., a "real time" computer for air traffic control must be able to process all the inputs from the radar system such that aircraft positional information is not lost.

The *response time* or total time for the system to respond to a demand stimulus is the sum of the *reaction time* (the time until a program is activated from the request time) plus the *processing time* (the time to process the request).

Response times for human users should vary in accordance with their requested demands. The response time for a computational demand, although known and determined by the system, can only be judged for acceptability by its users. In summary, "real time" for a mechanical process means keeping up with the process (not losing information, etc.). "Real time" for a human process is giving an appropriate response in accordance with requests.

Shared Criteria

The sharing of a system by multiple users represents an economic justification by ordering or optimizing random resource requests. The *allocation of resources* is a major system function and includes: processor *scheduling,* or the allocation of processing capacity for process or program execution; *file allocation* provides for the user assigned space from the available file space; *primary* or *memory allocation* is the allotment of memory space for the execution of processes; and *terminal allocation* or the assignment of terminals to users.

General Purpose Time-Sharing Criteria

All of the above criteria must be met for a time-sharing system. In addition, one other criteria, generality, or open endedness, separates special purpose and general purpose systems. A general purpose time-sharing system must provide for the open-ended creation of new processes or procedures during system operation time, which in themselves may be considered part of the "system." This ability, or *graceful creation* of an improved or ever-expanding system with increasing abilities defines an open-ended general system. In the limit, users concerned with the development of the operating system software may, for example, operate and test a complete, new time-sharing system program to replace the existing system within the framework of the old system. As new processes, languages, procedures, etc., are added to the operating system software or placed in the general user's public domain, the line delineating the operating system process and the user process becomes less sharp.

The method (or language) of procedure creation, testing, and execution is the measure of generality. In summary, a simple test for generality can be made by determining whether a new language can be added to the system from a normal terminal or console. The user should have freedom inherent in the hardware (or at least in the processor), including the ability to write programs in machine language.

SPECIAL PURPOSE AND GENERAL PURPOSE TIME-SHARING

In most new systems, basic time sharing hardware can be easily provided in the design at low cost. The general organization of all computers provides the inherent ability to form a time-sharing system. Indeed, time-sharing systems have been implemented on machines covering a wide range of problem applications. In general, the systems formed, using computers which have little or no supplementary hardware, are restricted to a single application. The ease with which a total system may be implemented on a configuration is determined for the most part by the configuration and the inherent hardware facilities that aid the configuration sharing. The features which assist resource allocation must be included for implementing general purpose systems. The hardware can limit the general purposeness in a fashion similar to the operating system software. The additional hardware to provide some form of resource sharing can be quite small.

Although the ability to implement a general purpose system on a specific hardware configuration may be a desirable design criteria for the hardware, a special purpose or dedicated system may be more desirable. A configuration dedicated to a particular use may be designed to provide a much more efficient utilization of the resources than one which attempts to serve all users solving all problems.

It may be more advantageous to form communities of users who share the same system and are only interested in solving specific classes of problems on single systems. Systems which already are limited by a single resource might stand alone. For example, present hardware file capacity and file access capabilities appear to limit desired library systems. (Thus, a general system cannot supply the necessary resources, nor can the resources be supplied even if a dedicated system were built.) Table 1 gives a list of dedicated computer applications.

A network of dedicated computers

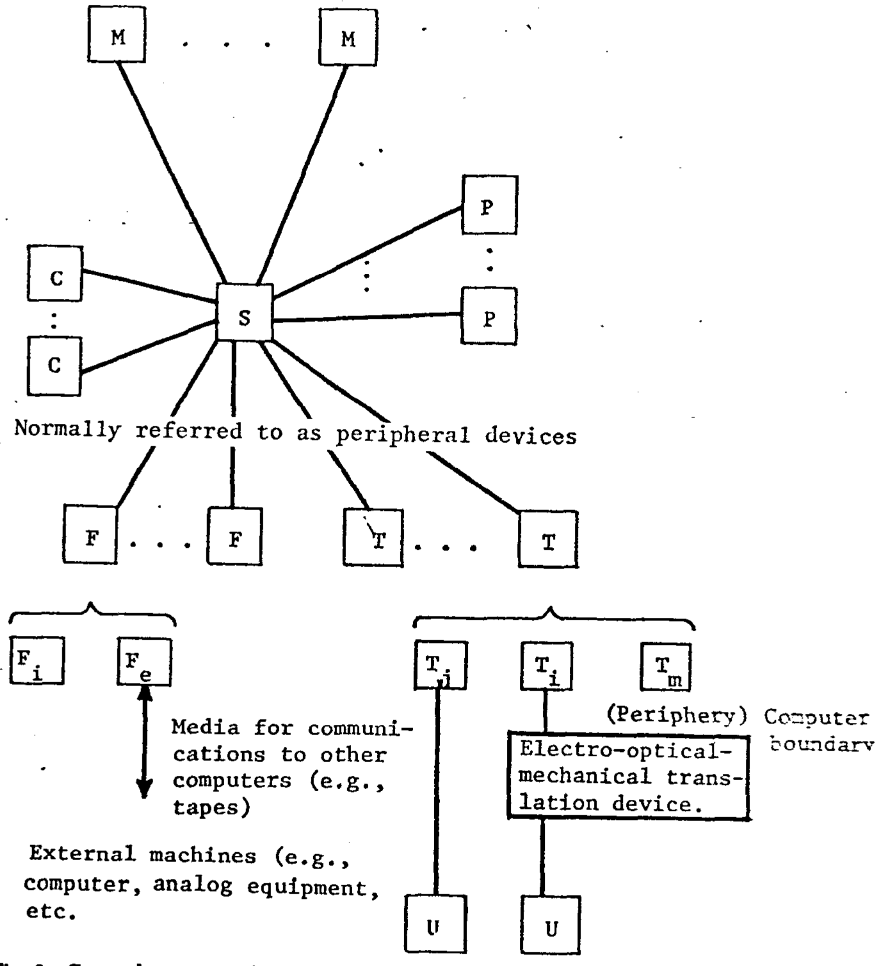

M
M
P
P
C
C
S
Normally referred to as peripheral devices
F
F
T
T
F_i
F_e
T_j
T_i
T_m
Media for communications to other computers (e.g., tapes)
(Periphery)
Computer boundary
Electro-optical-mechanical translation device.
External machines (e.g., computer, analog equipment, etc.
U
U

Fig. 3. General structure of present computers in terms of computer components.

which only solve specific problems, supply special resources, or "understand" specific languages may be a better solution to efficient usage of our machines than the large, general purpose systems which try to provide any or all services.

HARDWARE

COMPUTER STRUCTURE

Although hardware can be considered at various description levels from memories or processors down through "AND" gates, on to circuits, the level of interest for this discussion is the computer and its components. The general structure of the computer is shown in Figure 3. The computer's components are: primary memories, processors, controls, peripherals (terminals and memory files), and switches. The communication between any pair of components is via switches which provide both "data and control" information paths.

A single *computer* has any number of components (memories, processors, controls, peripherals) but every processor in the computer must access some of the common primary memory of the system.

A *multi-processor* computer has more than one processor. Multi-processing is the simultaneous processing of one or more computational programs or processes by multiple processors. Multi-processing methods can vary from non-anonymous job assignment, in which particular processors or types of processors are assigned to specific roles, to anonymous processors being assigned to any job in the system.

It is difficult to have complete anonymity because particular processors in the system can only handle a limited class of jobs (especially Input/Output Processors).

A *parallel processor* computer has multiple, anonymous processors, each of which can be assigned to different, independent, parallel (processed simultaneously) parts of a single task.

All computer structures are special cases of that shown in Figure 3. Most systems have hierarchial or tree-like structures like that of Figure 4. Each switch is, in fact, more closely associated with a particular component, and takes on the special properties necessary for switching or selection among particular components. Thus, a particular tape control unit may communicate with up to eight tape units and the particular kind of information exchanged between the two units is a function of the kind of units. The tree-like structure exists not only because of the number and type of units and the way they inter-communicate, but also because the computer is a simplex structure. That is, assuming that it is necessary for communication to be carried out from bottom to top (a terminal or file to primary memory), there is only one path for the communication flow. Figure 5 presents the structural forms the switches take.

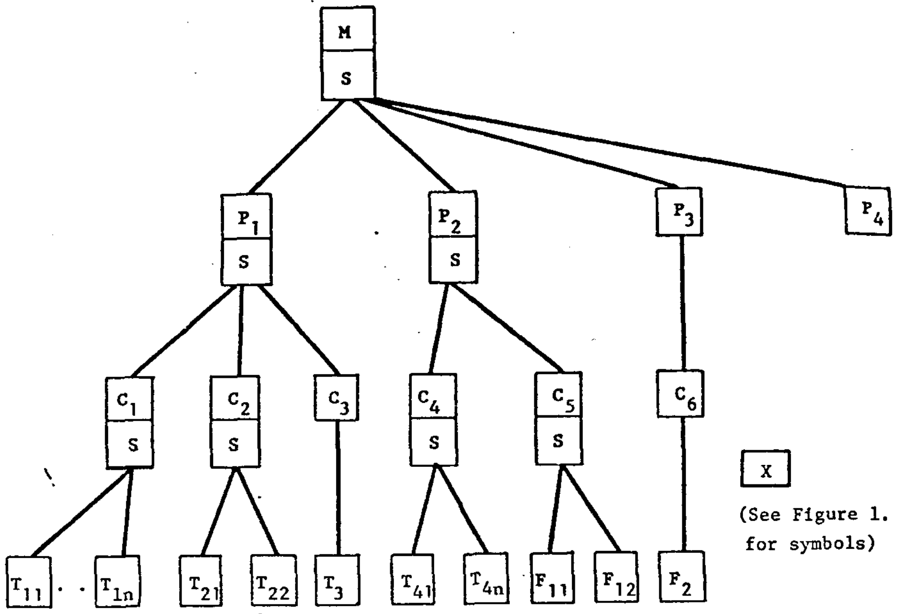


Fig. 4. Structure of a simplex computer system.

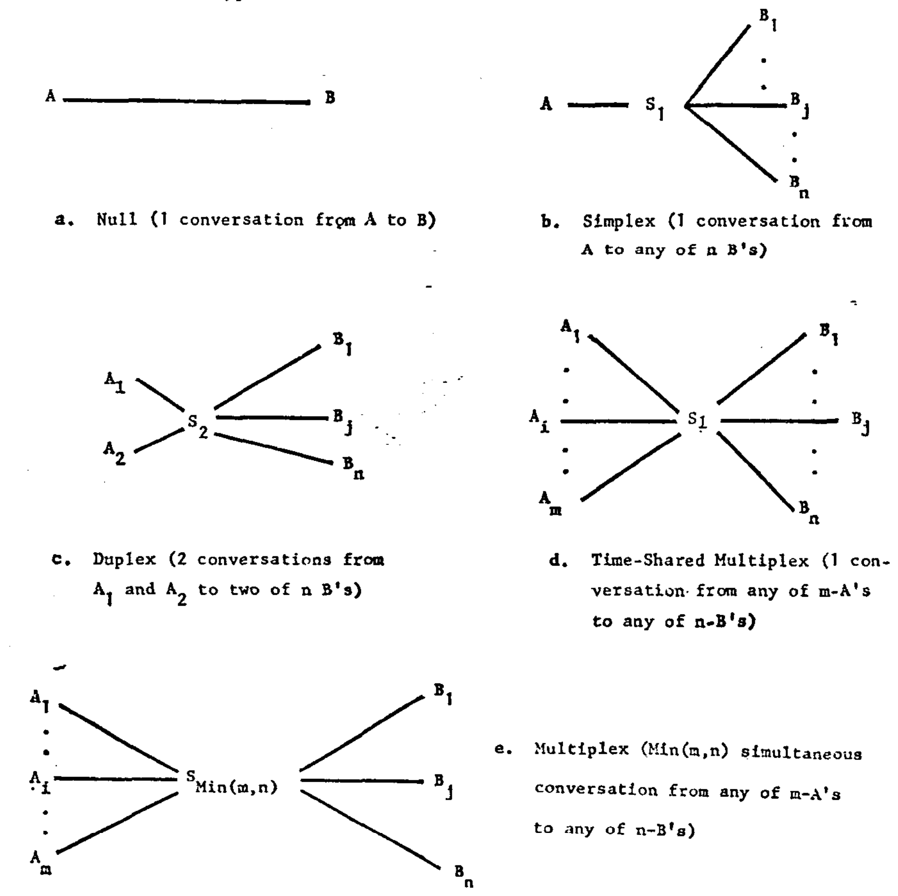

A
B
a. Null (1 conversation from A to B)
A
S1
B1
Bj
Bn
b. Simplex (1 conversation from A to any of n B's)
A1
A2
S2
B1
Bj
Bn
c. Duplex (2 conversations from A1 and A2 to two of n B's)
A1
Ai
Am
S1
B1
Bj
Bn
d. Time-Shared Multiplex (1 conversation from any of m-A's to any of n-B's)
A1
Ai
Am
SMin(m,n)
B1
Bj
Bn
e. Multiplex (Min(m,n) simultaneous conversation from any of m-A's to any of n-B's)

Fig. 5. Computer component switch or selection configurations.

Figure 6 gives a computer with multiple paths between a primary memory module and a given peripheral element. Since there is some redundancy among components, it can be shown that there is a higher probability that the computer will be in an operational state, as measured by some large fraction of memories, processors, terminals, and files being operational. Such an operational state would undoubtedly be at reduced performance. The probability of a system being operational is a function of the computer structure (the number of components and their interconnection) and each component's probability of failure.

For systems requiring a large fraction of availability or a high uptime, it is necessary to at least duplicate each component of the system. Such systems can be designed so that all units are constantly in service (including the duplicates), and when a system failure occurs, the faulty unit is removed or the system re-partitioned for maintenance. Such a design philosophy, called *graceful degradation* or *fail soft,* provides continuous usage even though the capacity may be degraded. Fail soft design imposes the constraint on the hardware that there be a duplicate of each unit and communication path in the system. It is possible to have similar functional duplicates to avoid complete duplication, i.e., a drum can be replaced by a disk. In such cases, the system will continue to function, but at very much reduced capacity. These computers also must have ability to detect first fault occurrence at a computer component so that errors will not propagate through the entire system, making fault location difficult. Once a faulty unit is detected, the system must be able to be dynamically reconfigured.

Multiple units can also provide a means of achieving better overall system performance since the units can be used for operation while they are standing-by.

PRIMARY MEMORY COMPONENT

The *primary memories* (usually core or thin films) retain the active portions of both user and operating system processes. These processes are either being enacted by a processor or are waiting for a processor. The primary memory may also contain memory maps and status information regarding the system's users.

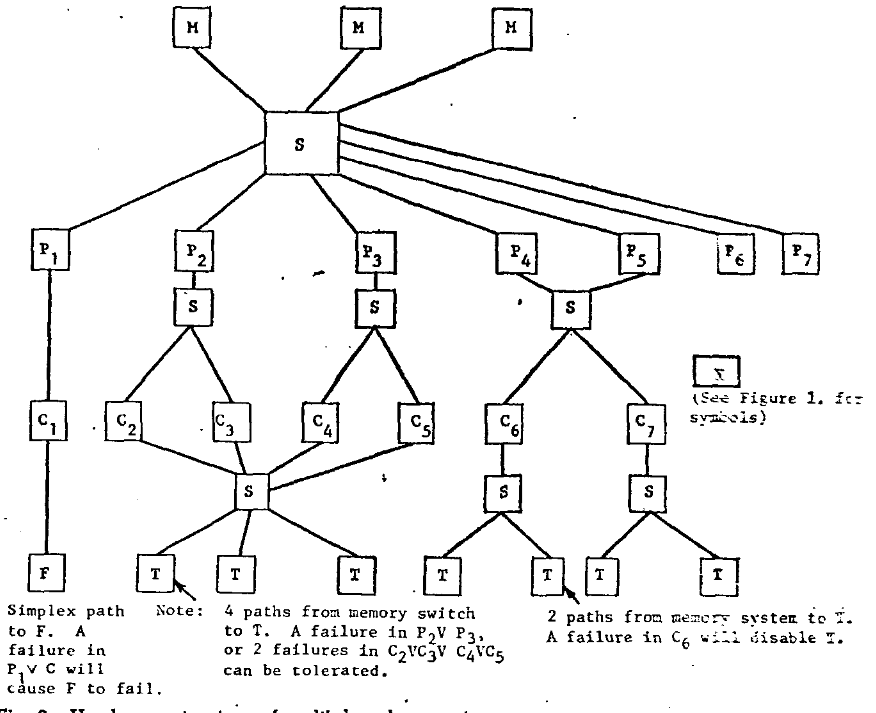

M
M
M
S
P_1
P_2
P_3
P_4
P_5
P_6
P_7
S
S
S
C_1
C_2
C_3
C_4
C_5
C_6
C_7
S
S
S
F
T
T
T
T
T
T
T
(See Figure 1. for symbols)
Simplex path to F. A failure in $P_1 \vee C$ will cause F to fail.
Note:
4 paths from memory switch to T. A failure in $P_2 V P_3$, or 2 failures in $C_2 V C_3 V C_4 V C_5$ can be tolerated.
2 paths from memory system to T. A failure in C_6 will disable T.

Fig. 6. Hardware structure of multiplexed computer.

The primary memory is the medium of logical intercommunication between the hardware and software components.

The arrangement of the memory subsystem, as shown in Figures 4 and 6, is such that from the processor's viewpoint, a number of *access points,* or *ports,* are provided with which the processors connect. The physical form that a memory subsystem (the memories and the switch to which the processors connect) takes is described by:

1. The number of independent memory modules.
2. The properties of each memory module.
- The data width (in bits) of information accessed at one time.
- The quantity of information stored (in bits).
- The *access time* — the time the module requires to obtain data, given that the module is free, from the time an access request has been made.
- The *cycle time* — the time the module requires to completely acknowledge a request, and become free for the next request.
- Memory failure probability (detected failures and undetected failures).
3. The method used to assign physical addresses (which the processor uses) to physical memory modules and memory words.
4. The switching network which connects with the processors. See Figure 5 for possible switches. These range from 1,2,P (where P is the number of processors), to M (or the number of memory modules) as possible simultaneous conversation among processors and memories.

All primary memories are functionally similar because they store programs while they are being interpreted by a processor; data for programs; and other state information required by the processors. The memories can be separated according to their specific functions on the basis of their cost, size, and speed.

Principal Primary Memory (Core or Thin Film Technology)

This memory is the principal storage for programs while they are run. In most computers, the assumption is made to provide a certain match between processor capacity (in bits/sec.) and the available primary memory cycles (in bits/sec.). In small computers this is the only Primary Memory in the computer.

Bulk Memory or Large Capacity Storage

These memories have the following characteristics relative to primary memory: — cheaper ($.02-.04/bit versus $.10-.20/bit); larger

(0.5-1) million words versus 32,000-256,000 words; and slower (8 μsec/word versus .8 μsec/word).

The assumptions about use are:

1. Problems involving large data structures in which data is randomly accessed.
2. As program base for seldom executed user and system programs.
3. As data base for seldom accessed data. (Whether a program is moved from bulk memory to principal memory is a function of movement overhead, and the expected activity.)

A bulk memory is also often used as a secondary storage device to hold programs and data (types 2 and 3 above) which are transferred to primary memory (higher speed) for execution. Thus, it is treated essentially as a fast, zero access, drum-like device for program swapping.

Scratch-Pad Memories

These memories have the following characteristics relative to primary memory — faster (by a factor of 5); more expensive (by a factor of 10-100); and smaller (20-1000 words). Such memories contain:

1. Short loops for high-speed program execution
2. Control information which may be referenced by I/O processors
3. Either the processor state or copies of the processor state (arithmetic, index registers, status information, etc.).

PROCESSORS

Processors connect with primary memory and enact user computational (arithmetic, symbolic, logical, etc.) processes. Large systems require several types of processors to efficiently handle the different tasks, to provide redundancy, and to match the capacity of the memory system.

Processors can be specified at the computer system level by the following parameters:

1. Instruction set ability
- Distribution of processing time required for the given algorithm being ocessed.
- Distribution of memory space for the algorithm.

2. The number of programs which are recognized as independent processes. (This number is roughly equivalent to the number of interruptor trap channels.)
3. *Program switching time* or the time to save a process state, and to reset a processor to a new process state.
4. The number of bits (or words) associated with a process which resides in the processor and must be swapped when a new process is selected.

Computation Processors, Central Processing Units, Arithmetic Processors, or General Purpose Processors

These interpret memory-provided processes, and most generally perform arithmetic, symbolic, and logical functions. This conventional processor handles user and operating system processes. In small systems, it is the only processor, and as such interprets input-output commands for peripheral devices.

Special Purpose Processors or Algorithm Processors

These (arithmetic/logical) processors interpret a limited command set for special languages or algorithms and augment a general purpose processor. This type of processor has so far only been used experimentally (e.g., to process IPL V statements or evaluate polynomials). Future possibilities include the use of special processors for cross/auto correlation, fast Fourier series transformation, Matrix Multiplic. ion, etc., algorithms (e.g., IBM 360/2938 Array Processor).

Peripheral Processors, Input-Output Processor, Input-Output Control Units, or Data Channels or Channels

These interpret a limited set of commands or instructions which handle controlling the transmission of data between peripheral control unit peripherals and primary memory.

Peripheral processor programs exist in primary memory, and are usually created by arithmetic processors. Though they do not usually have the arithmetic, logical or symbolic, capability, they do possess enough logic to do algorithm decoding. When necessary, arithmetic processors augment the peripheral processors.

The instructions interpreted by peripheral processors include:

1. Terminal initialization commands.
- Selection of data transmission path by selecting both the control unit and peripheral device.
- Device function specification commands. These include commands for — reading, writing, unit speed, and directions selection, data transmission formats, etc.
- Location of information within the peripheral. If the device is organized in such a fashion to regard its data as being addressable or accessible by a number, the location must be specified.

2. Peripheral status query commands. At various times, the processor queries the state of the control unit-peripheral device and places the status in primary memory.
3. Peripheral program execution (in addition to initialization and status query commands). These instructions include:
- Branching.
- Setting up of commands for block data transmission.
- Intercommunication with other processors, by issuing commands to the processors. Also, a peripheral processor trapping may transfer job completion information into a queue.

4. Supervision of actual data transmitted between peripheral-control and primary memory.

Block Data Transfer Processors

These processors are a special case of the peripheral processors, and are used to execute the special instruction to transfer an array or block of data in primary memory to another location in primary memory.

Display Processors

These processors are specialized peripheral processors which interpret display procedures. That is, a display processor program in memory, when interpreted by a display processor, yields a picture.

PERIPHERALS

The peripheral devices are at the physical and logical periphery of the computer as can be seen by the tree-like structure of Figure 4. The communication to peripherals is controlled from programs in primary memory which transfer information with the periphery from memory to processor to control unit to peripheral.

Two types of peripheral devices will be discussed: Terminals and Peripheral or File Memory.

The property which separates a file from a terminal is whether information can be *both* written into and read from the file. That is, the device is capable of both storing and retrieving information. The information stored on the file memory can be utilized in various ways according to other properties of the file.

The terminal serves a different function; that of providing the computer with a path with which to communicate with people, or other machines. A file and terminal may be considered almost identical from a program viewpoint. The terminal is restricted in that information can only be 1) written (reading occurs by some media outside the computer), or 2) read (writing occurs outside the computer), or 3) read or written (e.g., a typewriter can be both read or written by a computer, since the computer cannot read what it has written).

Terminals

Terminals are used to communicate with anything outside the computer and may further be subdivided according to with whom they communicate. The characteristics of the terminals are: information transmission time and form (character or blocks); information format or coding; transmission directions (In, Out, In or Out); and selection or addressing of terminal data, e.g. random, linear or sequential, etc.

Direct Terminals. Direct Terminals provide the human user with a node for direct communication with the computer. These terminals include: typewriters, scopes for display of text or graphical information, audio output devices, telephone input dialing units, and specialized terminals, such as bank teller window consoles, airlines reservations consoles or stock quotation terminals.

Indirect Terminals. Indirect terminals provide a communication path between the human user and the computer, but only via a path which requires off line transformation of information. Information is available at the indirect terminal in only a machine readable form (e.g., holes in a card or tape, or magnetization of an area of tape). A separate, mechanical translation process is required to convert from machine readable to "people readable" form. Indirect terminals include card or paper tape readers and punches, film or photograph readers, specialized format document readers, (e.g., magnetic ink or typewritten), TV cameras, photographic output devices, magnetic tape units, etc.

Machine Terminals. Machine Terminals are those which link other computers, or electrical form devices (such as temperature or pressure transducers, etc.) to the computer. Such a linkage may include the Dataphone, which is a channel or link for transmitting information outside the computer's periphery via telephone channels. Other forms include: analog-digital conversion, and discrete event, time duration, data encoding methods.

A computer is often used as a terminal to the main computer for the following functions:

1. Concentrating or managing a number of typewriter or other terminals on a text line at a time basis.
2. Pre- and post-processing of information on cards, magnetic tape, printers, and plotters.
3. Processing of high data rate terminals for the main computer, as in the case of CRT displays.
4. Connecting to a process of some other kind, e.g., process control, data logging, information collection, etc.

Peripheral or File Memories

These memories lie at the same structural position as terminals. A file's sole function is the storage of information for use by the process (or programs). The parameters which control how a device is to be used in a system are:

1. Cost.
2. Size of memory.
3. Access time and information quantity characteristics. Information selection or access time may be expressed in terms of the following operators:
   - Random — Data selection is a constant and is independent of the address (e.g., core address, drum head selection — generally electronic or optical).
   - Linear (uni-directional) — Data selection time varies proportionately with the address (e.g., tape) required.
   - Linear — same as above except that either direction of information address searching and data transmission is permitted (e.g., disk selection or track arm).
   - Cyclic Linear (or constant rotational) — Data selection time varies proportionally with the address. Addresses are being changed automatically, and take on cyclic values at some rate (e.g., drum).
4. Addressability of information. Some cases include:
   - Files with no explicit hardware addresses.
   - Files with addresses specified by embedded data.
   - Files with explicit hardware address information associated with access mechanism.
5. Replaceability of information. Information space can be recovered by exactly re-writing over existing information, to replace a single part of a file without the need to re-write the whole file.
6. Removeability or portability of information from the computer, i.e., transferability of information off-line among computers. This property provides for information to be removed from the system and stored off line.

The use to which a particular file is put in the system is a function of the above parameters of all storage devices. The present systems have the requirements for the hierarchy: bits, words, word groups (<100-1000 words), program size word blocks (1000-100,000 words), files, and multiple files. The secondary memory functions in the computer can be broken into the following different tasks for which different kinds of file memory can be used.

Program Swapping Memory. Program swapping memory is used for the retention of programs to be placed in primary memory for direct execution by a processor. "Program swapping memory" and "secondary memory" are considered to be synonymous.

Program Swapping, the underlying principle of many time-sharing systems, is the act of keeping programs in secondary, or file memory, until they are ready to be run (as the scheduler decides), and then exchanging them with programs in primary memory so that they may be executed by the processor and primary memory. The secondary memory may also be used to provide the user with the appearance of a large, homogeneous, one-level primary memory, if sufficient memory allocation hardware is provided (see memory allocation, below).

The transfer of data between the two levels of memory should be as near the primary memory speed as possible (still allowing some arithmetic processing). The single characteristic of time to exchange users between primary memory and program swapping memory affects the maximum number of users and their response time for swapping systems.

Fixed head drums or discs are most commonly used for swapping, since only a rotational or cyclic linear access is encountered to select data.

A program swapping device may not be necessary unless the system serves a large number of users. It is also possible to use some slower storage components, (e.g., program file memory), as swap data media. The substitution of one file type for ˋnother allows a system to be built ithout complete component redundancies and still satisfy uptime constraints.

Program File Memory. Program file memory is storage used for user data base and user programs which are not usually in a state to be run. The requirements for file memory necessitate the use of large, relatively fast, addressable storage in which data items can be replaced. The units which are used for this purpose include fixed or moving head drums or discs, magnetic card readers, and magnetic tape (whose data can be both addressed and replaced).

Backup File Memory. Backup file memory is storage which can be removed from the computer, and includes magnetic cards and tape, etc. This memory is used to retain a snapshot or state of the system at fixed intervals so that the state of the system can be re-established in the event of a failure. This hardware file does not require explicit address-ing, or the ability to replace data.

Archival Memory. Archival memory is used to store user files which are removed from the computer. These files exist principally for cost reasons, and the act of retrieving a file from the archives is one of manual selection from a library for which the computer does not have direct access. Magnetic tapes are used for this purpose, since acceptable retrieval time may range from ¼ hour to one day. The files are roughly equivalent to backup storage files.

CONTROL UNITS

The control units have little logical significance in the computer. The controls exist principally because of the cost ratios of control electronics to peripheral devices, and of control electronics to total system costs. It is desirable that all peripherals include controls so that the simultaneous transmission of data from all peripherals is possible.

The functions which the controls perform are:

1. Electrical logic signal conversion. Lines from peripheral devices, e.g., typewriters must have the same electrical characteristics as the computer logic.
2. Time information transformation (Information coding and decoding). The coding of information is an idiosyncrasy of each device, and as such information must be put in a computer compatible form of information.
3. Buffering or assembly of information. Since each device may inherently transfer bit strings which are a sub-multiple of a computer's word, a complete word may have to be formed prior to memory transmission. Very high speed bit rates for the peripheral data can be reduced to acceptable character or word data rates for transmission to memory by parallel data transmission path and buffering.
4. Selection of a specific peripheral from the set which connects with the control. The control retains the switch position information which selects the peripheral.
5. Selection of information within the peripheral. For devices which have information organized in addressable form, the control contains the value of address for the information to be accessed.
6. Error correction and detection.

SWITCHES

A switch provides a communication path between two different component types. Figure 5 lists the switch forms. The specific choice of which switch to use is a function of the allowable switch cost, the time allowed to transmit information through the switch, the number of simultaneous conversations, the number of units among which switching is to occur, and the expected reliability of the switch relative to the components from which it is constructed (together with requirements for partitioning parts of the switch which have failed).

The implication of the switch diagrams is that the switch is set to a particular value, and that information then flows along the switching paths, between the components (or rather between registers of the components). A large part of the switch consists of decision hardware for setting the switch positions. In particular, along a path for which information is to be switched, there exists a dialogue between the transmitting unit, the switch, and the receiving unit. The dialogue is: transmitter broadcasts a request for a dialogue to either one or all switch units; the appropriate switch setting or selection or closure is made; the information is sent from transmitter to receiver, i.e., the information dialogue takes place between the two units while the switch is in a given position; and finally, after the dialogue, the switch is opened. In some cases, the dialogue first consists of additional selection information. For example, in a multiple memory module system: a processor first makes a request for a particular memory module; the particular switch is closed which allows the processor-memory module dialogue to take place (the processor transmits a particular memory address to the memory so that a memory word is selected; the data transmission takes place between memory and processor); and, finally, the switch is opened, or the dialogue is terminated.

MULTI-PROGRAMMING AND MEMORY ALLOCATION HARDWARE

Multi-programming is the simultaneous existence of multiple, independent programs within primary memory being processed sequentially or in parallel by one or more processors. *Time-Slicing* describes the division or allocation of a processor's time among multiple programs prior

TABLE 2. MEMORY ALLOCATION METHODS

Hardware Designation	Method of Memory Allocation Among Multiple Users	Limits of Particular Method
Conventional computer — no memory allocation hardware	No special hardware. Completely done by interpretive programming.	Completely interpretive programming required. (Very high cost in time is paid for generality.)
1 + 1 users. Protection for each memory cell	A protection bit is added to each memory cell. The bit specifies whether the cell can be written or accessed.	Only 1 special user + 1 other user is allowed. User programs must be written at special locations or with special conventions, or loaded or assembled into place. The time to change bits if a user job is changed makes the method nearly useless. No memory allocation by hardware.
1 + 1 users. Protection bit for each memory page.	A protection bit is added for each page. (See above scheme.)	No memory allocation by hardware.
Page locked memory	Each block of memory has a user number which must coincide with the currently active user number.	Not general. Expensive. Memory relocation must be done by conventions or by relocation software. A fixed, small number of users are permitted by the hardware. No memory allocation by hardware.
One set of protection and relocation registers (base address and limit registers). Bounds register.	All programs written as though their origin were location 0. The relocation register specifies the actual location of the user, and the protection register specifies the number of words allowed. (See Fig. 7.)	As users enter and leave, primary memory holes form requiring the moving of users. Pure procedures can only be implemented by moving impure part adjacent to pure part.
Two sets of protection and relocation registers, 2 pairs of bounds register.	Similar to above. Two discontiguous physical areas of memory can be mapped into a homogeneous virtual memory.	Similar to above. Simple, pure procedures with one data array area can be implemented.
Memory page mapping*	For each page (2^6-2^{12} words) in a user's virtual memory, corresponding information is kept concerning the actual physical location in primary or secondary memory. *If the map is in primary memory, it may be desirable to have "associative registers" at the processor-memory interface to remember previous reference to virtual pages, and their actual locations. Alternatively, a hardware map may be placed between the processor and memory to transform processor virtual addresses into physical addresses. (See Fig. 8.)	Relatively expensive. Not as general as following method for implementing pure procedures.
Memory page/segmentation mapping	Additional address space is provided beyond a virtual memory above by providing a segment number. This segment number addresses or selects the page tables. This allows a user an almost unlimited set of addresses. Both segmentation and page map lookup is provided in hardware. (See Fig. 9.) May be thought of as two dimensional addressing.	Expensive. No experience to judge effectiveness.

to the completion of the programs.

Having multiple programs in primary memory may require special hardware for the protection of programs against each other and memory space allocation. Allocation or relocation provides a user address space which is independent of the computer's actual address space.

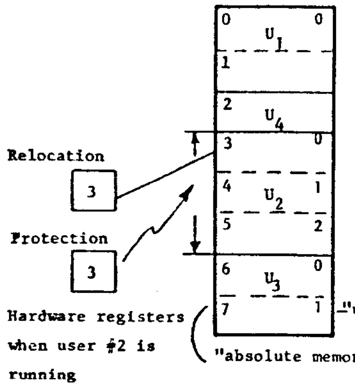
In general, the goal is to effectively provide each user or user's program with a large, continuous memory space as though he were the sole user. A further goal is to provide a method such that any two identical blocks in primary memory would not have to be duplicated. This ability has significance in implementing pure procedures.

A *pure procedure* is the constant or pure or read-only part of a program which has been separated from the variable or data part. Operating systems software (including compilers, assemblers, loaders, editors) is generally written as a set of pure procedures for primary memory conservation.

Unless allocation hardware exists, software may have to carry out this function, in which case, not only is the ability of the system limited, but time is consumed in relocating programs.

Sometimes primary memory is broken into pages of 2^6 to 2^{12} words for hardware allocation. A number of solutions are possible, and Table 2 gives a list of some current schemes. The methods, boundary registers, memory page mapping, and memory page mapping/segmentation mapping are elaborated in Figures 7, 8, and 9.

The *memory map* is part of the user's status information and is generally held in primary memory. The map contains information to transform user's or virtual addresses into physical addresses in primary memory. It may also contain access control information, including whether a page may be read, read as data, written, or read as program.


0
0
U_1
1
2
U_4
3
0
Relocation
3
4
U_2
1
Protection
5
2
3
6
U_3
0
Hardware registers when user #2 is running
7
1
"user-memory" addresses in 1000's of words
"absolute memory" addresses in 1000's of words

User	User Addresses in (1000's of words)	Relocation Register	Protection Register
1	0 < 2	0	2
2	0 < 3	3	3
3	0 < 2	2	2
4	0 < 1	6	1

Table of user location information

Fig. 7. Memory allocation using boundary or relocation and protection register.

PROGRAM INTERCOMMUNICATION

Although intercommunication among the various hardware elements occurs physically along the lines of the hierarchy, the primary memory provides the main communication path between programs. Communication could be via common files. Normally, two programs only communicate occasionally, and hardware must be used to signal when communication is to occur.

Hardware Interrupts or Traps

Hardware interrupts or traps are intra- and inter-processor state conditions which command the processor to begin the execution of another program or process. The number of conditions which can cause independent program starts is a measure of a processor's capabilities, since state change occurs frequently. *Intra-processor traps* occur for the following reasons:

1. Processor malfunction. The self-checking part of the processor has detected an error. (E.g., a memory access has resulted in an error.)
2. Program or process malfunctions. A program has:
• Made an arithmetic error (e.g., divide by zero) which, if continued, will yield meaningless results.
• Made reference to part of a program or data which does not exist or is not available to the program.
3. A timer associated with the processor has signaled that it may be time to do something else.

Intra-Processor Traps for Executive Calls. Hardware instructions are required for efficient intercommunication between the user process and the operating system. The commands for file and terminal activity, and the calling of executive or operating system defined functions is via these special instructions. When they are executed by a user, a trap or interrupt may occur (with a change in status to another mode or process) so that the operating system can carry them out. The limits of requirements of these instructions include: decreasing the time between request and action; increasing the number of permissible command types; allowing flexibility in the call type (e.g., subroutine calling with parameters, provisions for data storage on behalf of a user, and the ability of commands to call other commands or nested calls).

Inter-Processor Traps. Inter-processor communication between both arithmetic-arithmetic, and arithmetic-peripheral processors is also accomplished by trapping. Intercommunication among processors is required using interrupts usually when a processor has completed an assigned task or requires another processor's assistance. For example, peripheral processors do not usually have the ability to decide the number of times the reading of faulty records should be attempted before giving up, or what to do after a set of peripheral processes have been carried out.

HARDWARE WHICH FACILITATES GENERAL PURPOSE TIME-SHARING

Special Modes

Privileged instruction set or executive mode denotes a state when the operating system is running and a privileged set of instructions is being executed by the processor for the operating system software. These instructions would not be allowed by a user when running in *user mode* state. The two distinct states, *user mode-executive mode,* represent a minimum requirement to allow allocation and control of resources.

Executive mode allows the operating software system the freedom to activate any terminal, modify any data location, and, in general, do anything which is within the limits of the hardware. User mode implies a restricted set of abilities for the user: no ability to control a peripheral device; access to only a limited data set; etc. This implies that requests for terminal and file activity are via the operating system software. Other modes may be provided which allow the system to reference a user's data, as though the system were a specific user which facilitates data transmission between user and

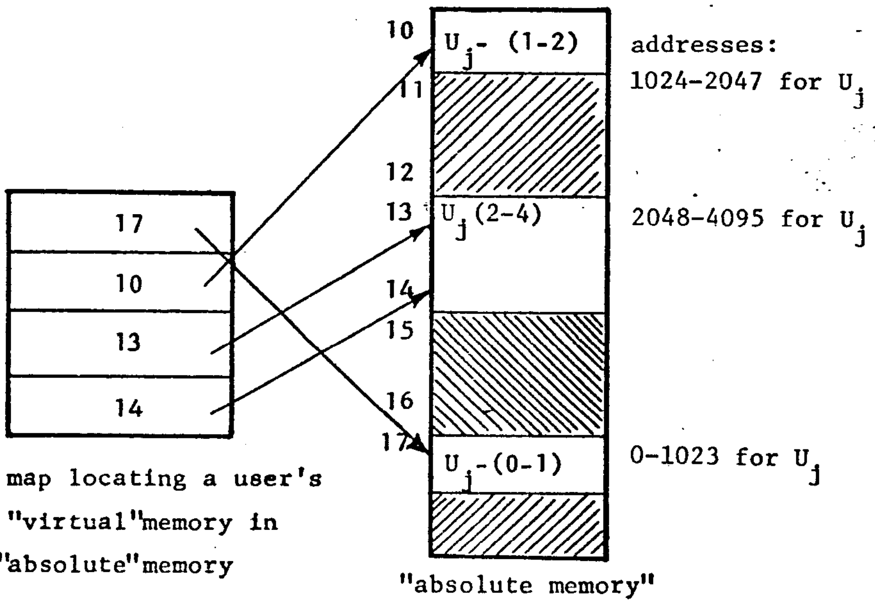
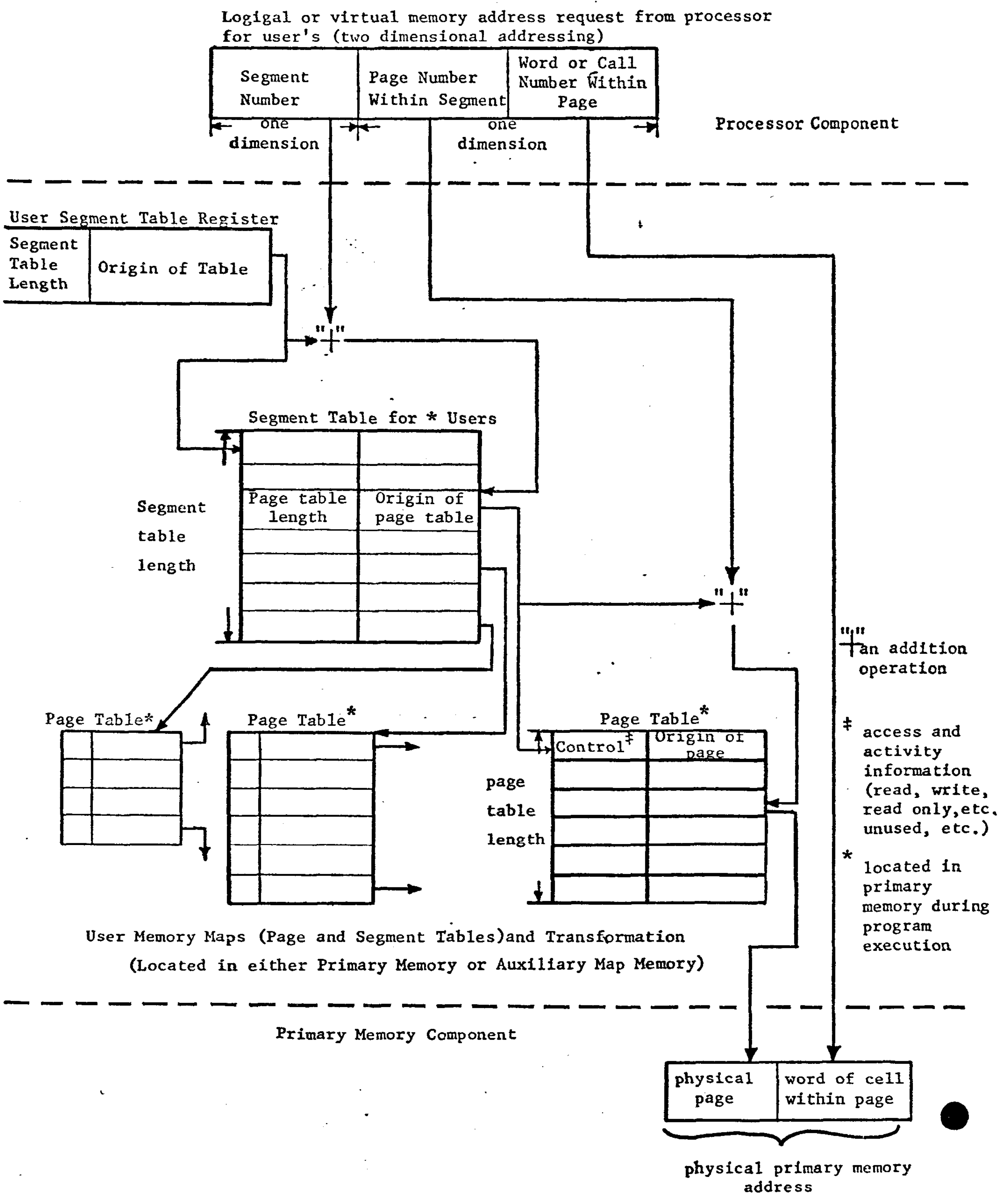


Fig. 8. Memory allocation using page allocation map.



Logigal or virtual memory address request from processor for user's (two dimensional addressing)
Segment Number
Page Number Within Segment
Word or Call Number Within Page
one dimension
one dimension
Processor Component
User Segment Table Register
Segment Table Length
Origin of Table
"+"
Segment Table for * Users
Segment table length
Page table length
Origin of page table
"+"
"+" an addition operation
Page Table*
Page Table*
Page Table*
Control‡
Origin of page
page table length
‡ access and activity information (read, write, read only, etc. unused, etc.)
* located in primary memory during program execution
User Memory Maps (Page and Segment Tables) and Transformation (Located in either Primary Memory or Auxiliary Map Memory)
Primary Memory Component
physical page
word of cell within page
physical primary memory address

Fig. 9. Memory allocation using pages and segments.

system. For example, users interested in specific terminals might directly control them with no system intervention or overhead. In some cases, a user must directly control a device to effectively utilize it. Additional levels of hardware resource allocation also allow peripherals to be added, and program testing to occur concurrently within normal system use.

Time Measurement Hardware

The switching of processors to processes is done by the scheduling part of the operating system. The software requires a clock or interval timer hardware to measure elapsed time. A processor interrupt accompanies the time interval's termination.

Inter-Processor Interlocks and Communication for Multi-Processing

When multiple arithmetic processors execute the same process or different processes which modifies a common data base (e.g., occurs in scheduling or core allocation procedures), it is necessary to provide hardware interlocks. The interlock prevents the simultaneous multi-processor execution by providing a single processor instruction which simultaneously tests *and* conditionally modifies a primary memory cell by setting into an interlock state. In this way, the first processor enters and locks the process by testing and modifying prior to another processor's use. The second processor must wait for the unlocking to occur before entering.

Inter-processor communication to handle faults and share jobs can take place by normal inter-processor traps or interruptions among processors.

User Status Preservation Hardware

The active user's processor hardware registers and status must be preserved as a processor is switched to a new user on the operating system. Hardware or special instructions which quickly save and restore user's status and set up another state are desirable to minimize job switching overhead time. They also may simplify the construction of the software and reduce the number of possible errors.

PROPOSED ADVANTAGES FOR TIME-SHARING OF COMPUTERS

In the following discussion, only the positive aspects of Time-Sharing are given. In emerging new systems, there have been just enough positive results to provide us with the ability to imagine how great Time-Sharing can be. Rather than point out how an on line system allows men to be controlled by computer, or how poorly the present machines, which have been adapted for Time-Sharing, perform, I will list the proposed advantages and suggest them as design aspirations.

In general, Time-Sharing replaces an existing form of processing because it offers to provide a better service or cost less, sometimes it offers to do a job that is difficult using another system. It also opens up new avenues of approach which enable a new class of problems to be attacked fruitfully. It is already changing the structure of programs; maybe because of the system structure, but also because of new hardware which might not have been available without Time-Sharing, (i.e., memory segmentation or two dimensional addresses).

ON LINE ADVANTAGES

The direct terminal (by providing a link between computers and man) forms a symbiotic problem solving system. The symbiotic system offers to provide a more complete problem solving system because of the tight coupling between the two components, and power in each processor's domain. For example, in computer aided design the human user synthesizes while the computer analyzes and optimizes. A circuit designer would suggest circuits while the computer would "breadboard" or analyze them. With configuration determined, the computer would optimize the parameter values. Thus, the reactive nature of the on line or direct terminal provides the user with a very responsive tool with which to probe the problem solution space.

A complete tool is available, including all files which hold a user's data base and his procedures are within the system. The problem in transporting physical data is eliminated. Thus, the necessity and inconvenience of relying on other human systems for the preparation of programs and handling of data is unnecessary. When there is need to create, modify, or destroy a file, the commands are executed quickly.

The total time to make a modification and have another attempt at problem solution, or the problem *turn around time* can be short or appropriate with the task size.

Direct terminal interaction with the system to create and edit files provides a constant monitoring and check on a user's input so that a wide variety of errors can be detected at all levels during the problem solving. That is, data format and validity checking, including the detection of misspelled words occurs at the earliest possible time and lowest level. Clerical functions, including program preparation, drawings, and report generation are part of the system.

Data may be presented in more useful forms to on line users without the need to transfer entire output files to paper. A user may specify only the part of the file or process of interest. More useful forms of data presentation, such as graphs, charts, and diagrams may be presented on displays and plotter.

USER COMMUNITY ADVANTAGES

A general purpose system provides an ever increasing set of procedures for problem solutions, created by its users. Procedures may enter the public domain more rapidly, the author need issue only a notice to the system (which informs other users). Procedures in the public domain become useful more quickly because a large community of users has immediate access to them and incidentally simultaneously checks them. Common or shared data bases (e.g., census data) need only be gathered once and appear in one file.

Routine inter-user administrative tasks such as updating the library, administrative message sending, and availability lists occur at time of origin and are automatically part of the system.

The accounting of resources is by the system with controls imposed by overall human administration. Not only is there better accuracy, but users can be monitored rather than being required to administer their own time. This, in turn, provides better information about the total utility of the system and its users.

A higher level of standardization is possible and can be achieved among users and hence the ease of using the system should improve. Trivial functions which tend to be rewritten (e.g., error handling of messages, lesser used arithmetic functions, the manipulation of characters to form words, etc.) are more likely to be shared because of the ease of sharing.

The possibility for improving the documentation associated with procedures should improve through the ease of documentation and perhaps pressure of the community to share procedures. The overall documentation (text, diagrams, etc.) which describe a process or problem solution may improve.

FLEXIBLE TERMINAL LOCATION

Most direct terminals may be located where they can most efficiently serve the users; in fact, they are even portable. No longer will it be necessary for the user to preschedule time, but he can now use the computer as his tool when and where he best is able to work. For some, this may be in an office, for others a laboratory, and still others, their home. Ultimately, consoles will be in all homes. For example, consider the salesman who has a terminal in his home (or a portable one in his car) such that he can help the computer determine a list of the best calls for that day.

ECONOMICAL ADVANTAGES

In general, a community is provided with a much larger system than any single member could afford. For on line or real time systems, the hardware and software overhead associated with this additional ability can be associated with a larger number of users.

A large number of facilities (coordination of all file activity, transmission of data to terminals, standard error handling, etc.) which are overhead functions are implemented within a system framework rather than repeatedly by each user as he attempts to form his own system. Parallel requests for resources rather than serial processing provide the system with more information to improve scheduling.

Since the system provides the users with the ability to "watch" the execution of a process, the likelihood of using large amounts of processing capability yielding erroneous results is lessened.

If the community of users is sufficiently large, there should be more than one hardware unit of each type, and in the event of hardware failure, the system can be repartitioned to maintain a working system although of lesser performance.

The second and concluding part of this article, which contains an extensive bibliography on time sharing, will be published in the March issue of Computer Design.

5

1-68
Part 2

"Time Sharing" is presented in its most general sense as any application of a computer system that involves simultaneous users. Concepts and equipment of time-shared systems are defined and described and criteria for system configurations are given in terms of application requirements.

FUNDAMENTALS OF TIME-

This is the second and concluding section of the article by C. Gordon Bell which appeared on pages 44 through 59 of the February issue of Computer Design. The first section discussed the hardware of time-shared computers and suggested advantages of time sharing. This section discusses operating system software and user components and includes an extensive bibliography on time sharing.

OPERATING SYSTEM SOFTWARE

Operating system, monitor, supervisor, and *executive* are names given to those processes that supervise and control the operation of the system for all users.

Unlike conventional operating systems that are static, a Time Sharing Operating system is growing and dynamic. New procedures may be added continuously.

The additional languages and facilities have a structure that may have a rather complex operating system as a major part of the language. For example, consider the administration of a teaching program. The program would undoubtedly schedule its users (pupils), and the hierarchy of the whole system would be: the operating system for the entire computer managing a central teaching program to manage all courses managing a course teaching program which would manage all individual users taking the particular course.

The objectives of the system software are:
1. Provide many user functions or facilities with easy-to-use processes.
2. Effective or efficient hardware utilization. Perhaps allow users to utilize the hardware directly. Provide special user services which utilize special hardware.

The criteria for the design might:
1. Meet the requirements for Time-Sharing (computer time and memory space) per user.
2. Provide for flexibility in the operating system using modular construction. Individual components can be independently designed, tested, and modified (or improved). If possible, the system components should be written as user processes.

In general, all systems are constrained by cost considerations. A special system may concentrate on a single objective, while a general system is forced to find a balance between many objectives.

The system software contains:
1. System data base, or information necessary for system management, and management procedures.
2. Resource allocation, control, and management procedures.
3. Common procedures or processes for the users, the library.
4. Miscellaneous elements: System initialization and shut-down; error recovery; file backup; creation of new system; and system debugging.

OPERATING SYSTEM DATA BASE

The operating system requires a large data base that is retained in primary memory and in files. Backup files (copies of files) must be regularly written so that the system can be restarted in a correct state in the event of system failure.

The data for a user include: his memory map or process location, generally found in primary memory while running or active; the processor status (the location counter, processor flags, accumulators, index registers, etc.); identity information (name, number, project numbers, etc.); the time used, allotted, last run, etc.; the run state (e.g., presently running, waiting to run, requiring special service, waiting for file transaction, terminal action, additional memory, etc.); permanent user data to allow the assignment of terminals and file space; accounting information; system temporary storage to enact user requested procedures; and active terminal and file buffering storage.

SHARED COMPUTERS

by
C. GORDON BELL
Associate Professor of Computer Science and Electrical Engineering, Carnegie Institute of Technology. Formerly manager of Computer Design, Digital Equipment Corporation, Maynard, Massachusetts.

In addition to the data base associated with each user there are inherent data associated with system components and resources. These include: hardware status and availability information; terminal names; file directories including descriptors of abilities, modes, etc.; primary memory free space; and file memory free space.

Historical, statistical, and accounting information are also kept, and historical or activity data provide tools for system improvement. They especially aid scheduling and memory allocation as well as indicate the system balance and load.

RESOURCE ALLOCATION, CONTROL AND MANAGEMENT

This responsibility includes: processor time or scheduling; process space (primary memory allocation) and assignment of a process to secondary memory or files; file space; and terminal/process/user allocation and assignment.

The two extreme philosophies that determine the number of users a system can have are "denied access" and "degraded service." "Denied access" provides for a fixed number of users, each of which will obtain a known or worse case response. "Degraded service" provides for more users and the service is at least inversely proportional to the number of active users.

Scheduling

The assignment of processors to processes is *scheduling*. The scheduling algorithms that compute the time a process is to run usually use the following input parameters: previous time used; memory space occupied; status of terminal or file data transmission; expected response time for the user; user information; and number of users.

The priority information available includes the user, his urgency, and willingness to pay. As economically realistic systems that charge for their actual uses come into existence, users will be able to get a broader range of service.

The round robin algorithm runs each user, in turn, for a fixed quanta of time, and when all users have been served, the process is repeated. If any user cannot run because he is waiting for input or output, or halted, he misses a turn. On completion of input or output the user is put at the head of the queue and run (subject to his allotted time).

The scheduling algorithm is a most subjective system component, and, therefore, might be written in a form that can be easily modified. How, when, and which components call the scheduler is also important.

Memory Allocation

Primary/secondary memory allocation occurs as users make demands for more space the system activates user processes. The hardware memory allocation scheme of Table 2 constrains the user map organization, and the process organization. This hardware constrains the user procedure with restrictions ranging from writing in interpretive languages; writing at particular addresses or using a convention determined index register as a base register; writing with no restrictions (over the basic machine); and finally providing a two-dimensional addressing space.

The memory paging-memory segmentation hardware will drastically influence future program structure and design. With two-dimensional addressing, the user is not required to manage primary memory, and is free to address data by two logical numbers rather than by physical numbers. (With such freedom, and ability one might expect a proportional cost.)

File Allocation and Control

File allocation and control are generally subject to extra-system constraints on the basis of user-size-restriction tables.

File allocation cannot easily be separated from detailed file management. The management includes the service of detailed user requests for data, while allocation is concerned with broader control of all file space.

Hardware's View of Files. The hardware parameters that affect file organization are: the hardware access time for words or sectors of the file; the word or record transfer time; the size of the records transferred; the total file size; and the file failure rate.

Operating System's View of Files. The apparent file parameters are: the size of files; the number of users and number of files per user; the access time to segments of a file; the nature of addressing the file information (sequential or random accessing); the file index; and the file data buffering.

File activities can be divided into operations: naming, or declarations, inter-file manipulation, intra-file utilization, and file closing.

User's View of Files. Parameters associated with the directory or index of files for users provide a means of controlling a file's activity, flexibility, general usage, name, users, record of its activity, and actual location of the file components. File accessibility control for the user is on the basis of the originator (owner), group, and public. The modes of file activity include read/write, read only, execute only (a procedure), and denied access. Other information about file access includes creation date, number of times used, last time used, times modified, etc. The user requests of functions for utilization include: reading, writing, naming, re-naming, deleting, appending, inserting, providing access restrictions, obtaining statistical information, or in general, any operation that can be done with the data in or about a file.

Terminal Allocation

Terminal allocation in general systems is either on a first-come-first-served basis or on a completely reserved basis. Requests for terminal reservations are via a control terminal, and as a job is initiated, the terminals required for job completion are requested. The terminal is the means by which a process is

TABLE 2. MEMORY ALLOCATION METHODS

Hardware Designation	Method of Memory Allocation Among Multiple Users	Limits of Particular Method
Conventional computer — no memory allocation hardware	No special hardware. Completely done by interpretive programming.	Completely interpretive programming required. (Very high cost in time is paid for generality.)
1 + 1 users. Protection for each memory cell	A protection bit is added to each memory cell. The bit specifies whether the cell can be written or accessed.	Only 1 special user + 1 other user is allowed. User programs must be written at special locations or with special conventions, or loaded or assembled into place. The time to change bits if a user job is changed makes the method nearly useless. No memory allocation by hardware.
1 + 1 users. Protection bit for each memory page.	A protection bit is added for each page. (See above scheme.)	No memory allocation by hardware.
Page locked memory	Each block of memory has a user number which must coincide with the currently active user number.	Not general. Expensive. Memory relocation must be done by conventions or by relocation software. A fixed, small number of users are permitted by the hardware. No memory allocation by hardware.
One set of protection and relocation registers (base address and limit registers). Bounds register.	All programs written as though their origin were location 0. The relocation register specifies the actual location of the user, and the protection register specifies the number of words allowed. (See Fig. 7.)	As users enter and leave, primary memory holes form requiring the moving of users. Pure procedures can only be implemented by moving impure part adjacent to pure part.
Two sets of protection and relocation registers, 2 pairs of bounds register.	Similar to above. Two discontiguous physical areas of memory can be mapped into a homogeneous virtual memory.	Similar to above. Simple, pure procedures with one data array area can be implemented.
Memory page mapping*	For each page (2^8-2^{12} words) in a user's virtual memory, corresponding information is kept concerning the actual physical location in primary or secondary memory. *If the map is in primary memory, it may be desirable to have "associative registers" at the processor-memory interface to remember previous reference to virtual pages, and their actual locations. Alternatively, a hardware map may be placed between the processor and memory to transform processor virtual addresses into physical addresses. (See Fig. 8.)	Relatively expensive. Not as general as following method for implementing pure procedures.
Memory page/segmentation mapping	Additional address space is provided beyond a virtual memory above by providing a segment number. This segment number addresses or selects the page tables. This allows a user an almost unlimited set of addresses. Both segmentation and page map lookup is provided in hardware. (See Fig. 9.) May be thought of as two dimensional addressing.	Expensive. No experience to judge effectiveness.

intiated and requests for additional terminals, primary memory, time, etc., are made through it. It is the medium for job control.

Resource management deals with servicing user demands after resource allocation has occurred. It is imperative to provide users with a system that requires little or no knowledge of particular device or terminal idiosyncrasies. Even though terminals have differing characteristics it is desirable for the system to provide users with a single basic set of characteristics. More flexible terminals would, of course, leave abilities in access of the common characteristics which could be utilized. On the other hand, it is important to allow users the freedom to control special terminal activity directly. This is particularly necessary in mixed experimental-production systems involving terminals that differ widely. For example, in flight simulation systems, the usage may range from program debugging, new terminal hardware-software debugging, and simulation.

The terminal characteristics are: speed or data rate of the terminal; amount of primary memory used for buffering and the location of the buffers; system overhead time for data requests, including processing time required for the data; and device data acquisition modes, and terminal data usage. Detailed terminal management includes the process that buffers data from the terminal and synchronizer user demands with terminal performance.

SYSTEM-PROVIDED PROCEDURES AND PROCESSES

In addition to providing the software framework within which users operate the hardware, the system also supplies many of the processes for a user. That is, the system includes a library of procedures for arithmetic function evaluation, special and procedure oriented language translations, computer aided instruction, file data conversion, text editing, program debugging, fact retrieval, simulation, etc. In fact, the difference between a user and a system process is that a user process can be altered.

The method of calling these procedures (or job setup) and the ability to have a hierarchy of procedure calls is important. A system-supplied procedure can be considered an extension of the system and called with the same mechanism with which a user would request file or terminal activity. In fact, the hardware instructions that provide communication between the system and the user should also be used for procedure calls. In this fashion, the system can conserve memory space by not providing duplicate copies of routines that are in use by multiple users. The data or temporary storage required by the system while enacting a procedure on behalf of a user is part of the user's memory. This structure conserves space both for users of small subroutines (e.g., arithmetic, data conversion, etc.) and large programs (translators, text editors, etc.).

A set of commands might include programmed floating point arithmetic (for a small system), common arithmetic functions, complex arithmetic, string processing, data conversion and operating libraries for the language translators, translators, editors, loaders, etc. Also desirable is the facility for a user to define and call his own functions in the same hierarchy and framework.

MISCELLANEOUS SYSTEM FUNCTIONS

These processes include record keeping, the periodic recording of the system state for backup, error detection, error recovery, error handling for a device, and communication with the user terminals for system requests.

The system clock is a part of the operating system that provides the actual time base and is used by the scheduler and the accountant, for example, to carry out their functions.

System start-up and shut-down procedures are necessary for initialization of system and the recording of history. Parts of the system can be written as pseudo users. This allows functions like data gathering and system analysis to go on by watching the system rather than being embedded in it. This operation is obtained by defining monitor instructions that allow a user to obtain behavioral characteristics on demand.

A debugging system for the operating system might have the following features: ability to examine or alter; ability to dump or save the complete system in the event of a "crash"; ability to control the substitution of a "new" system for the present one, etc. These features are extensions of a normal on line debugging program.

EXAMPLE OF TIME SHARING SYSTEM FOR THE DEC PDP-6

Figure 10 first presents a simplified view of the system in terms of the memory map of the user and operating system, together with terminals and files. The system runs either as a multi-programming or multi-programming/swapping system depending on whether a secondary memory device is available for program swapping.

A job for a user can be viewed as an area of memory which it occupies while running and I/O

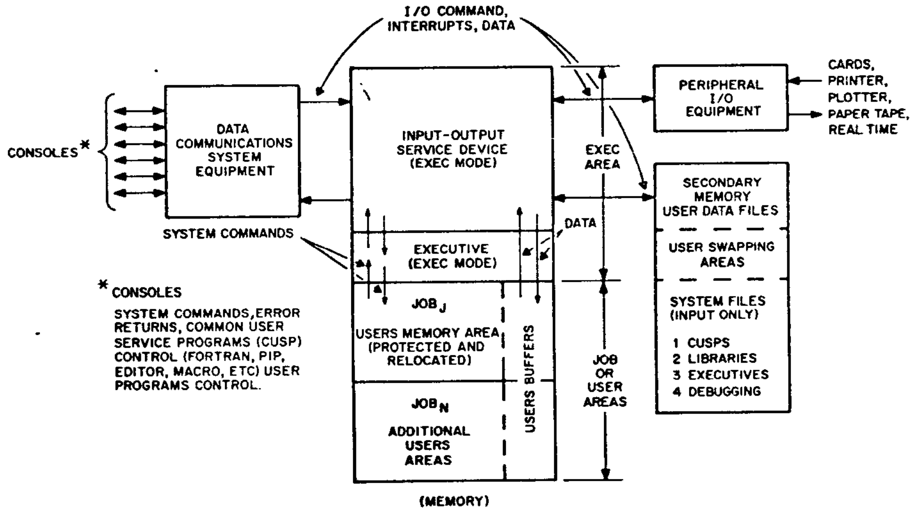

I/O COMMAND, INTERRUPTS, DATA
CONSOLES*
DATA COMMUNICATIONS SYSTEM EQUIPMENT
INPUT-OUTPUT SERVICE DEVICE (EXEC MODE)
EXEC AREA
PERIPHERAL I/O EQUIPMENT
CARDS, PRINTER, PLOTTER, PAPER TAPE, REAL TIME
SECONDARY MEMORY USER DATA FILES
USER SWAPPING AREAS
SYSTEM FILES (INPUT ONLY)
1 CUSPS
2 LIBRARIES
3 EXECUTIVES
4 DEBUGGING
SYSTEM COMMANDS
DATA
EXECUTIVE (EXEC MODE)
JOB J
USERS MEMORY AREA (PROTECTED AND RELOCATED)
JOB N
ADDITIONAL USERS AREAS
USERS BUFFERS
JOB OR USER AREAS
(MEMORY)
* CONSOLES
SYSTEM COMMANDS, ERROR RETURNS, COMMON USER SERVICE PROGRAMS (CUSP) CONTROL (FORTRAN, PIP, EDITOR, MACRO, ETC) USER PROGRAMS CONTROL.

Fig. 10 PDP-6 Multiprogramming System Diagram. (Courtesy of Digital Equipment Corporation.)

equipment assigned to the job, including the user's files and terminals. The operating system software has four main modules: the system files (e.g. FORTRAN, assembler, language translators); terminal control; file control; and the main body of the executive.

Figure 11 gives a more detailed view of what a user program looks like. The user program (e.g., a common user program such as a Fortran Compiler) has its own executive system which communicates with the operating system. The user executive translates user commands from a console into operating system commands for file and terminal activity, while the actual Fortran compiler only accepts input data and produces output data. The user executive is responsible for making it possible for the compiler to read and write files.

Figure 12 shows a memory map of a user's program. The space can grow (and contract) as the program is running, since a user program may make requests to the operating system for space. The first main area, that reserved for operating system parameters, is 140_8 long and is available to both the user and the operating system, although special commands must be given to the operating system to change it. The other areas are a function of what programs are being run.

The system's part of the user's job area contains temporary registers which store the processor state while the job is not running. These include:

1. Two groups of 20_8 registers to store the accumulators or general registers (AC's).
2. The Program Counter (PC) and processor flags.
3. The program's location or boundaries.

The registers that hold the organization to a particular program include:

1. Starting address of the program.
2. Starting address of the debugging program, DDT.
3. Location of various blocks in the user's area, i.e., the symbol table, free storage space, etc.
4. Assignment of I/O device names to numbers, so that a device can be referred to by name rather than on an absolute basis ($2 \times 20_8$ locations).

The registers used as working storage for the system include:

1. The STACK, a pushdown area of temporary storage, and stack pointer.
2. Input-Output data Buffers.
3. Job number.

User requests to the monitor are handled via a defined set of instructions which are called the un-used operation codes, or Programmed Operators, or UUO's. Any time the user program makes a call to the system for service it is via these instructions.

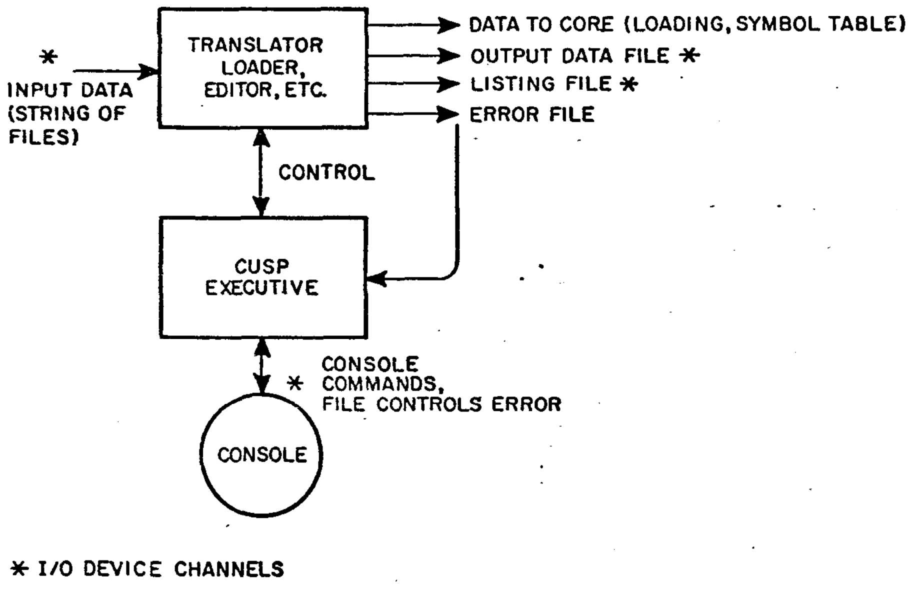

TRANSLATOR LOADER, EDITOR, ETC.
*
INPUT DATA (STRING OF FILES)
DATA TO CORE (LOADING, SYMBOL TABLE)
OUTPUT DATA FILE *
LISTING FILE *
ERROR FILE
CONTROL
CUSP EXECUTIVE
*
CONSOLE COMMANDS, FILE CONTROLS ERROR
CONSOLE
* I/O DEVICE CHANNELS

Fig. 11 General structure of common user service program (CUSP) for PDP-6. (Courtesty of Digital Equipment Corporation.)

The loader is a system routine that is placed in the user area initially and loads the various subprograms required into the user area. The loader links all symbolic references together and fetches needed library programs.

Figure 13 presents a memory map of the operating system which shows the kinds of program modules in it, together with some of the communication paths. The modules perform the following functions:

Job Status Table holds the state of each job in the system, whether a job is in core or residing within a secondary memory prior to running. The state is defined by several words and includes its condition for running, the time it is used, and the location of the job (which includes more status information).

IO Device Service exists for each peripheral device, and the module manages the transmission of data between primary memory and the device, the initiation of the device, and the processing of error or unusual conditions associated with the device (e.g., re-read trys for magnetic tape).

File Directory and File Free Storage Control is used with devices that have named files and directories. It provides the ability to enter new file names and delete files, and it manages the file's free storage.

Error Handling is a common routine that may be called whenever a job (or the monitor) detects an error. A notice of the error is passed on to the user at his console (or to his program), and the job status may be altered.

Run Control is called by other programs and is just concerned with starting and stopping a particular job.

Core Allocation is a common routine responsible for knowing the location of free core in the system and when told, it reserves core blocks.

Clock and Clock Queue are common routines that accept requests for future notification from other parts of the monitor. The clock (more correctly, a timer) notifies the caller at a specified future time.

(For example, the timer is called by the scheduling program so that the scheduler can be activated to schedule the next job.)

Scheduler makes a decision about the number of the next job to run, based on the variables associated with the system's state (each job status, time, core, etc.).

Programmed Operator Dispatcher processes the instructions that are given by the user program to the executive system. The dispatcher looks up the instruction in a directory, does common pre-processing, and passes control to the appropriate part of the monitor. Some of the instructions are defined by a mnemonic call name. A Call table is hash coded with the name, and corresponding monitor address for the processing.

Command Decoder processes console requests and decides the system routine to call.

Console Command Processors include the programs for actually processing the user console requests (or a user program request). These include programs for log in, save job, start, stop, assign a device, etc. Some programs may not be resident, in which case they are loaded and run in a fashion similar to that of a user program.

System Initialization starts the system just after it is loaded, and includes the freeing of devices and the initialization of all variables.

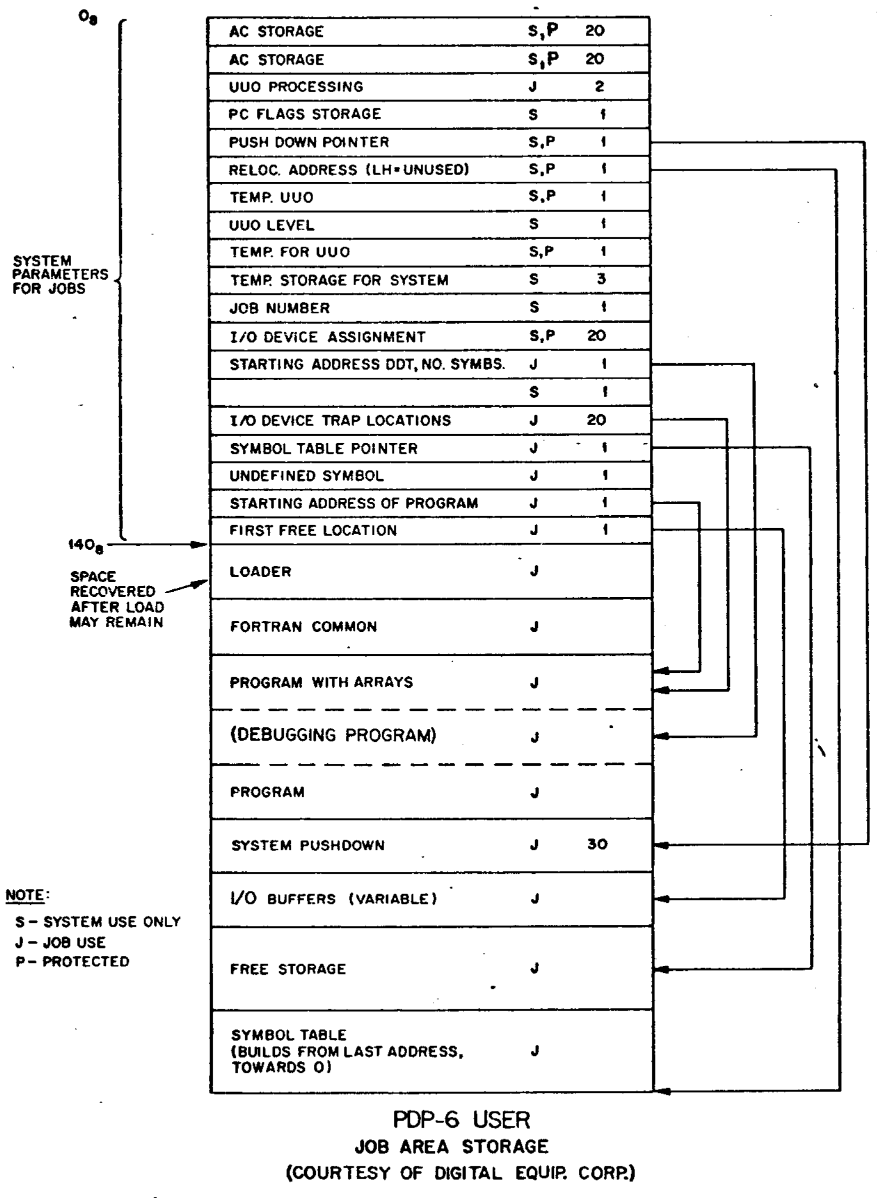


Fig. 12 PDP-6 user job area storage. (Courtesy of Digital Equipment Corporation.)

System Debugging Program is a version of the debugging program, DDT, and may be loaded with the system. It can be used in the event of system failure, to interrogate the state of the system, and includes facilities for preserving the system for future examination.

System Maker allows a complete new monitor to be made as a user program, and when called will copy the new monitor into the area occupied by the old monitor and transfers control to the new monitor.

USER COMPONENTS

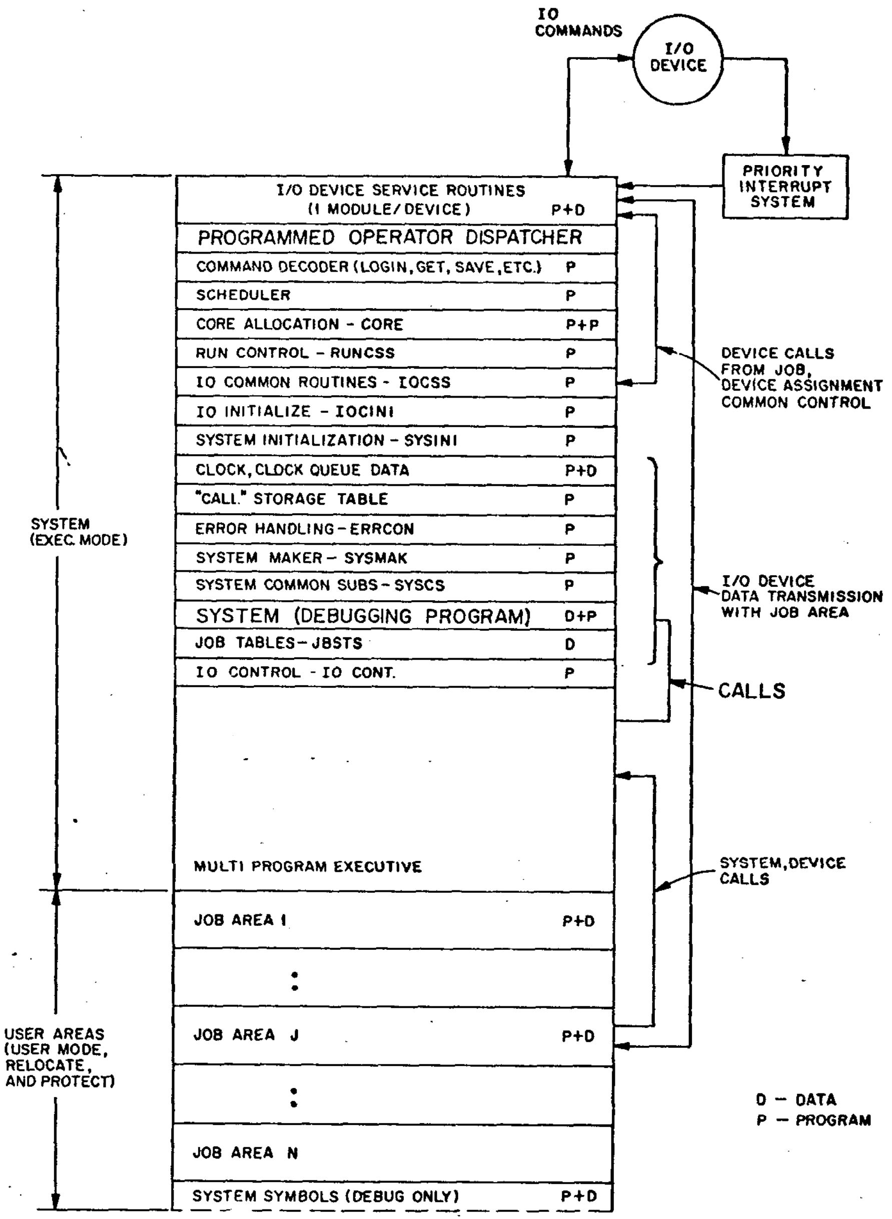
TERMINALS

Communication among the terminal, system software, and user process is very important because of process time, memory space, ease of use, and design modularity considerations. "Human engineering" design aspects include those that affect a user's apparent or actual response.

Although there are many aspects of terminals and their design, the following terminal unit groups will be used:

1. Typewriters.
2. Text—Keyboard Displays. (Text cathode ray tube displays with keyboard inputs)
3. General Graphic Displays or Consoles.
4. Direct Terminals.
5. Indirect Terminals.
6. Specialized Terminals.
7. Machine Links.
8. Peripheral Computers.
9. Other time-sharing systems or computer networks.

The parameters that are common to all terminals and that present the user with certain apparent characteristics have been discussed in the hardware section. The physical data transmission modes, character sets, speed, etc., and general appearance differ among terminals, but the "apparent" characteristics to a user program can be nearly constant, so that user programs can be written independent of their environment or terminals they use. The operating system software is responsible for translating basic user requests into common commands that operate the hardware.



IO COMMANDS
I/O DEVICE
PRIORITY INTERRUPT SYSTEM
I/O DEVICE SERVICE ROUTINES (1 MODULE/DEVICE) P+D
PROGRAMMED OPERATOR DISPATCHER
COMMAND DECODER (LOGIN, GET, SAVE, ETC.) P
SCHEDULER P
CORE ALLOCATION - CORE P+P
RUN CONTROL - RUNCSS P
IO COMMON ROUTINES - IOCSS P
IO INITIALIZE - IOCINI P
SYSTEM INITIALIZATION - SYSINI P
CLOCK, CLOCK QUEUE DATA P+D
"CALL" STORAGE TABLE P
ERROR HANDLING - ERRCON P
SYSTEM MAKER - SYSMAK P
SYSTEM COMMON SUBS - SYSCS P
SYSTEM (DEBUGGING PROGRAM) D+P
JOB TABLES - JBSTS D
IO CONTROL - IO CONT. P
MULTI PROGRAM EXECUTIVE
JOB AREA 1 P+D
JOB AREA J P+D
JOB AREA N
SYSTEM SYMBOLS (DEBUG ONLY) P+D
SYSTEM (EXEC. MODE)
USER AREAS (USER MODE, RELOCATE, AND PROTECT)
DEVICE CALLS FROM JOB, DEVICE ASSIGNMENT COMMON CONTROL
I/O DEVICE DATA TRANSMISSION WITH JOB AREA
CALLS
SYSTEM, DEVICE CALLS
D – DATA
P – PROGRAM

Fig. 13 PDP-6 Multiprogramming System Storage. (Courtesy of Digital Equipment Corporation.)

The typical commands or instructions a user program gives that deal with a terminal include:

1. Assignment of terminal to a process (including the ability to change the name of a terminal, so that programs do not have to address terminals in an absolute sense).

2. Initialization of the terminal to begin transmission, including the declaration of data buffering (number and size), specification of transmission modes, etc.

3. Actual transmission of data (a character, word, buffer, etc., at a time).

4. Termination of transmission, and relinquishing terminal.

Typewriters

Typewriters include both typewriters and Teletypes. The typewriter is the most important because people have been trained to use them. Although harder to use, Teletypes are a common system terminal because they can be used remotely (low bandwidth communication lines), hard copy oriented, low cost, and are available.

Although they are inherently character oriented, it is sometimes desirable to buffer terminal data on a page text line at a time basis or until a special data delimiting key has been struck by the user. (This requires less overhead time from the system to process the characters, since processing is done for each separate line of text rather than for each character of the text.)

It is necessary to allow some form of simultaneous input and output in order that a user can communicate with the system while it is printing, so that a user can stop or change the process. Full duplex Teletypes easily provide this; half duplex Teletypes can accomplish this by a form of "echo checking" during output. Most typewriter consoles must be supplied with special switches or keys to "break" the information output flow so that the user can stop runaway programs, for example.

Keyboard-Text Displays

These devices are similar to the typewriter in principle. The keyboard-text display does not have the hard copy provided by the typewriter (unless the terminal or console also has a printer), but it does provide the viewing of almost a full page of text, together with the ability to "point" anywhere on the page. These displays also require a higher output data rate from a computer in the form of "page turning" requests. This is the principal terminal for systems requiring simple graphical results or rapid scanning of text.

A small cursor, which is controlled by the terminal allows the user to "point" to any character on the page. The data associated with a single page of text is associated with the display.

The control of text displays requires more information processing than other terminals, since data can be randomly addressed by blocks both for input and output, rather than on a strictly sequential basis.

General Graphical Displays

These displays are similar to the text display, but have the added ability to display data by points, characters, lines, circles, etc., and in general have better resolution and are faster.

The information forming the picture may exist in primary memory (as a process or as data for a process) or within the display's own storage. The human eye requires a complete refresh or regenerate cycle about every 30 milliseconds, in which the data forming the picture must be sent to the display. This may impose a high data transmission rate on the memory system, interfering with processing, unless the display has an independent data memory to hold the picture.

For graphical input, a light pen is used to "point" to displayed information. The light pen can be used to "draw" on the scope face. The control and data structure problems of the text display are present to a much higher degree in general graphical displays.

The RAND Tablet is a very simple graphical input device. It allows one to draw on a 10″ × 10″ tablet with a stylus, and it can allow free hand drawing, printed character input, or curve tracing (through paper). It may be used independently or in conjunction with a graphical display. The resolution or number of electronically independent points over the 10″ × 10″ area corresponds to 1024 × 1024 points.

Plotters

These devices provide hard copies of general graphical data. Typically, a plotter operates on an incremental or discrete basis (0.01 inches/increment) at a rate of 300 points/second over a plotting area of 12-30 inches by several hundred feet.

Direct Terminals

The above terminals are special cases of direct terminals, but in them most of the problems of terminal hardware and software design can be seen. Namely, problems of providing continuous two-way dialogue, response time, and the other human engineering problems.

Indirect Terminals

These terminals include most terminals used by other systems, i.e., peripheral card readers and line printers. The interface from a user's viewpoint can be identical to the above terminals. The logical difference, for example, between a line printer and a typewriter printer may just be the number of allowable characters on a line; thus, a page output on a line printer would appear identical to that of a typewriter (but not vice versa).

Specialized Terminals

These terminals are used for special time-sharing systems such as airlines reservations, etc. They include: banking teller windows, airline reservation stations, stock quotation inquiry keyboards, production line data acquisition terminals, etc. They provide the best possible coupling between the user and his system and are designed to minimize the number of errors and the time required as data is entered and extracted from the terminal by restricting the format and by encoding the information.

Inter-Machine Links

The link to specialized "non-human user" devices imposes the highest performance requirements on the design because the data transmission rate is high and is determined by the device characteristics, rather than the system. That is, these devices have to be served in real time, at the demands of the device. Devices of this type include those used in process control applications, simulation equipment (aircraft or aerospace cockpits), film reading devices or scanners, hybrid linkages, etc.

By providing for this equipment in a system, hardware protection may also be required. A very complete interrupt or trap system may also be necessary in the hardware so that a job can be rescheduled rapidly to serve the device.

Peripheral Computers

These form a most necessary class of terminals by distributing terminal data transmission or loading to the system periphery. The peripheral computer provides the ability to lower the data rate for a larger system by providing local storage and processing capability. For example, display computers with the ability to detect light pen position and track the pen, and perform

TABLE 3. TERMINAL INPUT REQUESTS TO SYSTEM SOFTWARE

MESSAGES TO THE OPERATING SYSTEM:

1. Log in and log out. (Includes presentation of name, number, password, data, etc.)
2. Resource requests (assignment of terminals, primary memory, file space).
3. Setup of the job, or process.
4. Start, stop, and continuation of a process.
5. Examination and modification of elements of the primary memory process. (Presentation of a storage or memory map.)
6. Information requests:
   a. Run time, time of day
   b. Files used or space available
   c. Facts about system use.
7. Communication with other users or human operators.
8. Saving and restoring the complete state of a process.
9. Transmission of a job to a queue for batch processing.

MESSAGES TO EDITORS:

1. File name declarations including specification of access restrictions, formats, etc.
2. Transmission of data among files and/or terminals.
3. General file editing including creating, appending, inserting, modifying, deleting, etc.

MESSAGES TO TRANSLATORS:

1. File specifications including:
   a. Control statements.
   b. Source language inputs.
   c. Object output.
   d. Object listing.
   e. Object linkage information (if separated from output).
   f. Errors and diagnostics.
2. Control switches (e.g., what to do in case of errors).

MESSAGES FOR PROGRAM DEBUGGING:

Command messages to system debugging routines are similar to the system commands, except that they are in terms of the source language program. They include:

1. Start, stop, and continuation of the process.
2. Examination and modification of the process in terms of the source language. Insertion of program patches. Display of data in any format.
3. Data set searching.
4. Program tracing.
5. Conditional tracing via breakpoints which are executed only if program reaches a specific state.

MESSAGES TO SYSTEM OPERATORS (HUMAN) AND MANAGEMENT (HUMAN)

1. Equipment availability or status information.
2. Configuration specification.
3. Accounting and system status requests.
4. Appending user availability, cost, facility, priority lists.
5. Message broadcasts.
6. Manual instructions for tape mounting, card removal, etc.
7. System diagnostic reports.
8. Control of back-up or archival storage.

MESSAGES TO CONVERSATIONAL LANGUAGES

1. Language or Text Edit commands. Creation, modification, and deletion of programs is provided.
2. Direct Statement Commands Execution. For languages which allow arithmetic statements to be written, the ability to have a statement executed immediately (e.g., 2 + 2 = ?) is provided.
3. Commands for Control of the Programs.
4. Data entry and data output from the program.

some coordinate transformations on the display data may be desirable.

In process control applications, data sampling, limit checking, and data logging can be done by peripheral computers, on a more economical basis, since they do not require the generality of a large machine. Also, since the overhead time to switch to another program may be high, the high data rates associated with these processes would degrade the large machine.

External Time-Sharing Systems

These terminals form the link with other time-sharing systems. This form of intercommunication is new, but may be significant in total problem solving systems by allowing programs in one system to call on other systems.

Message switching centers with some local file storage might form the immediate link with users. As users require more advanced services, the switching centers would likely call either large, general systems or systems specializing in a particular service. Because of our geographical time zones, inter-system load sharing is possible in a fashion similar to that in which utilities share electrical generation capacity.

TERMINAL COMMUNICATION WITH THE OPERATING SYSTEM

In addition to the terminal connection with the process, a terminal must connect with the operating system software for the control of the job. All of the programs (translators, editors, loaders, etc.) that form the system also require control words or statements. Table 3 lists the information required from the user to specify tasks for the system.

Communication Dialogue

The format used for control information is an important design consideration, and it is important to have a "forgiving system," or one which does not affect a user too adversely when a wrong command is given.

It may be important that the user react (type in, observe output, etc.) as little as possible to specify a given situation. Abbreviated commands might be permitted in place of longer words (e.g., LOGIN = LI), although longer commands would also work. For example, two interesting possibilities are: a user types a command that has enough information to make the command unambiguous, and, the user types enough information to make the command unambiguous, followed by the system typing the rest of the command in a "ghost-like" fashion. When commands are given that irrecoverably affect files, the system might require some sort of verification that the command specified is actually desired.

User defined macro commands compose the most general method to provide users with the commands they want, and what they call the commands, because users define, name, and write them in terms of standard sets of system commands.

FILES

It is desirable to consider the file and terminal structure in a similar fashion from both a user and system software viewpoint; that is, the access, method of transmitting data, and data formats may be nearly identical for both files and terminals.

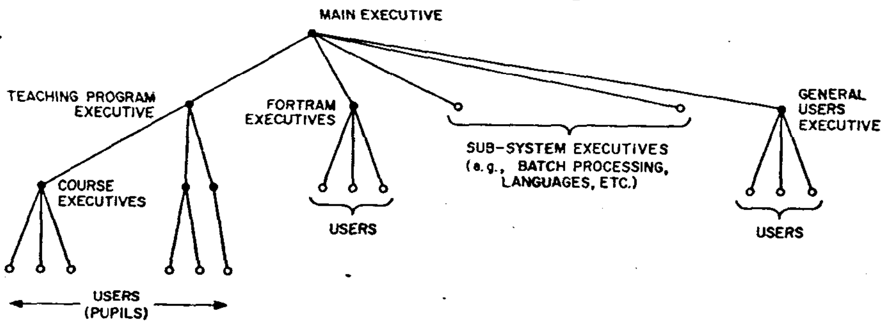


Fig. 14 Hierarchy of executives with a general purpose time-sharing system.

The file characteristics have been previously discussed as part of the operating system software in terms of what the hardware is, what the operating system provides, and what the file looks like to a user.

USER PROCESS

The user process or procedure includes: a memory map locating the process, the actual process, and user status information (terminal and file assignments).

Occasionally, a guaranteed service must be made available to a user both for specialized devices, and processing. For example, a user may have a particular terminal that requires service at regular intervals. A protected, assignable command subset to control the particular device may be required. Alternatively, control can sometimes be provided by incorporating the device in the normal system peripheral or input-output service programs. Scheduling of users now becomes more complex, since the device anomalies constrain the scheduling algorithm.

Guaranteed processing capabilities are provided by treating the total processing capacity as a resource. Thus, a guaranteed capacity at a guaranteed time can be scheduled according to request. Users of systems may get degraded service rather than be denied access because of poor service. With a supply of unattended jobs to process in a batch queue, or compute-bound problems to run as background, a combination denied/degraded service may be provided which balances the system's capacity.

The methods of communication with the system through a hierarchy of higher level operating systems pose the questions: "What is the user process?" and "What is the system?" A user's procedure may be appended to the system and become a system function or common user service procedure. This ever expanding set of program segments which form the system present the problems of segment naming, file location within the system, and protection while they are being run. Nevertheless, the ability to run normally while creating and testing other parts of a system, or to have a portion of the system removed and another one substituted gives rise to very powerful tools in the graceful creation of the system. As a minimum, a new system should be able to be created on a general purpose system, with the substitution for the existing system occurring at a time when the system is inoperative. We can look forward to complete systems that allow subsystems that do their own scheduling of time, etc., and allocate some resources. Thus, a completely general purpose system might allow complete freedom to incorporate any of the systems described in Table 1 in an efficient manner. Figure 14 shows the relationship processes might have to one another in a general purpose system.

CONVENTIONAL VERSUS CONVERSATIONAL LANGUAGE PROCESSING

Conventional processing or translation of a language occurs in the sequence:

1. Creation of a text format source file (cards or system file) which describes the process.
2. Translation of source files into object files with linkage, relocation, subroutine, listing, and error information.
3. Loading the object file together with library files to form the process.
4. Process execution.

In contrast, conversational language processing provides nearly simultaneous creation and execution of procedures. The input language can be checked at the time of entry at the terminal and is translated, being immediately available for execution.

The data may be transformed into an interpretive form with all sub-routines, linkages, etc., occurring directly on input with no intermediate files. The insertion of additional statements or program steps is done directly, and debugging is through the run time diagnostics and user abilities to examine variables directly and execute statements conditionally. The conversational system may require a slightly longer execution time, but is most effective because of its combined editor, translator, loader, library and debugging system. Clearly, for problems involving little computation, the turn-around time is very short for solving problems in this fashion. The main structure of programs is such that this interactive approach may be the common method in a few years.

Batch Processing

This is one of the most efficient methods of controlling the execution of a large number of programs, since jobs are always run to completion. In a time-sharing system which is principally serving on-line users, the batch process can be used as a background job or to absorb spare capacity. A fixed or guaranteed amount of processing can be allocated to batch processing. The batch must be able to be loaded by either external users with card decks or users who defer jobs that can be done anytime (or at batch convenience).

The handling of a batch need not be incorporated within the system, but rather a batch process can

TABLE 1. CAPACITY REQUIREMENTS FOR TIME-SHARING SYSTEM APPLICATIONS

Specialized System Service, or Application	Primary Memory for Process (in bits)	Primary Memory for User Data (in bits)	Processing Capacity/User (in operations*/interaction†)	File Organization and Size (10^6-10^9 bits)	Direct Terminals
Desk calculator	very small	very small ($<10^3$)	very small ($>10^4$)	none	typewriter, input keyboard, strip printer, scopes, audio output, or special console.
Stock quotation	small	small ($<10^4$)	very small ($>10^4$)	one (small-medium)	see above, stock ticker tape or transactions input, telephone.
Airline reservations	medium	small ($>10^4$)	small ($>10^5$)	approx. 6 (medium-large)	special consoles, typewriters, scopes.
On line banking	medium	small ($>10^4$)	small ($>10^5$)	approx. 10 (medium-large)	see above, special bank teller consoles.
General conversational computational languages (JOSS, CULLER-FRIED System)	medium	small-very large (10^3-10^5)	small-large unbounded (10^4-$>10^5$)	multiple files per user, with few file types (medium-large)	typewriter, printer, scope, plotter. (Culler-Fried consists of scope, keyboard, and tablet.)
Specialized computer aided design, engineering, problem solving languages (COGO, etc.)	medium-large	small-very large (10^3-10^5)	small-very large (10^4-$>10^5$)	see above	see above
Process control	medium-large	medium ($>10^5$)	small-very large (10^4-$>10^5$)	few (small)	physical quantity transducers, general user terminals.
Text editing (Administrative Terminal Service)	medium	small ($>10^4$)	small (10^4-10^5)	multiple single purpose files/user. (medium)	typewriter, printer, scope.
On line information retrieval of periodical headings, bibliographies, keywords, abstracts	medium-large	medium ($>10^5$)	medium (10^5-10^7)	one (very large)	see above. telephone (dial in, audio out)

*assumes a fairly sophisticated processor and instruction set
†maximum interaction intervals for user requests are $\simeq$ 10 sec.

be regarded as a special user. Thus, a common service program (the batch manager) would permit any user to "batch process."

CONCLUSIONS

PRESENT PROBLEMS

Before widespread time-sharing systems and system networks can be formed, standardization of data and file format descriptions will have to occur. Simple conventions must be established to control the actual format of the bits transmitted between computers. This will enable the transmission of problems, data, and procedures between systems. Present intersystem communication experiments should provide a framework for the standardization of information interchange formats, and detailed data representation.

Once a data representation for higher speed lines is established, it will be possible to remove the terminals we presently associate with the computer outside the computer's periphery. This will enable the cross-use of terminals among computers. It will also allow software that is more independent of the peripheral and computer to be written.

Current data transmission costs for the remote typewriter user (with an average input rate of ten bits per second) do not reflect the true cost-capacity (2400 bits per second for a voice grade line) or use of the line.

Although good, low cost computers (processor, memory, and minimum peripheral equipment) are available, the higher costs associated with file storage for smaller systems do not permit the design of low cost time-shared computers.

Present time-sharing structures for computers are extension organizations of the basic computer. Present systems were not initially designed for time-sharing, but were modified slightly to accommodate potential users. Hence, these systems create almost as many problems as they solve. A more reasonable approach for a system's design is an initial specification that includes Time-Sharing as a goal. A solution might take on the form of a network. For example, the very large computing machines that are built by computer manufacturers have: taken a long time to build (and technology has changed, invalidating industry's extrapolations before the computers were operational); required longer than expected to become operational; failed to meet initial design goals, have been uneconomical from a production standpoint; and only a few systems have been built. The current large, very general systems also suffer from the same kind of design thinking.

Each component of a general purpose time sharing system is constrained to supply such general service that the system as a whole may be so inefficient (and expensive) as to make the system impractical. The issue is similar to an organization consisting of either highly trained specialists or generalists. An organization of generalists is very flexible; but, on the other hand, it may not be economical to have people who are capable of being the president doing all the tasks within an organization. The general purpose systems just now becoming operational are constructed in such a flexible fashion as to probably be uneconomical. Each system component is so general (for example, the filing system) that, although it can perform any task (given enough time), the act of doing very trivial operations requires a great deal of time. Perhaps a better approach is to divide the systems's resources by allowing several independent operating systems to care for them (e.g., editing, assembling, filing, translating, and running).

FUTURE SYSTEMS

Future computers will be equipped with hardware to allow some form of time-sharing. For smaller computers, the additional hardware greatly enhances a system's utility, especially when being used in process control and in research requiring the direct links with other machines or to experimental equipment.

The form of Time-Sharing Computers will be:

1. The system with a single general user or batch process, *plus* one fixed job or a fixed multi-terminal community of special users (1+1, or 1+n special users). Process control and on-line special business data processing systems take this form.
2. Dedicated special systems which service a particular user community. These provide little or no communication with other systems. (E.g., library, airlines reservations, etc.)
3. Dedicated systems with switching ability so that a problem that requires other aids can be referred to other systems. More general systems may refer problems to them.
4. Message switching for other systems. These may have file processing, editing, and limited calculation capability, or message buffering; such a system would communicate with other systems for most demands from users.
5. Peripheral computers that service special terminals and control small local processes. Processing capacity for general purpose problem solving, file storage, program translation, and diagnostics for the peripheral system would be derived from a higher level system.
6. The totally general system with a large community of users. The general system would undoubtedly communicate with other systems.

Although the author has attempted to be objective, it is felt that the technique of computer Time-Sharing is a significant advance toward an effective use of computers. Time-Sharing removes one more restriction in computer usage — that of allowing only a single use of a machine. As such, the additional generality creates opportunities, as well as countless problems.

BIBLIOGRAPHY

1. Adams, E. N., "Reflections on the Design of a CAI Operating System." *AFIPS Conference Proceedings, Spring Joint Computer Conference* Vol. 30, 419–424 (1967).

2. Allan, Pryor T. and R. Warner Homer, "Time Sharing in Biomedical Research," *Datamation* Vol. 12 (4) (April 1966).

3. Amdahl, G. M., "New Concepts in Computing Systems Design," *Proc. IRE* Vol. 50 (5), 1073–1077 (Memory Protection) (May 1962).

4. American Management Association, "Advances in EDP and Information Systems," *AMA Report* No. 62.

5. Anderson, J. P., S. A. Hoffman, J. Shifman, and R. J. Williams, "D-825 — A Multiple Computer System for Command and Control," *Proc. Fall Joint Computer Conference* Vol. 25, 86–96 (December 1962). See also *D-825 Manual* — Burroughs Corporation.

6. Arden, B. W., et al, "Program and Addressing Structure in A Time-Sharing Environment," *Journal of the Association for Computing Machinery* Vol. 13 (1), 1–16 (January 1966).

7. Aschenbrenner, R. A., M. Flynn, G. A. Robinson, "Intrinsic Multiprocessing," *AFIPS Conference Proceedings, Spring Joint Computer Conference* Vol. 30, 81–86 (1967).

8. Avakian, Emik A. and F. Walter Jenison, Jr., "Voice Response and Visual Display Techniques for On-Line Information Handling Systems," The Bunker-Ramo Corporation, Stamford, Connecticut.

9. Bachman, C. W. and S. B. Williams, "A General Purpose Programming System for Random Access Memories," *Proc. Fall Joint Computer Conference* Vol. 26, 411–422 (1964).

10. Baldwin, F. R., W. B. Gibson, and C. B. Poland, "A Multiprocessing Approach to a Large Computer System," *IBM Systems Journal* Vol. 1, p. 61 (Sept. 1962).

11. Bauer, Walter F., "Why Multi-Computers?," *Datamation* Vol. 8 (9) (Sept. 1962).

12. Beckman, F. S., F. P. Brooks, Jr., and W. J. Lawless, Jr., "Developments in the Logical Organization of Computer Arithmetic and Control Units," *Proc. IRE* Vol. 49 (1), 53–66 (January 1961).

13. Bell, G. and M. W. Pirtle, "Time-Sharing Bibliography," *Proc. IEEE* Vol. 54 (12), 1764–1765 (December 1966).

14. Belluardo, R., R. Gocht, G. Paquette, "A Time Shared Hybrid Simulation Faculty," UAC, East Hartford, Conn., *AFIPS* Vol. 28 (1966).

15. Bitzer, P. L., and H. G. Slottow, "The Plasma Display Panel — A Digitally Addressable Display with Inherent Memory," *AFIPS Conference Proceedings, Fall Joint Computer Conference* Vol. 29 (7–10), 541–548 (1966).

16. Boilen, S., "User's Manual for the BBN Time-Sharing System," Bolt, Beranek, and Newman, 50 Moulton St., Cambridge, Mass.

17. Bolt, Richard H., "Man-Machine Partnership in Intellectual Pursuits: A Look Ahead," Publication No. 1191, Printing and Publishing Office, National Academy of Sciences.

18. Brillouin, Leon, "Science and Information Theory," Second Edition, Academic Press, Inc. (1962).

19. Brooks, F. P., Jr., "A Program Controlled Program Interrupt System," *Proc. Eastern Joint Computer Conference*, 128–132 (December 1957).

20. Buchholz, W. (Editor), "Planning a Computer System — Project Stretch," McGraw-Hill Book Company, Inc., New York (1962). (See also *IBM 7030 (Stretch) Manual*).

21. Burks, A. W., H. H. Goldstine and J. von Neumann, "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument," (reprinted) *Datamation*, 24–31 (September 1962).

22. Burroughs Corporation, "The Descriptor," Burroughs Corporation (1961).

23. Burton, A. J. and R. G. Mills, "Electronic Computers and Their Business Applications," London, E. Benn (1960).

24. Bush, Vannevar, "As We May Think," *Atlantic Monthly* Vol. 176, 101 (July 1945).

25. Calingaert, P., "System Performance Evaluation: Survey and Appraisal," *Comm. ACM 10* (1), 12–18 (January 1967).

26. Carnegie, "Carnegie Institute of Technology Computation Center Users Manual."

27. Castle, C. T., "Planning the 3600," *Proc. Eastern Joint Computer Conference*, 73 (December 1964). See also CDC-3600, *Datamation*, 37–40 (May 1964).

28. Clippinger, Richard F., "Programming Implications of Hardware Trends," *IFIP Congress, New York* Vol. 1, 207–212 (1965).

29. Codd, E. F., "Multiprogramming Scheduling," *Comm. ACM* Vol. 3 (6) (June 1960).

30. Codd, E. F., "Multiprogramming Stretch: A Report on Trials," *Proc. IFIP Congress, Munich*, 574, North Holland Publishing Co., Amsterdam (Aug. 27 to Sept. 1, 1962).

31. Coffman, E. G., "A General Flow Chart Description of the Time-Sharing System," *SDC TM-1639/000/00* (Dec. 12, 1963).

32. Comfort, W. T., "A Computing System Design for User Service," *Proc. Fall Joint Computer Conference, Las Vegas, Nevada*, Vol. 27 (Nov. 30, 1965).

33. Computer Research Corporation, "Time Sharing System Scorecard, No. 1," Computer Research Corporation, 747 Pleasant St., Belmont, Mass.

34. Conway, M. E., "A Multiprocessor System Design," *Fall Joint Computer Conference* Vol. 24, 139–146 (1963).

35. Cook, P. A. C., "Real-Time Monitoring of Laboratory Instruments," *AFIPS Conference Proceedings, Spring Joint Computer Conference* Vol. 30, 779–782 (1967).

36. Coons, S. A., "An Outline of the Requirements for a Computer Aided Design System," *1963 Spring Joint Computer Conference*, 229–304.

37. Corbato, F. J., et al, "The Compatible Time-Sharing System: A Programmer's Guide," M.I.T. Press, Cambridge, Mass. (1963).

38. Corbato, Fernando J., M. Merwin-Daggett, and R. C. Daley, "An Experimental Time-Sharing System," *AFIPS Conference Proceedings* Vol. 21, 335–344 (Spring 1962).

39. Corbato, F. J., V. A. Vyssotsky, "Introduction and Overview of the Multics System," *Proc. Fall Joint Computer Conference*, Las Vegas, Nevada (Nov. 30, 1965).

40. Crisman, P. A., Editor, "The Compatible Time-Sharing System," *A Programmer's Guide*, 2nd edition, M.I.T. Press, Cambridge, Mass. (1965).

41. Critchlow, A. J., "Generalized Multiprocessing and Multiprogramming Systems," *AFIPS Conference Proceedings, Fall Joint Computer Conference* Vol. 24, 107–126 (1963).

42. Culler, G. J. and B. D. Fried, "The TRW Two-Station, On-Line Scientific Computer: General Description," *Computer Augmentation of Human Reasoning*, Washington, D. C., June 1964, Spartan Books, Washington, D. C. (1965).

43. Daley, R. C. and P. G. Neumann, "A General Purpose File System for Secondary Storage," *Proc. Fall Joint Computer Conference, Las Vegas, Nevada* Vol. 27 (Nov. 30, 1965).

44. Dartmouth, "The Dartmouth Time-Sharing System," Computation Center, Dartmouth College (Oct. 19, 1964).

45. *Datamation*, "A Survey of Airline Reservation Systems," p. 53 (June 1962).

46. David, E. E., Jr. and R. M. Fano, "Some Thoughts About the Social Implications of Accessible Computing," *Proc. Fall Joint Computer Conference, Las Vegas, Nevada* Vol. 27 (Nov. 30, 1965).

47. Dearden, John, "Can Management Information Be Automated," Harvard Business Review (March–April, 1964).

48. Denning, P. J., "Effects of Scheduling on File Memory Operations," *AFIPS Conference Proceedings, Spring Joint Computer Conference* Vol. 30, 9–22 (1967).

49. Dennis, J. B., "A Multiuser Computation Facility for Education and Research," *Communications of the Acm* Vol. 7, 521–529 (Sept. 1964).

50. Dennis, J. B., "Segmentation and Design of Multiprogrammed Computer Systems," *IEEE International Convention Record, Institute of Electrical and Electronic Engineers*, New York, Vol. 13 (3), 214–225 (1965); and *JACM* Vol. 12 (4), 589–602 (Oct. 1965).

51. Dennis, J. B. and E. L. Glaser, "The Structure of On-Line Information Processing Systems," *Proceedings of the Second Congress on Information* Systems Sciences, 1–11, Spartan Books, Washington, D. C. (1965).

52. Dertouzos, M. L. and H. L. Graham, "A Parametric Graphical Display Technique for On-Line Use," *AFIPS Conference Proceedings, Fall Joint Computer Conference* Vol. 29 (7–10), 210–210 (1966).

53. Desmonde, William H., "Computers and Their Uses," Englewood Cliff, New Jersey, Prentice-Hall (1964); "Real Time Data Processing System — Introductory Concepts," Englewood Cliff, New Jersey, Prentice-Hall (1965).

54. Digital Equipment Corporation, Maynard, Mass., "Multiprogramming System Manual for PDP-6," DEC-6-EX-SYS-UM-IP-PREOO.

55. Dudas, J. F., "Concurrent Processing of Teletype Message Switching and Order Entry at Westinghouse Tele-Computer Center," Westinghouse Electric Corporation.

56. Duffy, G. F. and W. D. Timberlake, "A Business-Oriented Time-Sharing System," IBM, SDD Poughkeepsie, *AFIPS, Spring Joint Computer Conference*, Vol. 28, 265–275 (1966).

57. Dunn, T. M. and J. H. Morrissey, "Remote Computing — An Experimental System, Part 1: External Specifications," — J. M. Keller, E. C. Strum, and G. H. Yang, Part 2, *Proc. Spring Joint Computer Conference* Vol. 25, 413–443 (1964).

58. Eckert, J. P., J. C. Chu, A. B. Tonik, and W. F. Schmitt, "Design of UNIVAC — LARC System I," *Proc. Eastern Joint Computer Conference* (16), 59–65 (1959).

59. Edwards, J. D., "An Automatic Data Acquisition and Inquiry System Using Disk Files," (Lockheed Missiles and Space Co.), Disk File Symposium, March 6–7, 1963 (Informatics, Inc. Culver City, Calif.)

60. Evans, D. C. and Leclerc, J. Y., "Address Mapping and the Control of Access in an Interactive Computer," *AFIPS Conference Proceedings, Spring Joint Computer Conference* Vol. 30, 23–32 (1967).

61. Fano, Robert M., "The MAC System: The Computer Utility Approach." *IEEE Spectrum* Vol. 2, 56–64 (January 1965).

62. Fine, G. H., C. W. Jackson and P. V. McIsaac, "Dynamic Program Behavior Under Paging," *Proc. ACM 21st Conference,* 223–228.

63. Flynn, Michael J., "Very High-Speed Computing Systems," *Proc. IEEE* Vol. 54 (12), 1901–1909 (December 1966).

64. Forgie, R. W., "A Time- and Memory-Sharing Executive Program for Quick Response On-Line Applications," *Proc. Fall Joint Computer Conference, Las Vegas, Nevada* Vol. 27 (Nov. 30, 1965).

65. Fotheringham, J., "Dynamic Storage Allocation in the Atlas Computer," *Comm. ACM* Vol. 4 (10), 435–436 (Oct. 1961).

66. Frankovich, J. M. and H. P. Peterson, "A Functional Description of the Lincoln TX-2 Computer," *Western Computer Proceedings,* 146 (1957).

67. Fredkin, Edward, "The Time-Sharing of Computers," *Computers and Automation* Vol. 12 (11) (Nov. 1963).

68. Gallagher, James D., "Management Information Systems and the Computer," *AMA Research Study:* No. 51 (1961).

69. Gallenson, L., "On-Line I/O Processor for the Command Research Laboratory," The PDP-1-C-30, SDC TM-1653 (Dec. 23, 1963).

70. Gallup, G., "The Miracle Ahead," Harper and Row, New York (1964).

71. Gass, S. I., Marilyn B. Scott, R. Hoffman, W. K. Green, A. Peckar, R. D. Peavey and J. E. Hamlin, "Project Mercury Real-Time Computational and Data Flow System," *Proc. Eastern Joint Computer Conference* Vol. 20, 33–78 (Dec. 1961).

72. Ginzberg, M. G., "Notes on Testing Real-Time Systems Programs," *IBM System Journal* 4 (1), 58–72 (1965).

73. Glaser, E. L., "The Structure of On-Line Information Processing Systems," *Proc. Second Congress on Information Sciences, Homestead, Va.,* 1–11 (Nov. 1965).

74. Glaser, E. L. and F. G. Corbato, "Introduction to Time-Sharing," *Datamation* Vol. 10 (11) (Nov. 1964).

75. Glaser, E. L., J. F. Couleur and G. A. Oliver, "System Design of a Computer for Time-Sharing Applications," *Proc. Fall Joint Computer Conference, Las Vegas, Nevada,* Vol. 27 (Nov. 30, 1965).

76. Greenberger, Martin, "The Computers of Tomorrow," *Atlantic Monthly,* 63–67 (May 1964).

77. Greenberger, Martin, "Management and the Computer of the Future," The M.I.T. Press and John Wiley and Sons, Inc., (1962).

78. Gruenbeyer, Fred, "Are Small Free-Standing Computers Here to Stay?," *Datamation* Vol. 12 (4) (April 1966).

79. Harris, R. P., "The PDP-6," *Datamation* Vol. 10 (11) (Nov. 1964).

80. Hastings, Thomas N., "Real-Time Computing with Time-Sharing," *Computers and Automation* Vol. 14 (10) (Oct. 1965).

81. Hittel, L. A., "Some Problems in Data Communications Between the User and the Computer," *AFIPS Conference Proceedings, Fall Joint Computer Conference* Vol. 29 (7–10), 395–402 (1966).

82. Holland, J. H., "On Iterative Circuit Computers Constructed of Micro-Electronic Components and Systems, *Proc. Western Joint Computer Conference,* p. 259 (May 1960).

83. Holt, A. W., "Program Organization and Record Keeping for Dynamic Storage Allocation," *Comm. ACM* Vol. 4, 422–431 (Oct. 1961).

84. Hoover, E. S. and Eckhart, "Performance of a Monitor for a Real-Time Control System," *AFIPS Conference Proceedings, Fall Joint Computer Conference* Vol. 29 (7–10), 23–26 (1966).

85. IBM "1800 Time-Sharing Executive System Specifications," File 1800-36, Form No. C26-5990-0.

86. Iliffe, J. K. and J. G. Jodeit, "A Dynamic Storage Allocation Scheme," *Computer J.* Vol. 5, 200–209 (Oct. 1962).

87. Johnson, T. E., "Sketchpad III: A Computer Program for Drawing in Three Dimensions," *Proc. Spring Joint Computer Conference,* p. 347, Detroit, Michigan (May 1963).

88. "The JOSS System, Time Sharing at Rand," *Datamation* Vol. 10 (11) (Nov. 1964).

89. Kemper, D. A., "Operation of CRL Teletype System," SDC TM 1488/000/00 (Sept. 18, 1963).

90. Kennedy, J. R., "A System for Time-Sharing Graphic Consoles," *AFIPS Conference Proceedings, Fall Joint Computer Conference* Vol. 29 (7–10), 211–222 (1966).

91. Keydata, "Data Processing — On Line . . . In Real Time . . . The Keydata System," Keydata Corporation, 575 Technology Square, Cambridge, Mass.

92. Kilburn, T., R. B. Payne and D. J. Howarth, "The Atlas Supervisor," *Proc. Eastern Joint Computer Conference* Vol. 20, 279–294 (1961).

93. Kilburn, T., D. B. G. Edwards, M. J. Lanigan, and F. H. Sumner, "One Level Storage System," *IRE Transactions on Electronic Computers* Vol. EC-11 (2), 223–235 (April 1962).

94. King, Gilbert W. et al., "Automation and the Library of Congress," Washington, D. C.: Library of Congress (1963).

95. Kinslow, H. A., "The Time-Sharing Monitor System," *Fall Joint Computer Conference,* Vol. 26, Part 1, 443–454 (1964).

96. Kolsky, "Centralization vs. Decentralization." Tenth Annual Symposium on Computers and Data Processing (June 26–27, 1963).

97. Lampson, Butler W., "Time Sharing System Reference Manual," Working Document, University of California. Document No. 30.1030; issued Sept. 30, 1965; revised Dec. 30, 1965.

98. Lampson, B. W., W. W. Lichtenberger, M. W. Pirtle, "A User Machine in a Time-Sharing System," *Proc. IEEE* Vol. 54 (12), 1766–1774 (Dec. 1966).

99. Landis, N., A. Manos, and L. R. Turner, "Initial Experience with An Operating Multiprogramming System," *Comm. ACM,* Vol. 5 (5) (May 1962).

100. Lawless, W. J., "Developments in Computer Logical Organization," *Advances in Electronics and Electron Physics* Vol. 100, Academic Press, Inc., New York (1959).

101. Lehman, M., "A Survey of Problems and Preliminary Results Concerning Parallel Processing and Parallel Processors." *Proc. IEEEE* Vol. 54 (12), 1889–1901 (Dec. 1966).

102. Leiner, A. L., W. A. Notz, J. L. Smith and W. W. Youden, "PILOT Multiple Computer System (Manual)." National Bureau of Standards Report 6688. See also *Journal of ACM* Vol. 6 (3) (July 1959).

103. Lehrer, N. H. and Ketchpel, R. D., "Recent Progress in a High-Resolution, Meshless, Direct-View Storage Tube," *AFIPS Conference Proceedings,* Fall Joint Computer Conference Vol. 29 (7–10), 531–540 (1966).

104. Levine, S. et al, "A Fast Response Data Communications System for Airline Reservations," *Communication and Electronics* (Nov. 1961).

105. Lichtenberger, W. W. and M. W. Pirtle, "A Facility for Experimentation in Man-Machine Interaction," *Proc. Fall Joint Computer Conference, Las Vegas, Nevada* Vol. 27 (Nov. 30, 1965).

106. Licklider, J. C. R., "Man Computer Symbiosis," *IRE Transactions on Human Factors in Electronics* Vol. HFE-1, 4–11 (March 1960).

107. Licklider, J. C. R. and W. E. Clark, "On-Line Man-Computer Communication," *Proc. Spring Joint Computer Conference,* 113–128 (1962).

108. Lonergan, L. and P. King, "Design of the B5000 System," *Datamation* Vol. 7 (5) (May 1961).

109. McCarthy, J., "Time Sharing Computer Systems," *Management and the Computer of the Future* (M. Greenberger, Editor), M.I.T. Press, Cambridge, 221–236 (1962).

110. McCarthy, J., S. Boilen, E. Fredkin, and J. C. R. Licklider, "A Time-Sharing Debugging System for a Small Computer," Proc. *Spring Joint Computer Conference* Vol. 23, 355–363 (1963).

111. McClung, L. W., "A Disc-Oriented IBM 7094 System," Paper #3, Disk File Symposium, March 6–7, 1963, Hollywood, California (Sponsored by Informatics, Inc.)

112. Maher, R. J., "Principles of Storage Allocation in a Multiprocessor Multiprogrammed System," *Comm. of ACM* Vol. 4, 421–422 (Oct. 1961).

113. Malcom, Donald G. and Alan J. Rowe, "Management Control Systems," John Wiley and Sons, Inc.

114. Marcotty, M. J., F. M. Longstaff, and Audrey P. M. Williams, "Time-Sharing on the Ferranti Packard FP6000 Computer System," *Proc. Spring Joint Computer Conference* Vol. 23, 29–40 (1963).

115. Marill, T. and Roberts, L. G., "A Proposed Communications Network to Tie Together Existing Computers." *AFIPS Conference Proceedings, Fall Joint Computer Conference* Vol. 29 (7–10), 425–433 (1966).

116. Mendelson, M. J. and A. W. England, "The SDS SIGMA 7: A Real-Time Time-Sharing Computer," *AFIPS Conference Proceedings, Fall Joint Computer Conference* Vol. 29 (7–10), 51–64 (1966).

117. M.I.T. Digital Computer Lab., "Comprehensive System Manual — A System of Automation Codes for the Whirlwind Corporation," Memo M-2539-2 (Dec. 1953).

118. Nebel, B. E., "A Multiprogrammed Teleprocessing System for Computer Typesetting," *AFIPS Conference Proceedings. Fall Joint Computer Conference* Vol. 29 (7–10), 115–124 (1966).

119. Nelson, T. H., "A File Structure for the Complex, the Changing and the Indeterminate," *ACM National Conference* (Aug. 1965).

120. Nisenoff, N., "Hardware for Information Processing Systems: Today and in the Future," *Proc. IEEE* Vol. 54 (12), 1820–1835 (Dec. 1966).

121. Ochsner, B. P., "Controlling a Multiprocessor System," Bell Telephones Lab. Record (Feb. 1966).

122. Ossanna, J. F., L. E. Mikus, and S. D. Dunten, "Communications and Input/Output Switching in a Multiplex Computing System," *Proc. Fall Joint Computer Conference, Las Vegas,* Nevada Vol. 27 (Nov. 30, 1965).

123. Parkhill, D. F., "The Challenge of the Computer Utility," *Addison-Wesley Publishing Company* LC-66-24245, (1966). Addison-Wesley Publishing Company LC-66-24245, (1966).

124. Penny, J. D. and T. Pearcey, "Use of Multiprogramming in the Design of a Low Cost Digital Computer," *Comm. ACM* Vol. 5 (9), p. 473 (Sept. 1962).

125. Perlis, A. J., "A Disk File Oriented Time Sharing System," Disk File Symposium, March 1963 (sponsored by Informatics, Inc., Culver City, Calif.).

126. Peters, B., "Security Consideration in Multi-Programmed Computer System," *AFIPS Conference Proceedings, Spring Joint Computer Conference* Vol. 30, 283–286 (1967).

127. Proctor, James W., Jr., "The Voice Response System," *Datamation* Vol. 12, 43–44 (Aug. 1966).

128. Ramamoorthy, C. A., "The Analytic Design of a Dynamic Lookahead and Program Segment — System for Multiprogrammed Computers," *Proc. ACM 21st Conference,* 229–239.

129. Ramsay, Karl and J. C. Strauss, "A Real Time Priority Scheduler," *Proc. ACM 21st National Conference,* 161–166.

130. Reiter, A., "A Resource Allocation Scheme for Multi-User On-Line Operation of a Small Computer," *AFIPS Conference Proceedings, Spring* Joint Computer Conference Vol. 30, 1–8 (1967).

131. Roberts, L. G., "The Lincoln Wand," AFIPS Conference Proceedings, *Fall Joint Computer Conference* Vol. 29 (7–10), 223–228 (1966).

132. Rosenberg, A. M. (Editor), "Command Research Laboratories Users Guide," SDC TM-1354 (Nov. 19, 1965).

133. Ross, D. T. and J. E. Rodriguez, "Theoretical Foundations for the Computer-Aided Design System," *Computer Aided Design, Spring Joint Computer Conference,* p. 305 (1963).

134. Samuel, A. L., "Time Sharing on a Computer," *New Scientist* Vol. 26, 583–587 (May 27, 1965).

135. Scherr, Alan L., "Time Sharing Measurement," *Datamation* Vol. 12 (4) (April 1966).

136. Schwartz, E. S., "Automatic Sequencing Procedure with Application to Parallel Programming," *Journal of ACM* Vol. 8, 513–537 (Oct. 1961).

137. Schwartz, J. I., E. G. Coffman, and C. Weissman, "A General Purpose Time-Sharing Systems," *Spring Joint Computer Conference* Vol. 25, 397–411 (1964).

138. Schwartz, J. I., "Observations on Time-Shared Systems," *ACM Proceedings of the 20th National Conference,* p. 525 (1965).

139. Schwartz, Jules I., "The SDC Time-Sharing System Part 1," *Datamation* Vol. 10 (1), Part 2, (Nov. 1964); *Datamation* Vol. 10 (12) (Dec. 1964).

140. Scott, M. B. and R. Hoffman, "The Mercury Programming System," *Proc. Eastern Joint Computer Conference,* Vol. 20, 47–53 (Dec. 1961).

141. Sprague, Richard E., "On Line-Real Time Systems — 1964," *Management Services* (May-June 1964).

142. Stanga, D. C., "Univac 1108 Multiprocessor System," *AFIPS Conference Proceedings, Spring Joint Computer Conference* Vol. 30, 67–74 (1967).

143. Stotz, R., "Man-Machine Console Facilities for Computer-Aided Design," *Computer Aided Design, Spring Joint Computer Conference,* p. 323 (1963).

144. Strachey, C., "Time Sharing in Large Fast Computers," *Proc. of the International Conference on Information Processing, Paris, UNESCO,* 336–341 (1960).

145. Summer, F. H. and E. C. Y. Chen, "The Central Control Unit of the ATLAS Computer," *Proc. of IFIP Congress,* p. 657 (1962).

146. Sutherland, I. E., "Sketchpad: A Man-Machine Graphical Communication System," Lincoln Lab Technical Report No. 296, M.I.T., January 30, 1963, *Computer Aided Design, Spring Joint Computer Conference,* 329–346 (1963).

147. Teleregister, "200 Display System," The Bunker-Ramo Corporation, Stamford, Conn.

148. Teleregister, "On-Line Data Processing for Hotels," The Bunker-Ramo Corporation, Stamford, Conn.

149. Vyssotsky, V. A., F. J. Corbato, and R. M. Graham, "Structure of the Multics Supervisor," *Proc. Fall Joint Computer Conference,* Las Vegas, Nevada, Vol. 27 (Nov. 30, 1965).

150. Weil, J. W., "A Heuristic for Page Turning in a Multiprogrammed Computer," *Comm. ACM* Vol. 5 (9), p. 480 (Sept. 1962).

151. Weil, J. W., "The Impact of Time-Sharing on Data Processing Management," DPMA Quarterly 2, 2, 2–16 (Jan. 1966).

152. Wilkes, M., "A Programmer's Utility Filing System," *Computer Journal* 7, 180–184 (Oct. 1964).

153. Yates, John E., "A Time-Sharing System for the PDP-1 Computer," M.I.T. Press (1962).