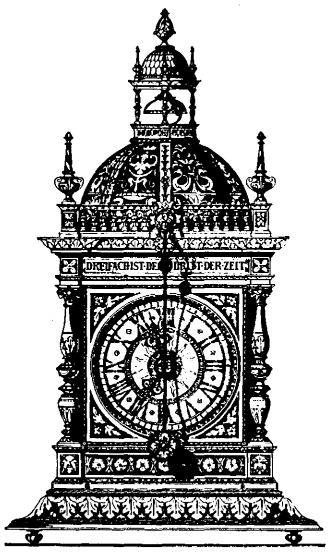
Fundamentals of Time Shared Computers

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.

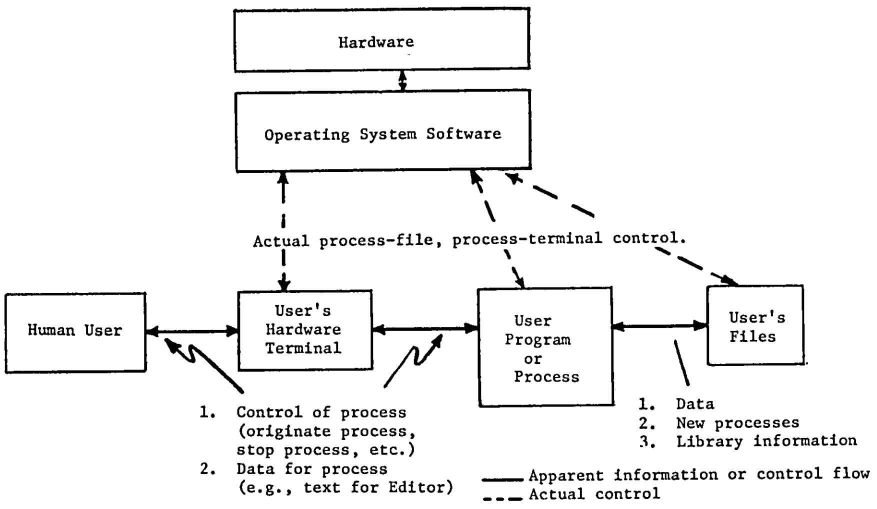
Time sharing provides a level of service to a user who could only previously have had the service by owning his own computer. The sharing is based on the principle that there is enough capacity in a computer for multiple users, assuming: the proper ordering of requests; the user consoles are active only 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.



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 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.

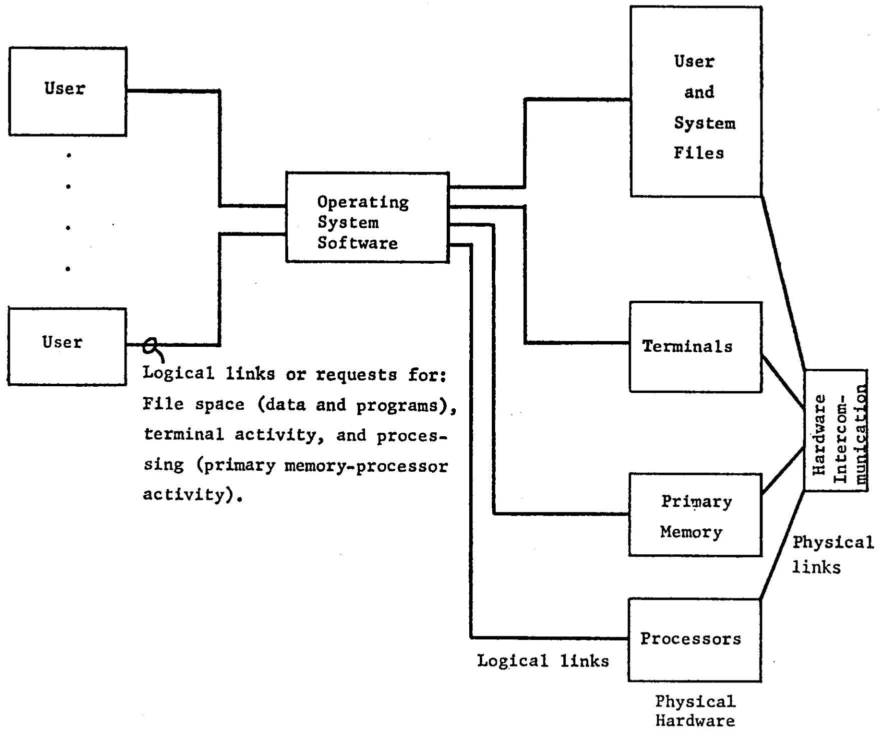
The User's Apparent System

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.



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

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

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.

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.

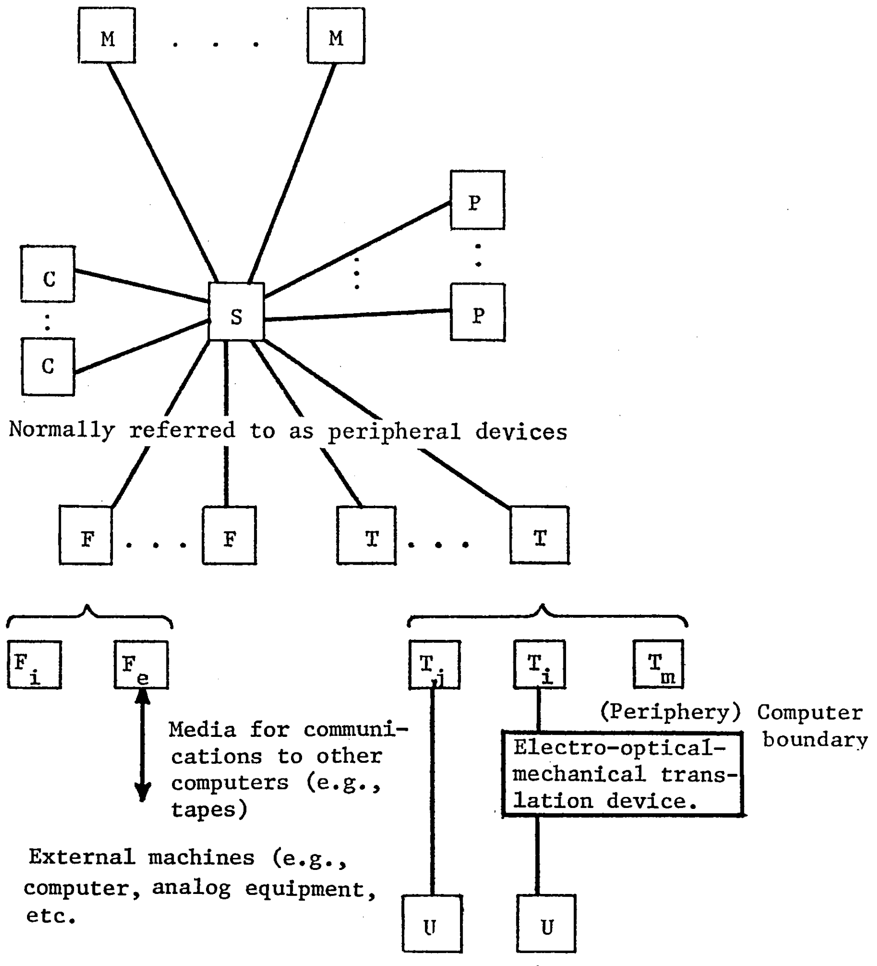
SPECIAL PURPOSE AND GENERAL PURPOSE TIME-SHARING

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.

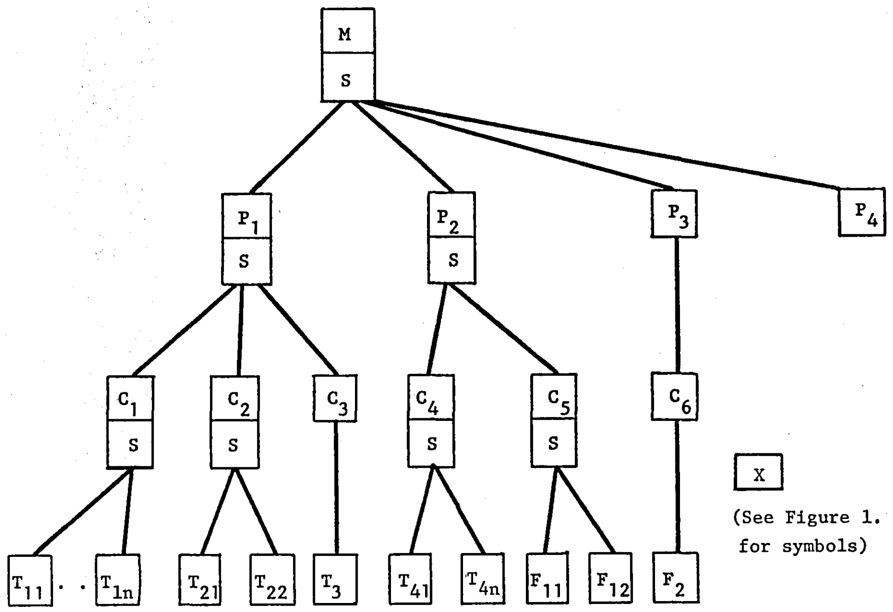


Fig. 4. Structure of a simplex computer system.

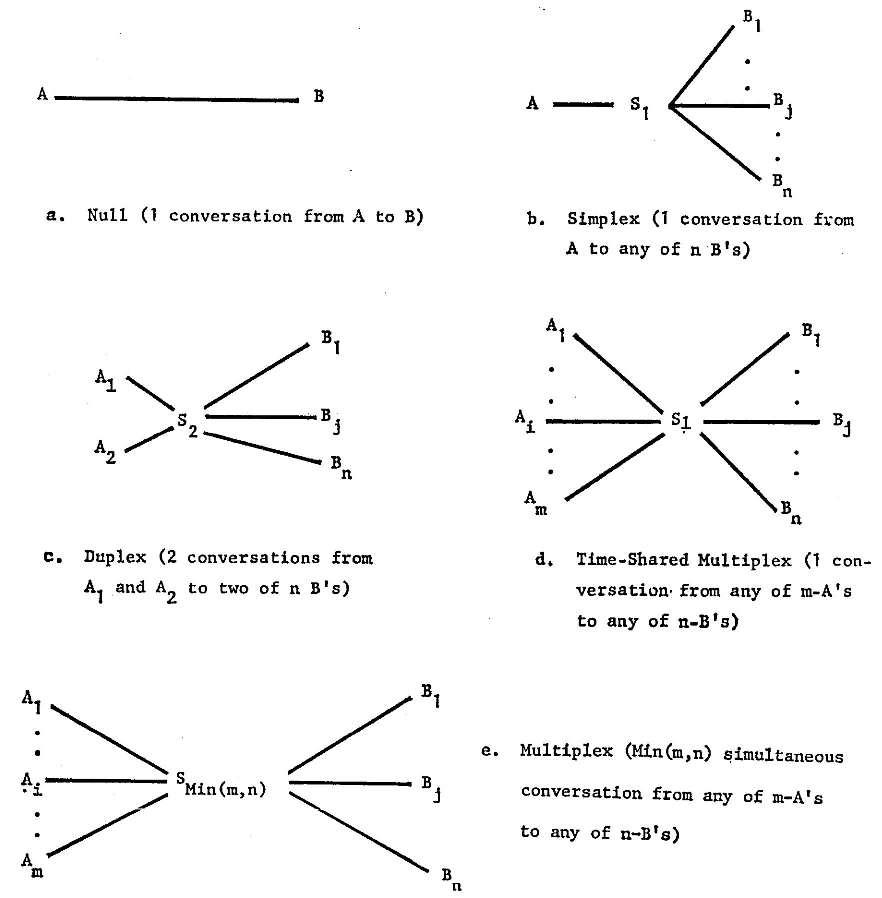

A
B
a. Null (1 conversation from A to B)
A
S1
B1
Bj
Bn
b. Simplex (1 conversation from A to any of n B's)
A1
A2
S2
B1
Bj
Bn
c. Duplex (2 conversations from A1 and A2 to two of n B's)
A1
Ai
Am
S1
B1
Bj
Bn
d. Time-Shared Multiplex (1 conversation from any of m-A's to any of n-B's)
A1
Ai
Am
SMin(m,n)
B1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

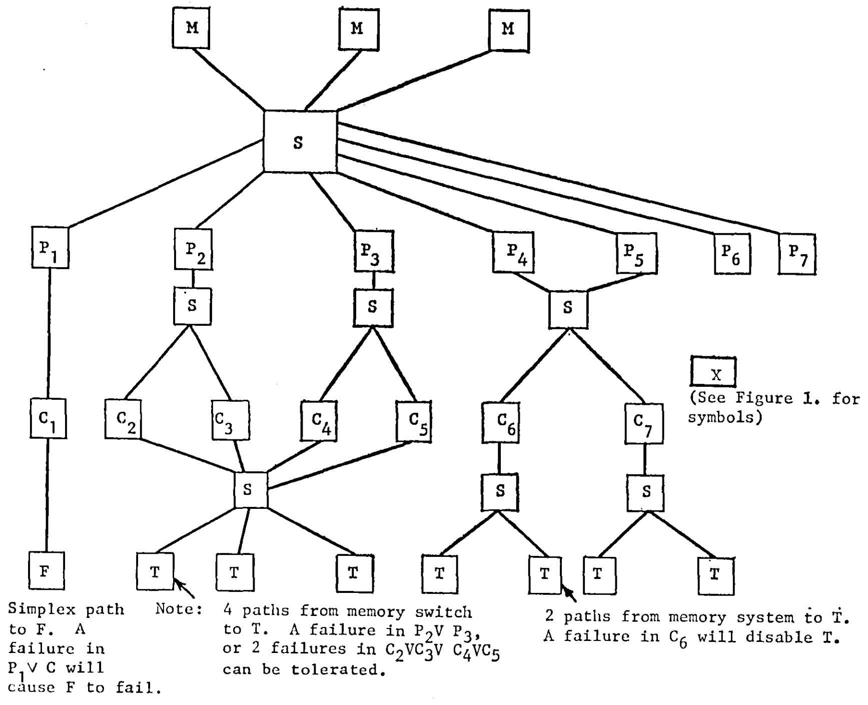

M
M
M
S
P_1
P_2
P_3
P_4
P_5
P_6
P_7
S
S
S
X
(See Figure 1. for symbols)
C_1
C_2
C_3
C_4
C_5
C_6
C_7
S
S
S
F
T
T
T
T
T
T
T
Simplex path to F. A failure in $P_1 \vee C$ will cause F to fail.
Note: 4 paths from memory switch to T. A failure in $P_2 \vee P_3$, or 2 failures in $C_2 \vee C_3 \vee C_4 \vee C_5$ 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 processed.
• 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 Multiplication, 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 another allows a system to be built without 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 addressing, 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 1/4 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.

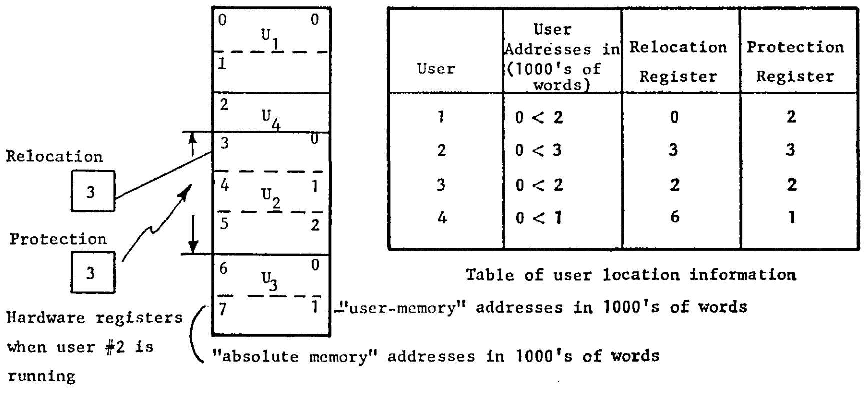
In general, the goal is to effectively provide each user or user's program with a large, continuous memory space as though he were the sole user. A further goal is to provide a method such that any two identical blocks in primary memory would not have to be duplicated. This ability has significance in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


U1
U4
U2
U3
Relocation
3
Protection
3
Hardware registers when user #2 is running
"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

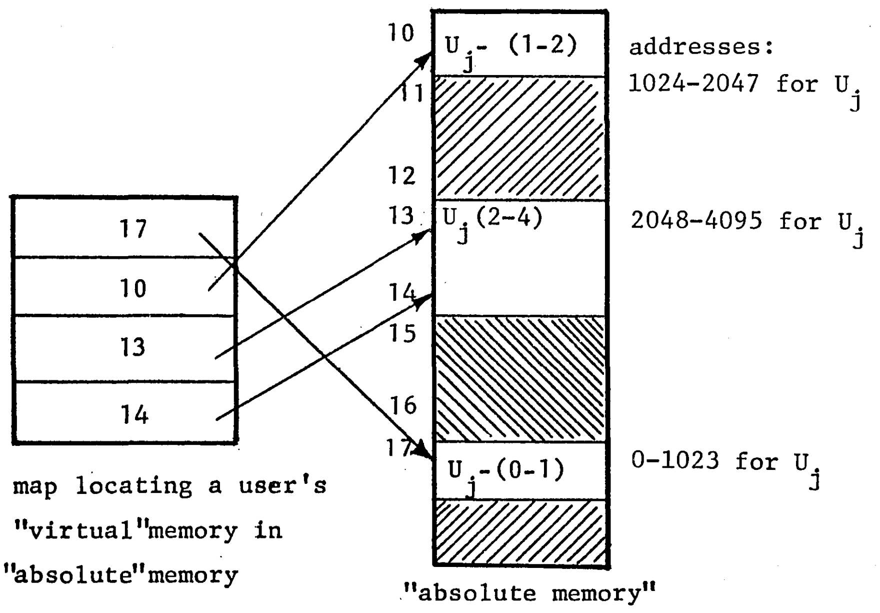
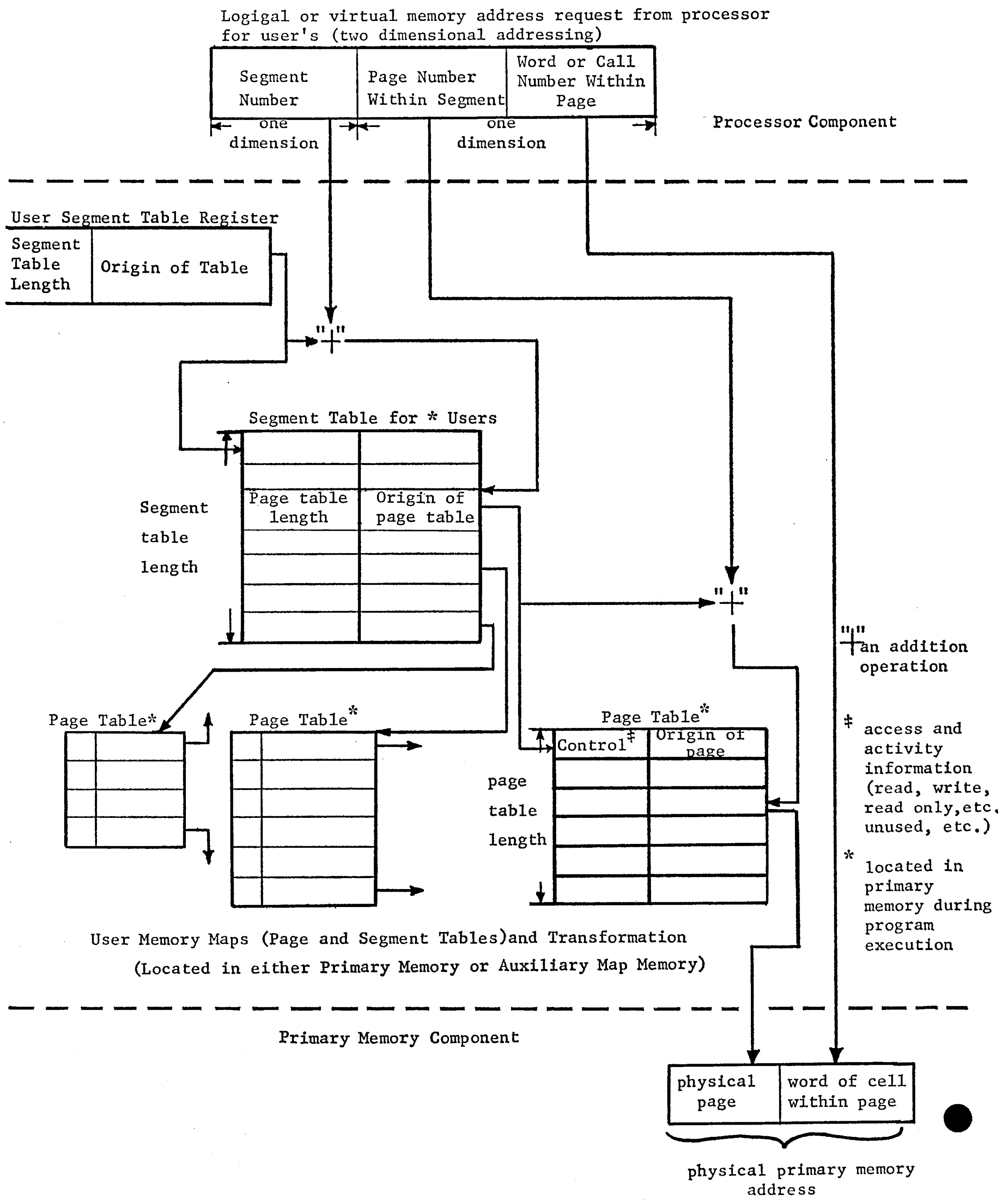


Fig. 8. Memory allocation using page allocation map.



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
‡ access and activity information (read, write, read only,etc, unused, etc.)
* located in primary memory during program execution
Page Table*
Page Table*
Page Table*
Control‡
Origin of page
page table length
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 a 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

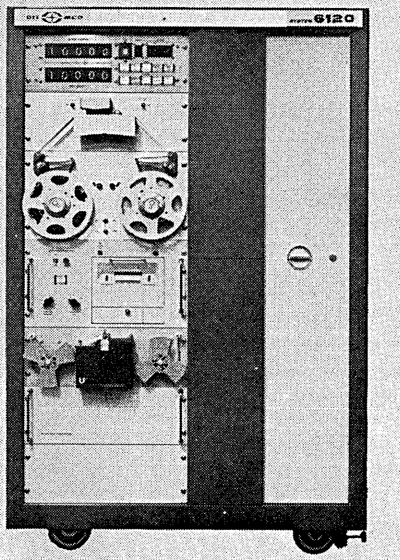
In general, a community is provided with a much larger system than any single member could afford. For on line or real time systems, the hardware and software overhead associated with this additional ability can be associated with a larger number of users.

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.



FASTEST WIRING SYSTEM ANALYZER IN THE WEST AND EAST AND NORTH AND SOUTH.

The DIT-MCO System 6120 walks tall in the world of wiring system analyzers. It's a tough, versatile and highly adaptable testing unit that's ready, willing and able to meet today's demand for speed, accuracy and flexibility. Works on the latest fully automatic taped program and printout concept.

INSULATION TEST CAPACITY:
2010 PER MINUTE!

CONTINUITY TEST CAPACITY:
2963 PER MINUTE!

The DIT-MCO System 6120 has been thoroughly lab and field tested. Switching Console contains terminal selector and high speed 500-termination Reed relay switching modules that give you a total system capacity of up to 50,000 terminations. Save time, save manpower, save dollars — and — improve both the testing function and the tested product with the DIT-MCO System 6120.

Write for Detailed Specs and Full Information.

DIT MCO

DIT-MCO INTERNATIONAL
A DIVISION OF XEBEC CORPORATION

5612 BRIGHTON TERRACE
KANSAS CITY, MISSOURI 64130
TELEPHONE (816) 363-6288
TELEX NUMBER 42-6149

IT'S NO WONDER APPROXIMATELY 90% OF ALL MAJOR MANUFACTURERS IN THE COMPUTER AND AEROSPACE INDUSTRIES ARE SATISFIED USERS OF DIT-MCO SYSTEMS.

CIRCLE NO. 39 ON INQUIRY CARD