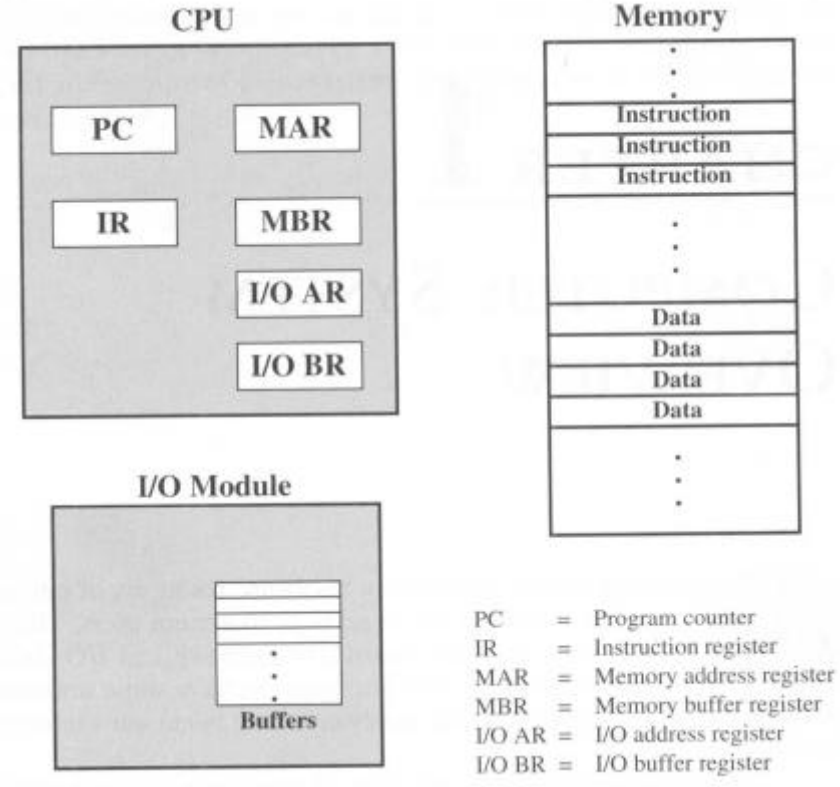
Basic Elements of Hardware

At a top level, a computer consists of processor, memory, and 1/0 components, with one or more modules of each type. These components are interconnected in some fashion to ichieve the main function of the computer, which is to **execute programs**. Thus, there are four main structural elements:

- **Processor:** Controls the operation of the computer and performs its data processing functions. When there is only one processor, it is often referred to as the central processing unit (CPU).
- **Main memory:** Stores data and programs. This memory is typically volatile: it is also referred to as real memory or primary memory.
- 1/0 **modules:** Move data between the computer and its external environmcnt. The external environment consists of a variety of external devices, including secondary memory devices, communications equipment, and terminals.
- **System interconnection:** Some structures and mechanisms that provide for communication among processors, main memory, and 1/0 modules.



The Memory Hierarchy

The design constraints on a computer's memory can be summed up by three questions: How much? How fast? How expensive?

The question of how much is somewhat open ended. If the capacity is there, applications will likely be developed to use it. The question of how fast is, in a sense, easier to answer. To achieve the greatest performance, the memory must be able to keep up with the processor. That is, as the processor is executing instructions, we would not want it to have to pause for instructions or operands. The final question must also be considered. For a practical system the cost of memory must be reasonable in relationship to other components.

As might be expected, there is a tradeoff among the three key characteristics of memory: namely, cost, capacity, and access time. At any given time, a variety of technologies are used to implement memory systems. Across this spectrum of technologies, the following relationships hold:

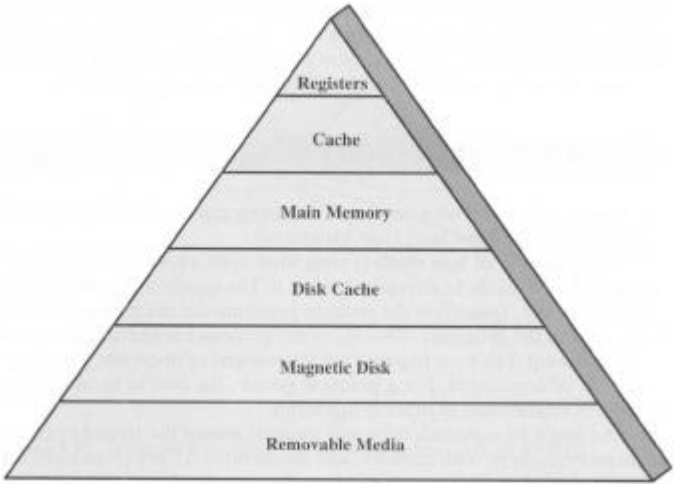
- Smaller access time, greater cost per bit
- Greater capacity, smaller cost per bit
- Greater capacity, greater access time.

The dilemma facing the designer is clear. The designer would like to use memory technologies that provide for large-capacity memory, both because the capacity is needed and because the cost per bit is low. However, to meet performance requirements, the designer needs to use expensive, relatively lower-capacity memories with fast access times.

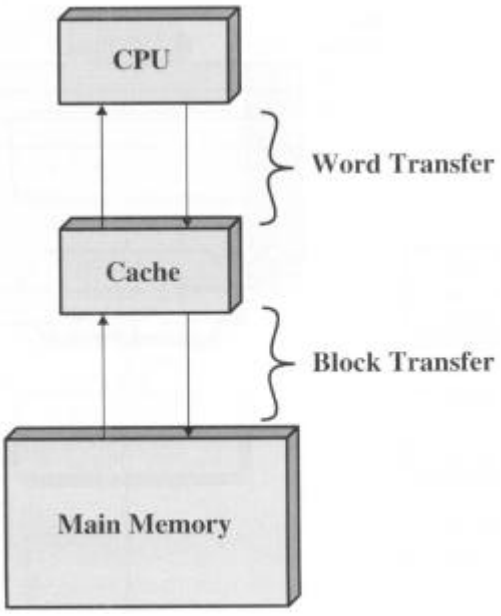
The way out of this dilemma is not to rely on a single memory component or technology, but to employ a **memory hierarchy.** A typical hierarchy is illustrated below. As one goes down the hierarchy, the following occur:

a. Decreasing cost per bit
b. Increasing capacity
c. Increasing access time
d. Decreasing frequency of access of the memory by the processor.

Thus, smaller, more expensive, faster memories are supplemented by larger, cheaper, slower memories. The key to the success of this organization is the last item: decreasing frequency of access.



Cache Memory



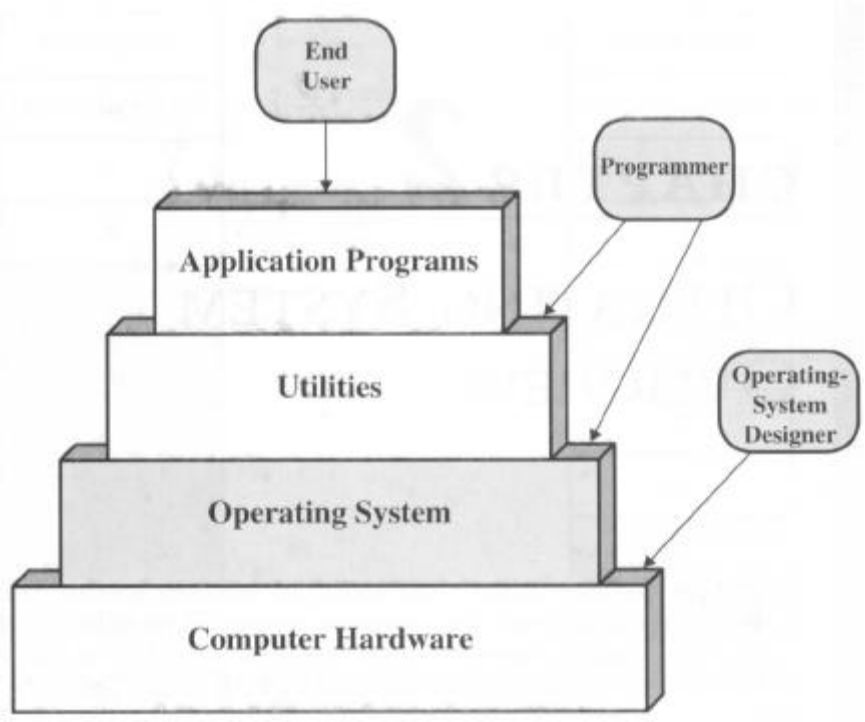
Operating System Objectives and **Functions**

An operating system is a program that controls the execution of application programs and acts as an interface between the user of a computer and the computer hardware. It can be thought of as having three objectives:

Convenience: An operating system makes a computer more convenient to use.

Efficiency: An operating system allows the computer system resources to be used in an efficient manner.

Ability to evolve: An operating system should be constructed in such a way as to permit the effective development, testing, and introduction of new system functions without interfering with service.

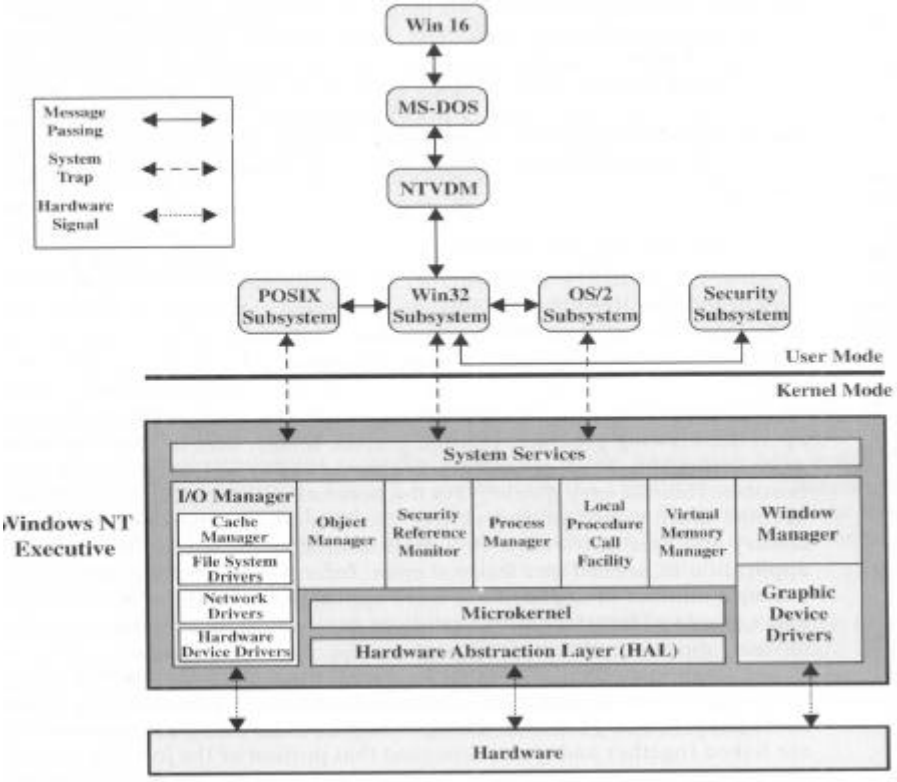


The Operating System as a User/Computer Interface

The hardware and software used in providing applications to a user can be viewed in a layered or hierarchical fashion, as depicted in figure. The user of those applications, the end user, generally is not concerned with the computer's architecture.

Thus the end user views a computer system in terms of an application. That application can be expressed in a programming language and is developed by an application programmer. If one were to develop an application program as a set of machine instructions that is completely responsible for controlling the computer hardware, one would be faced with an overwhelmingly complex task.

To ease this task, a set of system programs is provided. Some of these programs are referred to as utilities. These implement frequently used functions that assist in program creation, the management of files, and the control of I/O devices. A programmer will make use of these facilities in developing an application, and the application, while it is running, will invoke the utilities to perform certain functions.



The most important system program is the operating system. The operating system masks the details of the hardware from the programmer and provides the programmer with a convenient interface for using the system. It acts as mediator, making it easier for the programmer and for application programs to access and use those facilities and services.

Briefly, the operating system typically provides services in the following areas:

Program creation: The operating system provides a variety of facilities and services, such as editors and debuggers, to assist the programmer in creating programs. Typically, these services are in the form of utility programs that are not actually part of the operating system but are accessible through the operating system.

Program execution: A number of tasks need to be performed to execute a program. Instructions and data must be loaded into main memory, 1/0 devices and files must be initialized, and other resources must be prepared. The operating system handles all of this for the user.

Access to 1/0 devices: Each 1/0 device requires its own peculiar set of instructions or control signals for operation. The operating system takes care of the details so that the programmer can think in terms of simple reads and writes.

Controlled access to files: In the case of files, control must include an understanding of not only the nature of the 1/0 device (disk drive, tape drive) but also the file format on the storage medium. Again, the operating system worries about the details. Further, in the case of a system with multiple simultaneous users, the operating system can provide protection mechanisms to control access to the files.

System access: In the case of a shared or public system, the operating system controls access to the system as a whole and to specific system resources. The access function must provide protection of resources and data from unauthorized users and must resolve conflicts for resource contention.

Error detection and response: A variety of errors can occur while a computer system is running. These include internal and external hardware errors, such as a memory error or a device failure or malfunction; and various software errors, such as arithmetic overflow, attempt to access forbidden memory location, and inability of the operating system to grant the request of an application. In each case, the operating system must make the response that clears the error condition with the least impact on running applications. The response may range from ending the program that caused the error, to retrying the operation, to simply reporting the error to the application.

Accounting: A good operating system will collect usage statistics for various resources and monitor performance parameters such as response time. On any system, this information is useful in anticipating the need for future enhancements and in tuning the system to improve performance. On a multiuser system, the information can be used for billing purposes.

The Operating System as Resource Manager

A computer is a set of resources for the movement, storage, and processing of data and for the control of these functions. The operating system is responsible for managing these resources.

Can we say that it is the operating system that controls the movement, storage, and processing of data? From one point ot'view, the answer is yes: By managing the computer's resources, the operating system is in control of the computer's basic functions. But this control is exercised in a curious way. Normally, we think of a control mechanism as something external to that which is controlled, or at least as something that is a distinct and separate part of that which is controlled. (For example, a residential heating system is controlled by a thermostat, which is completely distinct from the heat-generation and heat-distribution apparatus.) This is not the case with the

operating system, which as a control mechanism is unusual in two respects:

The operating system functions in the same way as ordinary computer software; that is, it is a program executed by the processor.
The operating system frequently relinquishes control and must depend on the processor to allow it to regain control.

The operating system is, in fact, nothing more than a computer program. Like other computer programs, it provides instructions for the processor. The key difference is in the intent of the program. The operating system directs the processor in the use of the other system resources and in the timing of its execution of other programs. But in order for the processor to do any of these things, it must cease executing the operating system program and execute other programs. Thus, the operating system relinquishes control for the processor to do some "useful" work and then resumes control long enough to prepare the processor to do the next piece of work. The mechanisms involved in all this should become clear as the chapter proceeds.

Figure suggests the main resources that are managed by the operating system. A portion of the operating system is in main memory. This includes the **kernel,** or **nucleus,** which contains the most-frequently-used functions in the operating system and, at a given time, other portions of the operating system currently in use. The remainder of main memory contains other user programs and data. The allocation of this resource (main memory) is controlled jointly by the operating system and memory-management hardware in the processor, as we shall see. The operating system decides when an 1/0 device can be used by a program in execution and controls access to and use of files. The processor itself is a resource, and the operating system must determine how much processor time is to be devoted to the execution of a particular user program. In the case of a multiple-processor system, this decision must span all of the processors.

Ease of Evolution of an Operating System

A major operating system will evolve over time for a number of reasons:

* **Hardware upgrades plus new types of hardware:** For example, early versions of UNIX and OS/2 did not employ a paging mechanism because they were run on machines without paging hardware. More recent versions have been modified to exploit paging capabilities. Also, the use of graphics terminals and page-mode terminals instead of line-at-a-time scroll mode terminals may affect operating system design. For example, such a terminal may allow the user to view several applications at the same time through "windows" on the screen. This requires more sophisticated support in the operating system.
* **New services:** In response to user demand or in response to the needs of system managers, the operating system will expand to offer new services. For example, if it is found to be difficult to maintain good performance for users with existing tools, new measurement and control tools may be added to the operating system. Another example is new applications that require the use of windows on the display screen. This feature will require major upgrades to the operating system.
* **Fixes:** Any operating system has faults. These are discovered over the course of time and fixes are made. Of course, the fix may introduce new faults.

The need to change an operating system on a regular basis places certain requirements on its design. An obvious statement is that the system should be modular in construction, with clearly defined interfaces between the modules, and that it should be well documented. For large programs, such as the typical contemporary operating system, what might be referred to as straightforward modularization is inadequate. That is, much more must be done than simply partitioning a program into subroutines.