This portion consists of the page table entries that correspond to the most recently accessed pages. A small cache, usually called the *Translation Lookaside Buffer* (TLB) is incorporated into the MMU for this purpose. The operation of the TLB with respect to the page table in the main memory is essentially the same as the operation we have discussed in conjunction with the cache memory. In addition to the information that constitutes a page table entry, the TLB must also include the virtual address of the entry. Figure 5.28 shows a possible organization of a TLB where the associative-mapping technique is used. Set-associative mapped TLBs are also found in commercial products.

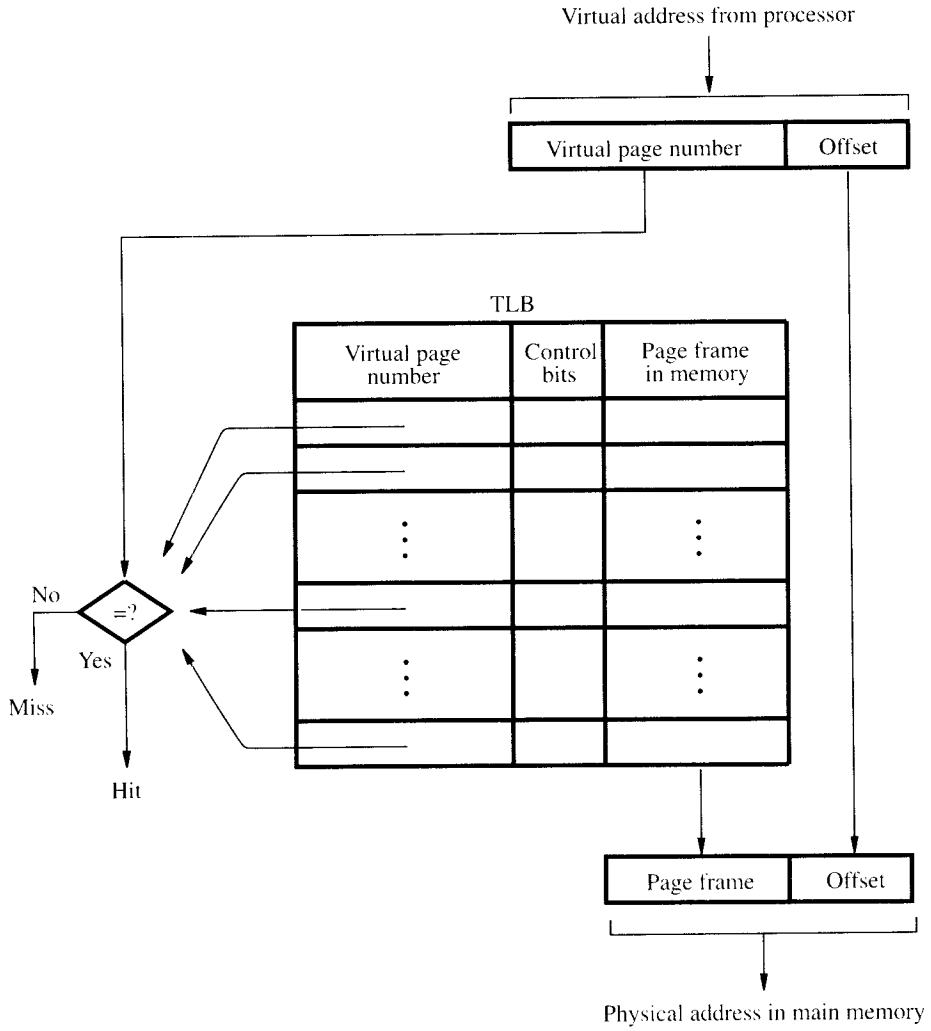


Figure 5.28 Use of an associative-mapped TLB.

An essential requirement is that the contents of the TLB be coherent with the contents of page tables in the memory. When the operating system changes the contents of page tables, it must simultaneously invalidate the corresponding entries in the TLB. One of the control bits in the TLB is provided for this purpose. When an entry is invalidated, the TLB will acquire the new information as part of the MMU's normal response to access misses.

Address translation proceeds as follows. Given a virtual address, the MMU looks in the TLB for the referenced page. If the page table entry for this page is found in the TLB, the physical address is obtained immediately. If there is a miss in the TLB, then the required entry is obtained from the page table in the main memory and the TLB is updated.

When a program generates an access request to a page that is not in the main memory, a *page fault* is said to have occurred. The whole page must be brought from the disk into the memory before access can proceed. When it detects a page fault, the MMU asks the operating system to intervene by raising an exception (interrupt). Processing of the active task is interrupted, and control is transferred to the operating system. The operating system then copies the requested page from the disk into the main memory and returns control to the interrupted task. Because a long delay occurs while the page transfer takes place, the operating system may suspend execution of the task that caused the page fault and begin execution of another task whose pages are in the main memory.

It is essential to ensure that the interrupted task can continue correctly when it resumes execution. A page fault occurs when some instruction accesses a memory operand that is not in the main memory, resulting in an interruption before the execution of this instruction is completed. Hence, when the task resumes, either the execution of the interrupted instruction must continue from the point of interruption, or the instruction must be restarted. The design of a particular processor dictates which of these options should be used.

If a new page is brought from the disk when the main memory is full, it must replace one of the resident pages. The problem of choosing which page to remove is just as critical here as it is in a cache, and the idea that programs spend most of their time in a few localized areas also applies. Because main memories are considerably larger than cache memories, it should be possible to keep relatively larger portions of a program in the main memory. This will reduce the frequency of transfers to and from the disk. Concepts similar to the LRU replacement algorithm can be applied to page replacement, and the control bits in the page table entries can indicate usage. One simple scheme is based on a control bit that is set to 1 whenever the corresponding page is referenced (accessed). The operating system occasionally clears this bit in all page table entries, thus providing a simple way of determining which pages have not been used recently.

A modified page has to be written back to the disk before it is removed from the main memory. It is important to note that the write-through protocol, which is useful in the framework of cache memories, is not suitable for virtual memory. The access time of the disk is so long that it does not make sense to access it frequently to write small amounts of data.

The address translation process in the MMU requires some time to perform, mostly dependent on the time needed to look up entries in the TLB. Because of locality of reference, it is likely that many successive translations involve addresses on the same page. This is particularly evident in fetching instructions. Thus, we can reduce the average translation time by including one or more special registers that retain the virtual page number and the physical page frame of the most recently performed translations. The information in these registers can be accessed more quickly than the TLB.

5.8 MEMORY MANAGEMENT REQUIREMENTS

In our discussion of virtual-memory concepts, we have tacitly assumed that only one large program is being executed. If all of the program does not fit into the available physical memory, parts of it (pages) are moved from the disk into the main memory when they are to be executed. Although we have alluded to software routines that are needed to manage this movement of program segments, we have not been specific about the details.

Management routines are part of the operating system of the computer. It is convenient to assemble the operating system routines into a virtual address space, called the *system space*, that is separate from the virtual space in which user application programs reside. The latter space is called the *user space*. In fact, there may be a number of user spaces, one for each user. This is arranged by providing a separate page table for each user program. The MMU uses a page table base register to determine the address of the table to be used in the translation process. Hence, by changing the contents of this register, the operating system can switch from one space to another. The physical main memory is thus shared by the active pages of the system space and several user spaces. However, only the pages that belong to one of these spaces are accessible at any given time.

In any computer system in which independent user programs coexist in the main memory, the notion of *protection* must be addressed. No program should be allowed to destroy either the data or instructions of other programs in the memory. Such protection can be provided in several ways. Let us first consider the most basic form of protection. Recall that in the simplest case, the processor has two states, the *supervisor state* and the *user state*. As the names suggest, the processor is usually placed in the supervisor state when operating system routines are being executed and in the user state to execute user programs. In the user state, some machine instructions cannot be executed. These *privileged instructions*, which include such operations as modifying the page table base register, can only be executed while the processor is in the supervisor state. Hence, a user program is prevented from accessing the page tables of other user spaces or of the system space.

It is sometimes desirable for one application program to have access to certain pages belonging to another program. The operating system can arrange this by causing these pages to appear in both spaces. The shared pages will therefore have entries in two different page tables. The control bits in each table entry can be set to control the access privileges granted to each program. For example, one program may be allowed to read and write a given page, while the other program may be given only read access.

5.9 SECONDARY STORAGE

Semiconductor memories discussed in the previous sections cannot be used to provide all of the storage capability needed in computers. Their main limitation is the cost per bit of stored information. Large storage requirements of most computer systems are economically realized in the form of magnetic disks, optical disks, and magnetic tapes, which are usually referred to as secondary storage devices.

5.9.1 MAGNETIC HARD DISKS

As the name implies, the storage medium in a magnetic-disk system consists of one or more disks mounted on a common spindle. A thin magnetic film is deposited on each disk, usually on both sides. The disks are placed in a rotary drive so that the magnetized surfaces move in close proximity to read/write heads, as shown in Figure 5.29*a*. The disks rotate at a uniform speed. Each head consists of a magnetic yoke and a magnetizing coil, as indicated in Figure 5.29*b*.

Digital information can be stored on the magnetic film by applying current pulses of suitable polarity to the magnetizing coil. This causes the magnetization of the film in the area immediately underneath the head to switch to a direction parallel to the applied field. The same head can be used for reading the stored information. In this case, changes in the magnetic field in the vicinity of the head caused by the movement of the film relative to the yoke induce a voltage in the coil, which now serves as a sense coil. The polarity of this voltage is monitored by the control circuitry to determine the state of magnetization of the film. Only changes in the magnetic field under the head can be sensed during the Read operation. Therefore, if the binary states 0 and 1 are represented by two opposite states of magnetization, a voltage is induced in the head only at 0-to-1 and at 1-to-0 transitions in the bit stream. A long string of 0s or 1s causes an induced voltage only at the beginning and end of the string. To determine the number of consecutive 0s or 1s stored, a clock must provide information for synchronization. In some early designs, a clock was stored on a separate track, where a change in magnetization is forced for each bit period. Using the clock signal as a reference, the data stored on other tracks can be read correctly.

The modern approach is to combine the clocking information with the data. Several different techniques have been developed for such encoding. One simple scheme, depicted in Figure 5.29*c*, is known as *phase encoding* or *Manchester encoding*. In this scheme, changes in magnetization occur for each data bit, as shown in the figure. Note that a change in magnetization is guaranteed at the midpoint of each bit period, thus providing the clocking information. The drawback of Manchester encoding is its poor bit-storage density. The space required to represent each bit must be large enough to accommodate two changes in magnetization. We use the Manchester encoding example to illustrate how a *self-clocking* scheme may be implemented, because it is easy to understand. Other, more compact codes have been developed. They are much more efficient and provide better storage density. They also require more complex control circuitry. The discussion of such codes is beyond the scope of this book.

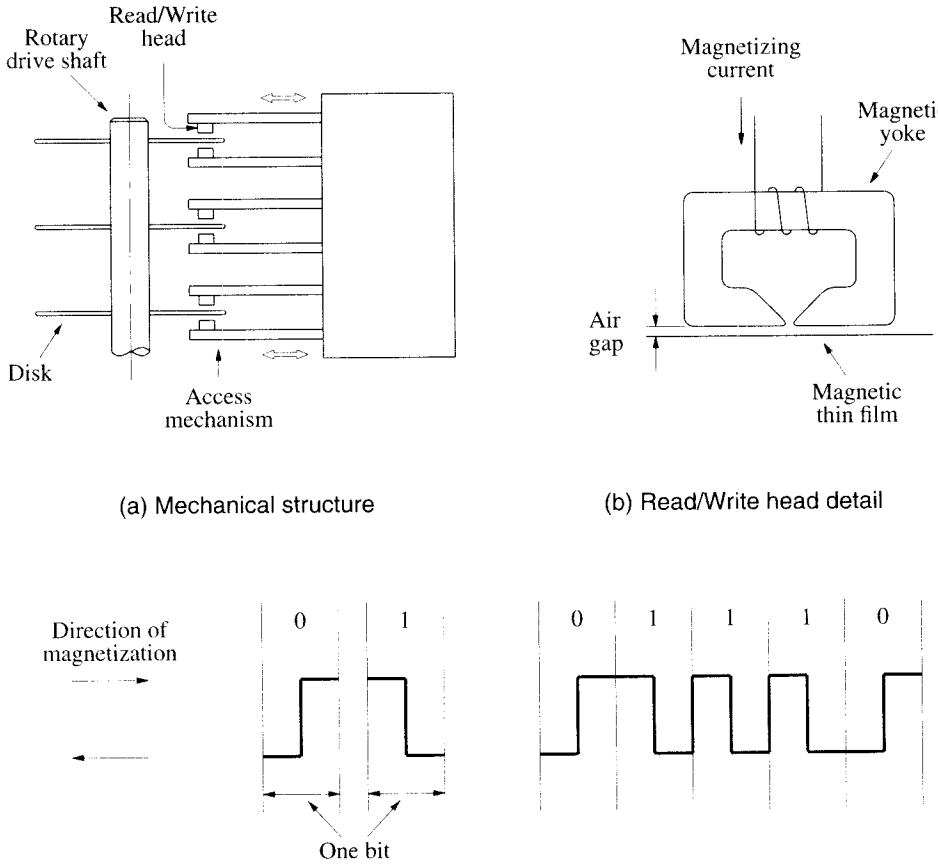


Figure 5.29 Magnetic disk principles.

Read/write heads must be maintained at a very small distance from the moving disk surfaces in order to achieve high bit densities and reliable read/write operations. When the disks are moving at their steady rate, air pressure develops between the disk surface and the head and forces the head away from the surface. This force can be counteracted by a spring-loaded mounting arrangement for the head that allows it to be pressed toward the surface. The flexible spring connection between the head and its arm mounting permits the head to fly at the desired distance away from the surface in spite of any small variations in the flatness of the surface.

In most modern disk units, the disks and the read/write heads are placed in a sealed, air-filtered enclosure. This approach is known as *Winchester technology*. In such units, the read/write heads can operate closer to the magnetized track surfaces because dust particles, which are a problem in unsealed assemblies, are absent. The closer the heads

are to a track surface, the more densely the data can be packed along the track, and the closer the tracks can be to each other. Thus, Winchester disks have a larger capacity for a given physical size compared to unsealed units. Another advantage of Winchester technology is that data integrity tends to be greater in sealed units where the storage medium is not exposed to contaminating elements.

The read/write heads of a disk system are movable. There is one head per surface. All heads are mounted on a comb-like arm that can move radially across the stack of disks to provide access to individual tracks, as shown in Figure 5.29*a*. To read or write data on a given track, the arm holding the read/write heads must first be positioned to that track.

The disk system consists of three key parts. One part is the assembly of disk platters, which is usually referred to as the *disk*. The second part comprises the electromechanical mechanism that spins the disk and moves the read/write heads; it is called the *disk drive*. The third part is the electronic circuitry that controls the operation of the system, which is called the *disk controller*. The disk controller may be implemented as a separate module, or it may be incorporated into the enclosure that contains the entire disk system. We should note that the term disk is often used to refer to the combined package of the disk drive and the disk it contains. We will do so in the sections that follow when there is no ambiguity in the meaning of the term.

Organization and Accessing of Data on a Disk

The organization of data on a disk is illustrated in Figure 5.30. Each surface is divided into concentric *tracks*, and each track is divided into *sectors*. The set of corresponding tracks on all surfaces of a stack of disks forms a logical *cylinder*. The data on all tracks of a cylinder can be accessed without moving the read/write heads. The data are accessed by specifying the surface number, the track number, and the sector number. The Read and Write operations start at sector boundaries.

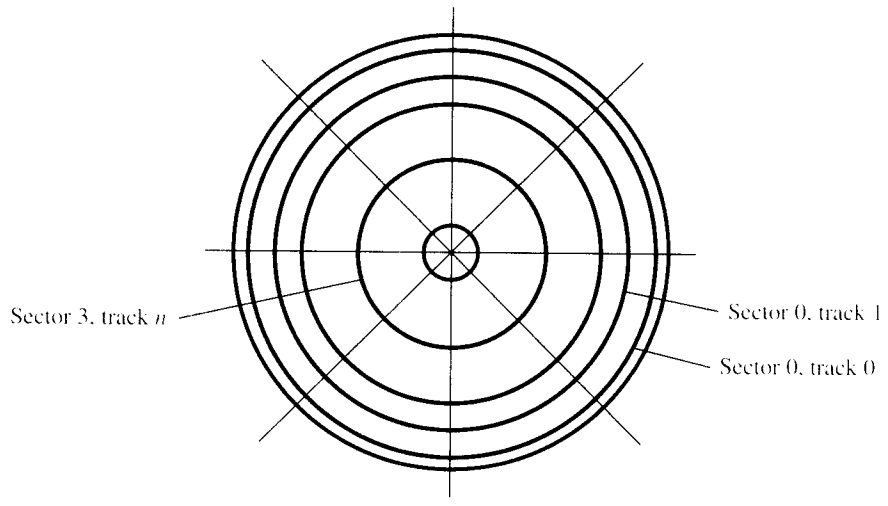


Figure 5.30 Organization of one surface of a disk.

Data bits are stored serially on each track. Each sector usually contains 512 bytes of data, but other sizes may be used. The data are preceded by a *sector header* that contains identification (addressing) information used to find the desired sector on the selected track. Following the data, there are additional bits that constitute an *error-correcting code* (ECC). The ECC bits are used to detect and correct errors that may have occurred in writing or reading of the 512 data bytes. To easily distinguish between two consecutive sectors, there is a small *intersector gap*.

An unformatted disk has no information on its tracks. The formatting process divides the disk physically into tracks and sectors. This process may discover some defective sectors or even whole tracks. The disk controller keeps a record of such defects and excludes them from use. The capacity of a formatted disk is a proper indicator of the storage capability of the given disk. The formatting information accounts for about 15 percent of the total information that can be stored on a disk. It comprises the sector headers, the ECC bits, and intersector gaps. In a typical computer, the disk is subsequently divided into logical partitions. There must be at least one such partition, called the primary partition. There may be a number of additional partitions.

Figure 5.30 indicates that each track has the same number of sectors. So all tracks have the same storage capacity. Thus, the stored information is packed more densely on inner tracks than on outer tracks. This arrangement is used in many disks because it simplifies the electronic circuits needed to access the data. But, it is possible to increase the storage density by placing more sectors on outer tracks, which have longer circumference, at the expense of more complicated access circuitry. This scheme is used in large disks.

Access Time

There are two components involved in the time delay between receiving an address and the beginning of the actual data transfer. The first, called the *seek time*, is the time required to move the read/write head to the proper track. This depends on the initial position of the head relative to the track specified in the address. Average values are in the 5- to 8-ms range. The second component is the *rotational delay*, also called *latency time*. This is the amount of time that elapses after the head is positioned over the correct track until the starting position of the addressed sector passes under the read/write head. On average, this is the time for half a rotation of the disk. The sum of these two delays is called the disk *access time*. If only a few sectors of data are moved in a single operation, the access time is at least an order of magnitude longer than the actual data transfer period.

Typical Disks

A 3.5-inch (diameter) high-capacity, high-data-rate disk available today may have the following representative parameters. There are 20 data-recording surfaces with 15,000 tracks per surface. There is an average of 400 sectors per track, and each sector contains 512 bytes of data. Hence, the total capacity of the formatted disk is $20 \times 15{,}000 \times 400 \times 512 \approx 60 \times 10^9 = 60$ gigabytes. The average seek time is 6 ms. The platters rotate at 10,000 revolutions per minute, so that the average latency is 3 ms, which is the time for a half-rotation. The average internal transfer rate, from a track to the data buffer in the disk controller, is 34 Mbytes/s. When connected to a SCSI bus, a drive

of this type may have an external transfer rate of 160 Mbytes/s. Thus, a buffering scheme is needed to deal with the difference in transfer speeds, as explained in the next section.

There are also some very small disks. For example, a one-inch disk may store one gigabyte of data. Its physical size is comparable to a matchbook and it weighs less than an ounce. Such disks are attractive for use in portable equipment and hand-held devices. In a digital camera, such a disk could store 1000 photographs. It is interesting to observe that the first disk drive that had a 1-gigabyte capacity was produced by IBM in 1980. It was the size of a kitchen appliance. It weighed 250 kilograms and cost $40,000.

Data Buffer/Cache

A disk drive is connected to the rest of a computer system using some standard interconnection scheme. Normally, a standard bus, such as the SCSI bus discussed in Section 4.7.2, is used. A disk drive that incorporates the required SCSI interface circuitry is usually referred to as a SCSI drive. The SCSI bus is capable of transferring data at much higher rates than the rate at which data can be read from disk tracks. An efficient way to deal with the possible differences in transfer rates between the disk and the SCSI bus is to include a *data buffer* in the disk unit. This buffer is a semiconductor memory, capable of storing a few megabytes of data. The requested data are transferred between the disk tracks and the buffer at a rate dependent on the rotational speed of the disk. Transfers between the data buffer and other devices connected to the bus, normally the main memory, can then take place at the maximum rate allowed by the bus.

The data buffer can also be used to provide a caching mechanism for the disk. When a read request arrives at the disk, the controller can first check to see if the desired data are already available in the cache (buffer). If so, the data can be accessed and placed on the SCSI bus in microseconds rather than milliseconds. Otherwise, the data are read from a disk track in the usual way and stored in the cache. Since it is likely that a subsequent read request will be for data that sequentially follow the currently accessed data, the disk controller can cause more data than needed to be read and placed into the cache, thus potentially shortening the response time for the next request. The cache is typically large enough to store entire tracks of data, so a possible strategy is to begin transferring the contents of the track into the data buffer as soon as the read/write head is positioned over the desired track.

Disk Controller

Operation of a disk drive is controlled by a *disk controller* circuit, which also provides an interface between the disk drive and the bus that connects it to the rest of the computer system. The disk controller may be used to control more than one drive. Figure 5.31 shows a disk controller which controls two disk drives.

A disk controller that is connected directly to the processor system bus, or to an expansion bus such as PCI, contains a number of registers that can be read and written by the operating system. Thus, communication between the OS and the disk controller is achieved in the same manner as with any I/O interface, as discussed in Chapter 4. The disk controller uses the DMA scheme to transfer data between the disk and the main memory. Actually, these transfers are from/to the data buffer, which is implemented

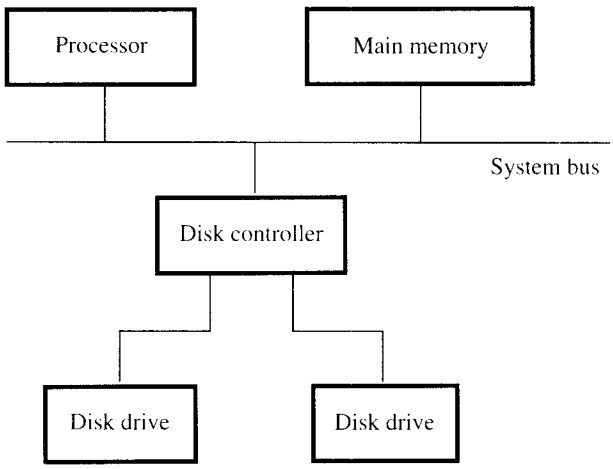


Figure 5.31 Disks connected to the system bus.

as a part of the disk controller module. The OS initiates the transfers by issuing Read and Write requests, which entail loading the controller's registers with the necessary addressing and control information, typically:

Main memory address — The address of the first main memory location of the block of words involved in the transfer.

Disk address — The location of the sector containing the beginning of the desired block of words.

Word count — The number of words in the block to be transferred.

The disk address issued by the OS is a logical address. The corresponding physical address on the disk may be different. For example, bad sectors may be detected when the disk is formatted. The disk controller keeps track of such sectors and substitutes other sectors instead. Normally, a few spare sectors are kept on each track, or on another track in the same cylinder, to be used as substitutes for the bad sectors.

On the disk drive side, the controller's major functions are:

Seek — Causes the disk drive to move the read/write head from its current position to the desired track.

Read — Initiates a Read operation, starting at the address specified in the disk address register. Data read serially from the disk are assembled into words and placed into the data buffer for transfer to the main memory. The number of words is determined by the word count register.

Write — Transfers data to the disk, using a control method similar to that for the Read operations.

Error checking — Computes the error correcting code (ECC) value for the data read from a given sector and compares it with the corresponding ECC value read from the disk. In case of a mismatch, it corrects the error if possible; otherwise, it

raises an interrupt to inform the OS that an error has occurred. During a write operation, the controller computes the ECC value for the data to be written and stores this value on the disk.

If a disk drive is connected to a bus that uses packetized transfers, then the controller must be capable of handling such transfers. For example, a controller for a SCSI drive conforms to the SCSI bus protocol described in Chapter 4.

Software and Operating System Implications

All data transfer activities involving disks are initiated by the operating system. The disk is a nonvolatile storage medium, so the OS itself is stored on a disk. During normal operation of a computer, parts of the OS are loaded into the main memory and executed as needed.

When power is turned off, the contents of the main memory are lost. When the power is turned on again, the OS has to be loaded into the main memory, which takes place as part of a process known as *booting*. To initiate booting, a tiny part of main memory is implemented as a nonvolatile ROM. This ROM stores a small *monitor* program that can read and write main memory locations as well as read one block of data stored on the disk at address 0. This block, referred to as the *boot block*, contains a loader program. After the boot block is loaded into memory by the ROM monitor program, it loads the main parts of the OS into the main memory.

Disk accesses are very slow compared to main memory accesses, mostly due to long seek times. After the OS initiates a disk transfer operation, it normally attempts to switch execution to some other task, to make use of the time it would otherwise spend waiting for the transfer to complete. The disk controller informs the OS when the transfer is completed by raising an interrupt.

In a computer system that has multiple disks, the OS may require transfers from several disks. Efficient operation is achieved if the DMA transfer from/to one disk occurs while another disk is doing a seek. The OS can schedule such overlapped I/O activities.

Floppy Disks

The devices previously discussed are known as hard or rigid disk units. *Floppy disks* are smaller, simpler, and cheaper disk units that consist of a flexible, removable, plastic *diskette* coated with magnetic material. The diskette is enclosed in a plastic jacket, which has an opening where the read/write head makes contact with the diskette. A hole in the center of the diskette allows a spindle mechanism in the disk drive to position and rotate the diskette.

One of the simplest schemes used in the first floppy disks for recording data is phase or Manchester encoding mentioned earlier. Disks encoded in this way are said to have *single density*. A more complicated variant of this scheme, called *double density*, is most often used in current standard floppy disks. It increases the storage density by a factor of 2 but also requires more complex circuits in the disk controller.

The main feature of floppy disks is their low cost and shipping convenience. However, they have much smaller storage capacities, longer access times, and higher failure rates than hard disks. Current standard floppy disks are 3.25 inches in diameter and store

1.44 or 2 Mbytes of data. Larger super-floppy disks are also available. One type of such disks, known as the *zip* disk, can store more than 100 Mbytes. In recent years, the attraction of floppy-disk technology has been diminished by the emergence of rewritable compact disks, which we discuss below.

RAID Disk Arrays

Processor speeds have increased dramatically during the past decade. Processor performance has doubled every 18 months. Semiconductor memory speeds have improved more modestly. The smallest relative improvement in terms of speed has been in disk storage devices, for which access times are still on the order of milliseconds. Of course, there has been a spectacular improvement in the storage capacity of these devices.

High-performance devices tend to be expensive. Sometimes it is possible to achieve very high performance at a reasonable cost by using a number of low-cost devices operating in parallel. In Chapter 12 we will see how many commodity processors can be used to implement a high-performance multiprocessor computer system. Multiple magnetic disk drives can be used to provide a high-performance storage unit.

In 1988, researchers at the University of California-Berkeley proposed a storage system based on multiple disks [5]. They called it RAID, for Redundant Array of Inexpensive Disks. Using multiple disks also makes it possible to improve the reliability of the overall system. Six different configurations were proposed. They are known as RAID levels even though there is no hierarchy involved.

RAID 0 is the basic configuration intended to enhance performance. A single large file is stored in several separate disk units by breaking the file up into a number of smaller pieces and storing these pieces on different disks. This is called *data striping*. When the file is accessed for a read, all disks can deliver their data in parallel. The total transfer time of the file is equal to the transfer time that would be required in a single-disk system divided by the number of disks used in the array. However, access time, that is, the seek and rotational delay needed to locate the beginning of the data on each disk, is not reduced. In fact, since each disk operates independently of the others, access times vary, and buffering of the accessed pieces of data is needed so that the complete file can be reassembled and sent to the requesting processor as a single entity. This is the simplest possible disk array operation in which only data-flow-time performance is improved.

RAID 1 is intended to provide better reliability by storing identical copies of data on two disks rather than just one. The two disks are said to be mirrors of each other. Then, if one disk drive fails, all read and write operations are directed to its mirror drive. This is a costly way to improve the reliability because all disks are duplicated.

RAID 2, RAID 3, and RAID 4 levels achieve increased reliability through various parity checking schemes without requiring a full duplication of disks. All of the parity information is kept on one disk.

RAID 5 also makes use of a parity-based error-recovery scheme. However, the parity information is distributed among all disks, rather than being stored on one disk.

Some hybrid arrangements have subsequently been developed. For example, RAID 10 is an array that combines the features of RAID 0 and RAID 1. A more detailed treatment of RAID schemes can be found in References [6] to [10].

The RAID concept has gained commercial acceptance. For example, the Dell Computer Corporation offers products based on RAID 0, RAID 1, RAID 5, and RAID 10. Finally, we should note that as the price of magnetic disk drives has decreased greatly during the past few years, it may be inappropriate to refer to "inexpensive" disks in RAID. Indeed, the term RAID has been redefined by the industry to refer to "independent" disks.

Commodity Disk Considerations

Most disk units are designed to connect to standard buses. The performance of a disk unit depends on its internal structure and the interface used to connect it to the rest of the system. The cost depends largely on the storage capacity, but it is also affected greatly by the sales volume of a particular product.

ATA/EIDE Disks The most widely used computers are the personal computers (PCs) introduced by IBM in 1980, familiarly known as IBM PCs. A disk interface suitable for connection to the IBM PC bus was developed. Its present (enhanced) version has become a standard known as EIDE (Enhanced Integrated Drive Electronics) or as ATA (Advanced Technology Attachment). Many disk manufacturers have a range of disks that have EIDE/ATA interfaces. Such disks can be connected directly to the PCI bus (discussed in Section 4.7.1), which is used in many PCs. In fact, Intel's Pentium chip sets include a controller that allows EIDE/ATA disks to be connected directly to the motherboard. A significant advantage of EIDE/ATA drives is their low price, due to their use in the PC market. One of their main drawbacks is that a separate controller is needed for each drive if two drives are to be used concurrently to improve performance.

SCSI Disks As we have already explained in a previous example, many disks have an interface designed for connection to a standard SCSI bus. These disks tend to be more expensive, but they exhibit better performance, made possible by the advantages of the SCSI bus in comparison with the PCI bus. Concurrent accesses can be made to multiple disk drives because the drive's interface is actively connected to the SCSI bus only when the drive is ready for a data transfer. This is especially useful in applications where there is a large number of requests for small files, which is often the case in computers used as file servers.

RAID Disks RAID disks offer excellent performance and provide a large and reliable storage. They are used either in high-performance computers, or in systems where a higher than normal degree of reliability is required. However, as their price drops to a more affordable level, they are becoming attractive for use even in average-size computer systems.

5.9.2 OPTICAL DISKS

Large storage devices can also be implemented using optical means. The familiar compact disk (CD), used in audio systems, was the first practical application of this technology. Soon after, the optical technology was adapted to the computer environment to provide high-capacity read-only storage referred to as CD-ROM.

The first generation of CDs was developed in the mid-1980s by the Sony and Philips companies, which also published a complete specification for these devices. The technology exploited the possibility of using digital representation of analog sound signals. To provide high-quality sound recording and reproduction, 16-bit samples of the analog signal are taken at a rate of 44,100 samples per second. This sampling rate is twice the highest frequency in the original sound signal, thus allowing for accurate reconstruction. The CDs were required to hold at least an hour of music. The first version was designed to hold up to 75 minutes, which requires a total of about 3×10^9 bits (3 gigabits) of storage. Since then, higher-capacity devices have been developed. A video CD is capable of storing a full-length movie. This requires approximately an order of magnitude more bit-storage capacity than that of audio CDs. Multimedia CDs are also suitable for storing large amounts of computer data.

CD Technology

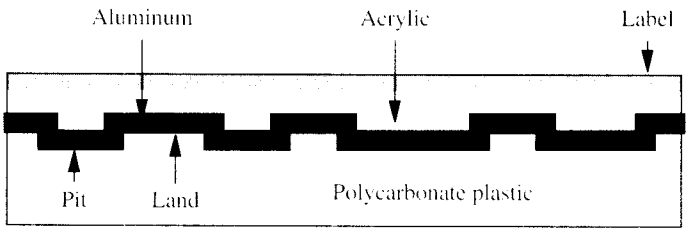
The optical technology that is used for CD systems is based on a laser light source. A laser beam is directed onto the surface of the spinning disk. Physical indentations in the surface are arranged along the tracks of the disk. They reflect the focused beam toward a photodetector, which detects the stored binary patterns.

The laser emits a coherent light beam that is sharply focussed on the surface of the disk. Coherent light consists of synchronized waves that have the same wavelength. If a coherent light beam is combined with another beam of the same kind, and the two beams are in phase, then the result will be a brighter beam. But, if the waves of the two beams are 180 degrees out of phase, they will cancel each other. Thus, if a photodetector is used to detect the beams, it will detect a bright spot in the first case and a dark spot in the second case.

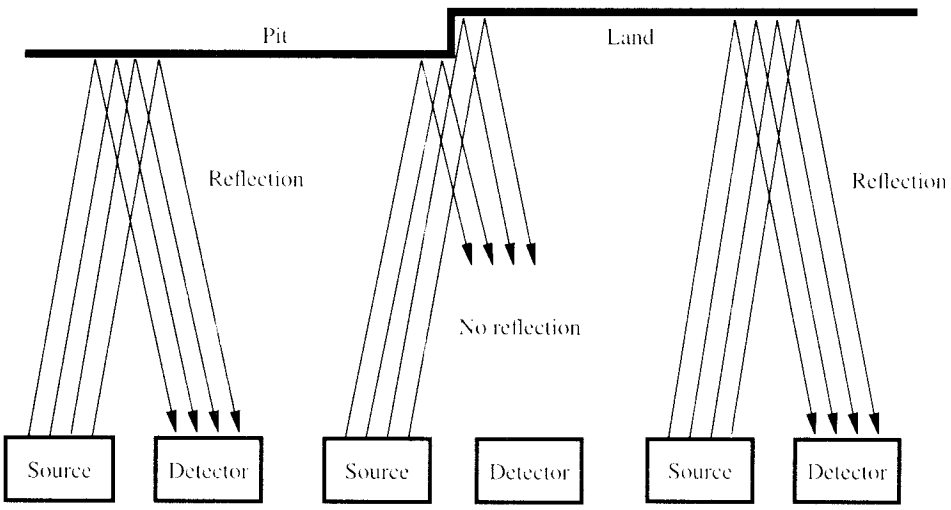
A cross-section of a small portion of a CD is shown in Figure 5.32*a*. The bottom layer is polycarbonate plastic, which functions as a clear glass base. The surface of this plastic is programmed to store data by indenting it with *pits*. The unindented parts are called *lands*. A thin layer of reflecting aluminum material is placed on top of a programmed disk. The aluminum is then covered by a protective acrylic. Finally, the topmost layer is deposited and stamped with a label. The total thickness of the disk is 1.2 mm. Almost all of it is contributed by the polycarbonate plastic. The other layers are very thin.

The laser source and the photodetector are positioned below the polycarbonate plastic. The emitted beam travels through this plastic, reflects off the aluminum layer, and travels back toward the photodetector. Note that from the laser side, the pits actually appear as bumps with respect to the lands.

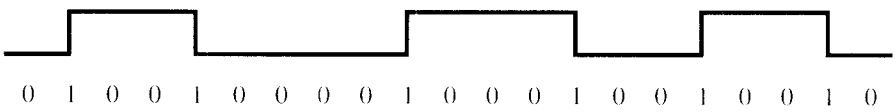
Figure 5.32*b* shows what happens as the laser beam scans across the disk and encounters a transition from a pit to a land. Three different positions of the laser source and the detector are shown, as would occur when the disk is rotating. When the light reflects solely from the pit, or solely from the land, the detector will see the reflected beam as a bright spot. But, a different situation arises when the beam moves through the edge where the pit changes to the land, and vice versa. The pit is recessed one quarter of the wavelength of the light. Thus, the reflected wave from the pit will be 180 degrees out of phase with the wave reflected from the land, cancelling each other. Hence, at



(a) Cross-section



(b) Transition from pit to land



(c) Stored binary pattern

Figure 5.32 Optical disk.

the pit-land and land-pit transitions the detector will not see a reflected beam and will detect a dark spot.

Figure 5.32c depicts several transitions between lands and pits. If each transition, detected as a dark spot, is taken to denote the binary value 1, and the flat portions represent 0s, then the detected binary pattern will be as shown in the figure. This pattern is not a direct representation of the stored data. CDs use a complex encoding

scheme to represent data. Each byte of data is represented by a 14-bit code, which provides considerable error detection capability. We will not delve into details of this code.

The pits are arranged along tracks on the surface of the disk. Actually, there is just one physical track, spiraling from the middle of the disk toward the outer edge. But, it is customary to refer to each circular path spanning 360 degrees as a separate track, which is analogous to the terminology used for magnetic disks. The CD is 120 mm in diameter. There is a 15-mm hole in the center. Data are stored on tracks that cover the area from 25-mm radius to 58-mm radius. The space between the tracks is 1.6 microns. Pits are 0.5 microns wide and 0.8 to 3 microns long. There are more than 15,000 tracks on a disk. If the entire track spiral were unraveled, it would be over 5 km long!

These numbers indicate a track density of about 6000 tracks/cm, which is much higher than the density achievable in magnetic disks. The density ranges from 800 to 2000 tracks/cm in hard disks, and is less than 40 tracks/cm in floppy disks.

CD-ROM

Since information is stored in binary form in CDs, they are suitable for use as a storage medium in computer systems. The biggest problem is to ensure the integrity of stored data. Because the pits are very small, it is difficult to implement all of the pits perfectly. In audio and video applications, some errors in data can be tolerated because they are unlikely to affect the reproduced sound or image in a perceptible way. However, such errors are not acceptable in computer applications. Since physical imperfections cannot be avoided, it is necessary to use additional bits to provide error checking and correcting capability. CDs used in computer applications have such capability. They are called *CD-ROMs*, because after manufacture their contents can only be read, as with semiconductor ROM chips.

Stored data are organized on CD-ROM tracks in the form of blocks that are called *sectors*. There are several different formats for a sector. One format, known as Mode 1, uses 2352-byte sectors. There is a 16-byte header that contains a synchronization field used to detect the beginning of the sector and addressing information used to identify the sector. This is followed by 2048 bytes of stored data. At the end of the sector, there are 288 bytes used to implement the error-correcting scheme. The number of sectors per track is variable; there are more sectors on the longer outer tracks.

Error detection and correction is done at more than one level. As mentioned in the introduction to CDs, each byte of stored information is encoded using a 14-bit code that has some error-correcting capability. This code can correct single-bit errors. Errors that occur in short bursts, affecting several bits, are detected and corrected using the error-checking bits at the end of the sector.

CD-ROM drives operate at a number of different rotational speeds. The basic speed, known as 1X, is 75 sectors per second. This provides a data rate of 153,600 bytes/s (150 Kbytes/s), using the Mode 1 format. With this speed and format, a CD-ROM based on the standard CD designed for 75 minutes of music has a data storage capacity of about 650 Mbytes. Note that the speed of the drive affects only the data transfer rate but not the storage capacity of the disk. Higher speed CD-ROM drives are identified in relation to the basic speed. Thus, a 40X CD-ROM has a data transfer rate that is 40 times higher than that of the 1X CD-ROM. Observe that this transfer rate ($<$6 Mbytes/s)

is considerably lower than the transfer rates in magnetic hard disks, which are in the range of tens of megabytes per second. Another big difference in performance is the seek time, which in CD-ROMs may be several hundred milliseconds. So, in terms of performance, CD-ROMs are clearly inferior to magnetic disks. Their attraction lies in the small physical size, low cost, and ease of handling as a removable and transportable mass-storage medium.

The importance of CD ROMs for computer systems stems from their large storage capacity and fast access times compared to other inexpensive portable media, such as floppy disks and magnetic tapes. They are widely used for the distribution of software, databases, large texts (books), application programs, and video games.

CD-Recordables

Previously described CDs are read-only devices in which the information is stored using a special procedure. First, a master disk is produced using a high-power laser to burn holes that correspond to the required pits. A mold is then made from the master disk, which has bumps in the place of holes. This is followed by injecting molten polycarbonate plastic into the mold to make a CD that has the same pattern of holes (pits) as the master disk. This process is clearly suitable only for volume production of CDs.

A new type of CD was developed in the late 1990s on which data can be easily recorded by a computer user. It is known as CD-Recordable (CD-R). A spiral track is implemented on a disk during the manufacturing process. A laser in a CD-R drive is used to burn pits into an organic dye on the track. When a burned spot is heated beyond a critical temperature, it becomes opaque. Such burned spots reflect less light when subsequently read. The written data are stored permanently. Unused portions of a disk can be used to store additional data at a later time.

CD-ReWritables

The most flexible CDs are those that can be written multiple times by the user. They are known as CD-RWs (CD-ReWritables).

The basic structure of CD-RWs is similar to the structure of CD-Rs. Instead of using an organic dye in the recording layer, an alloy of silver, indium, antimony and tellurium is used. This alloy has interesting and useful behavior when it is heated and cooled. If it is heated above its melting point (500 degrees C) and then cooled down, it goes into an amorphous state in which it absorbs light. But, if it is heated only to about 200 degrees C and this temperature is maintained for an extended period, a process known as *annealing* takes place, which leaves the alloy in a crystalline state that allows light to pass through. If the crystalline state represents land area, pits can be created by heating selected spots past the melting point. The stored data can be erased using the annealing process, which returns the alloy to a uniform crystalline state. A reflective material is placed above the recording layer to reflect the light when the disk is read.

The CD-RW drive uses three different laser powers. The highest power is used to record the pits. The middle power is used to put the alloy into its crystalline state; it is referred to as the "erase power." The lowest power is used to read the stored information. There is a limit on how many times a CD-RW disk can be rewritten. Presently, this can be done up to 1000 times.

CD-RW drives can usually also deal with other compact disk media. They can read CD-ROMs, and both read and write CD-Rs. They are designed to meet the requirements of standard interconnection interfaces, such as EIDE, SCSI, and USB.

CD-RWs provide a low-cost storage medium. They are suitable for archival storage of information that may range from databases to photographic images. They can be used for low-volume distribution of information, just like CD-Rs. The CD-RW drives are now fast enough to be used for daily hard disk backup purposes. The CD-RW technology has made CD-Rs less relevant because it offers superior capability at only slightly higher cost.

DVD Technology

The success of CD technology and the continuing quest for greater storage capability has led to the development of DVD (Digital Versatile Disk) technology. The first DVD standard was defined in 1996 by a consortium of companies. The objective is to be able to store a full-length movie on one side of a DVD disk.

The physical size of a DVD disk is the same as for CDs. The disk is 1.2 mm thick, and it is 120 mm in diameter. Its storage capacity is made much larger than that of CDs by several design changes:

- A red light laser with a wavelength of 635 nm is used instead of the infrared light laser used in CDs, which has a wavelength of 780 nm. The shorter wavelength makes it possible to focus the light to a smaller spot.
- Pits are smaller, having a minimum length of 0.4 micron.
- Tracks are placed closer together; the distance between tracks is 0.74 micron.

Using these improvements leads to a DVD capacity of 4.7 Gbytes.

Further increases in capacity have been achieved by going to two-layered and two-sided disks. The single-layered single-sided disk, defined in the standard as DVD-5, has a structure that is almost the same as the CD in Figure 5.32*a*. A double-layered disk makes use of two layers on which tracks are implemented on top of each other. The first layer is the clear base, as in CD disks. But, instead of using reflecting aluminum, the lands and pits of this layer are covered by a translucent material that acts as a semireflector. The surface of this material is then also programmed with indented pits to store data. A reflective material is placed on top of the second layer of pits and lands. The disk is read by focusing the laser beam on the desired layer. When the beam is focused on the first layer, sufficient light is reflected by the translucent material to detect the stored binary patterns. When the beam is focused on the second layer, the light reflected by the reflective material corresponds to the information stored on this layer. In both cases, the layer on which the beam is not focused reflects a much smaller amount of light, which is eliminated by the detector circuit as noise. The total storage capacity of both layers is 8.5 Gbytes. This disk is called DVD-9 in the standard.

Two single-sided disks can be put together to form a sandwich-like structure where the top disk is turned upside down. This can be done with single-layered disks, as specified in DVD-10, giving a composite disk with a capacity of 9.4 Gbytes. It can also be done with the double-layered disks, as specified in DVD-18, yielding a capacity of 17 Gbytes.

Access times for DVD drives are similar to CD drives. However, when the DVD disks rotate at the same speed, the data transfer rates are much higher because of the higher density of pits.

DVD-RAM

A rewritable version of DVD devices, known as DVD-RAM, has also been developed. It provides a large storage capacity. Its only disadvantages are the higher price and the relatively slow writing speed. To ensure that the data have been recorded correctly on the disk, a process known as write verification is performed. This is done by the DVD-RAM drive, which reads the stored contents and checks them against the original data. A detailed discussion of optical disk technology can be found in Reference [11].

5.9.3 MAGNETIC TAPE SYSTEMS

Magnetic tapes are suited for off-line storage of large amounts of data. They are typically used for hard disk backup purposes and for archival storage. Magnetic-tape recording uses the same principle as used in magnetic-disk recording. The main difference is that the magnetic film is deposited on a very thin 0.5- or 0.25-inch wide plastic tape. Seven or 9 bits (corresponding to one character) are recorded in parallel across the width of the tape, perpendicular to the direction of motion. A separate read/write head is provided for each bit position on the tape, so that all bits of a character can be read or written in parallel. One of the character bits is used as a parity bit.

Data on the tape are organized in the form of *records* separated by gaps, as shown in Figure 5.33. Tape motion is stopped only when a record gap is underneath the read/write heads. The record gaps are long enough to allow the tape to attain its normal speed before the beginning of the next record is reached. If a coding scheme such as that in Figure 5.29*c* is used for recording data on the tape, record gaps are identified as areas where there is no change in magnetization. This allows record gaps to be detected independently of the recorded data. To help users organize large amounts of data, a group of related records is called a *file*. The beginning of a file is identified by a *file mark*, as shown in Figure 5.33. The file mark is a special single- or multiple-character

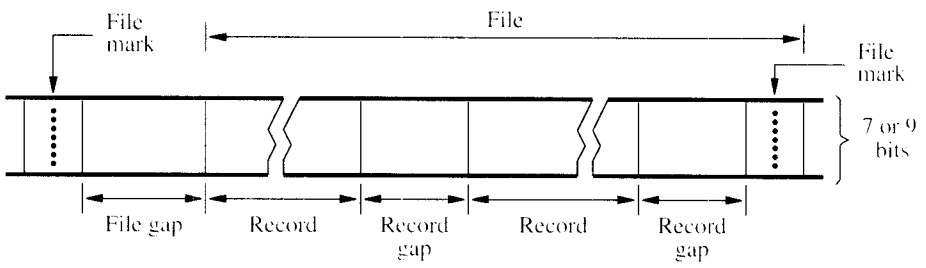


Figure 5.33 Organization of data on magnetic tape.

record, usually preceded by a gap longer than the interrecord gap. The first record following a file mark can be used as a *header* or *identifier* for this file. This allows the user to search a tape containing a large number of files for a particular file.

The controller of a magnetic tape drive enables the execution of a number of control commands in addition to read and write commands. Control commands include the following operations:

- Rewind tape
- Rewind and unload tape
- Erase tape
- Write tape mark
- Forward space one record
- Backspace one record
- Forward space one file
- Backspace one file

The tape mark referred to in the operation "Write tape mark" is similar to a file mark except that it is used for identifying the beginning of the tape. The end of the tape is sometimes identified by the EOT (end of tape) character (see Appendix E).

Two methods of formatting and using tapes are available. In the first method, the records are variable in length. This allows efficient use of the tape, but it does not permit updating or overwriting of records in place. The second method is to use fixed-length records. In this case, it is possible to update records in place. Although this may seem to be a significant advantage, in practice it has turned out to be of little importance. The most common uses of tapes are backing up information on magnetic disks and archival storage of data. In these applications, a tape is written from the beginning to the end so that the size of the records is irrelevant.

Cartridge Tape System

Tape systems have been developed for backup of on-line disk storage. One such system uses an 8-mm video format tape housed in a cassette. These units are called cartridge tapes. They have capacities in the range of 2 to 5 gigabytes and handle data transfers at the rate of a few hundred kilobytes per second. Reading and writing is done by a helical scan system operating across the tape, similar to that used in video cassette tape drives. Bit densities of tens of millions of bits per square inch are achievable. Multiple-cartridge systems are available that automate the loading and unloading of cassettes so that tens of gigabytes of on-line storage can be backed up unattended.

5.10 CONCLUDING REMARKS

The memory is a major component in any computer. Its capacity and speed characteristics are important in determining the performance of a computer. In this chapter, we presented the most important technological and organizational details of memory design.

Developments in semiconductor technology have led to spectacular improvements in the speed and capacity of memory chips, accompanied by a large decrease in the cost per bit. But processor chips have evolved even more spectacularly. In particular, improvement in the operating speed of processor chips has outpaced that of memory chips. To exploit fully the capability of a modern processor, the computer must have a large and fast memory. Since cost is also important, the memory cannot be designed simply by using fast SRAM components. As we saw in this chapter, the solution lies in the memory hierarchy.

Today, an affordable large main memory is implemented with dynamic memory chips. Such a memory may be an order of magnitude slower than a fast processor, in terms of clock cycles, which makes it imperative to use an SRAM cache memory to reduce the effective memory access time seen by the processor. Memory latency is an important parameter in determining the performance of a computer. Much research effort has focused on developing schemes that minimize the effect of memory latency. We described how write buffers and prefetching can reduce the impact of latency by performing less urgent memory accesses at times when no higher-priority memory accesses (caused by read misses) are required. The effect of memory latency can also be reduced if blocks of consecutive words are accessed at one time; new memory chips are being developed to exploit this fact.

Secondary storage, in the form of magnetic and optical disks, provides the largest capacity in the memory hierarchy. The virtual memory mechanism makes interaction between the disk and the main memory transparent to the user. Hardware support for virtual memory has become a standard feature of modern processors.

Magnetic disks are a fascinating example of evolution in computer technology. They have always been the slow part of the memory hierarchy. At various times, the continued viability of magnetic disks has been questioned as new technologies appeared to have greater promise for future development. In the early 1980s, it seemed that "magnetic bubble" memories would pose a large threat. More recent contenders are flash drives and optical disks. However, instead of being pushed aside, the magnetic disk technology keeps getting better. The disk drives are continuously becoming larger in storage capacity, smaller in physical size, and cheaper in terms of the cost per bit stored.

PROBLEMS

5.1 Give a block diagram similar to the one in Figure 5.10 for a 8M × 32 memory using 512K × 8 memory chips.

5.2 Consider the dynamic memory cell of Figure 5.6. Assume that $C = 50$ femtofarads (10^{-15} F) and that leakage current through the transistor is about 9 picoamperes (10^{-12} A). The voltage across the capacitor when it is fully charged is equal to 4.5 V. The cell must be refreshed before this voltage drops below 3 V. Estimate the minimum refresh rate.

5.3 In the bottom right corner of Figure 5.8 there are data input and data output registers. Draw a circuit that can implement one bit of each of these registers, and show the

required connections to the block "Read/Write circuits & latches" on one side and the data bus on the other side.

5.4 Consider a main memory constructed with SDRAM chips that have timing requirements depicted in Figure 5.9, except that the burst length is 8. Assume that 32 bits of data are transferred in parallel. If a 133-MHz clock is used, how much time does it take to transfer:

(*a*) 32 bytes of data

(*b*) 64 bytes of data

What is the latency in each case?

5.5 Criticize the following statement: "Using a faster processor chip results in a corresponding increase in performance of a computer even if the main memory speed remains the same."

5.6 A program consists of two nested loops — a small inner loop and a much larger outer loop. The general structure of the program is given in Figure P5.1. The decimal memory addresses shown delineate the location of the two loops and the beginning and end of the total program. All memory locations in the various sections, 17–22, 23–164, 165–239, and so on, contain instructions to be executed in straight-line sequencing. The program is to be run on a computer that has an instruction cache organized in the direct-mapped manner (see Figure 5.15) and that has the following parameters:

Main memory size	64K words
Cache size	1K words
Block size	128 words

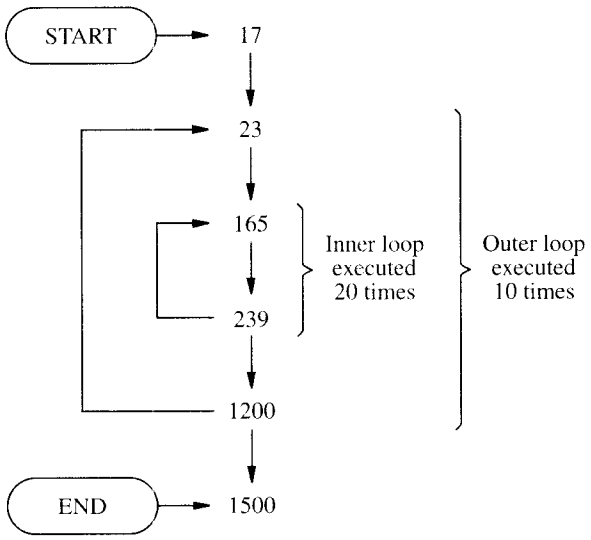


Figure P5.1 A program structure for Problem 5.6.

The cycle time of the main memory is 10τ s, and the cycle time of the cache is 1τ s.

(*a*) Specify the number of bits in the TAG, BLOCK, and WORD fields in main memory addresses.

(*b*) Compute the total time needed for instruction fetching during execution of the program in Figure P5.1.

5.7 A computer uses a small direct-mapped cache between the main memory and the processor. The cache has four 16-bit words, and each word has an associated 13-bit tag, as shown in Figure P5.2*a*. When a miss occurs during a read operation, the requested word is read from the main memory and sent to the processor. At the same time, it is copied into the cache, and its block number is stored in the associated tag. Consider the following loop in a program where all instructions and operands are 16 bits long:

```
LOOP   Add         (R1)+,R0
       Decrement   R2
       BNE         LOOP
```

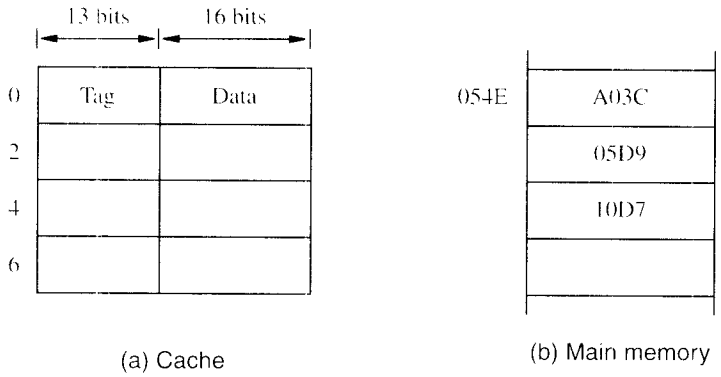


Figure P5.2 Cache and main memory contents in Problem 5.7.

Assume that, before this loop is entered, registers R0, R1, and R2 contain 0, 054E, and 3, respectively. Also assume that the main memory contains the data shown in Figure P5.2*b*, where all entries are given in hexadecimal notation. The loop starts at location LOOP = 02EC.

(*a*) Show the contents of the cache at the end of each pass through the loop.

(*b*) Assume that the access time of the main memory is 10τ and that of the cache is 1τ. Calculate the execution time for each pass. Ignore the time taken by the processor between memory cycles.

5.8 Repeat Problem 5.7, assuming only instructions are stored in the cache. Data operands are fetched directly from the main memory and not copied into the cache. Why does this choice lead to faster execution than when both instructions and data are written into the cache?

5.9 A block-set-associative cache consists of a total of 64 blocks divided into 4-block sets. The main memory contains 4096 blocks, each consisting of 128 words.

(*a*) How many bits are there in a main memory address?

(*b*) How many bits are there in each of the TAG, SET, and WORD fields?

5.10 A computer system has a main memory consisting of 1M 16-bit words. It also has a 4K-word cache organized in the block-set-associative manner, with 4 blocks per set and 64 words per block.

(*a*) Calculate the number of bits in each of the TAG, SET, and WORD fields of the main memory address format.

(*b*) Assume that the cache is initially empty. Suppose that the processor fetches 4352 words from locations 0, 1, 2, . . . , 4351, in that order. It then repeats this fetch sequence nine more times. If the cache is 10 times faster than the main memory, estimate the improvement factor resulting from the use of the cache. Assume that the LRU algorithm is used for block replacement.

5.11 Repeat Problem 5.10, assuming that whenever a block is to be brought from the main memory and the corresponding set in the cache is full, the new block replaces the most recently used block of this set.

5.12 Section 5.5.3 illustrates the effect of different cache-mapping techniques, using the program in Figure 5.19. Suppose that this program is changed so that in the second loop the elements are handled in the same order as in the first loop, that is, the control for the second loop is specified as

for $i := 0$ **to** 9 **do**

Derive the equivalents of Figures 5.20 through 5.22 for this program. What conclusions can be drawn from this exercise?

5.13 A byte-addressable computer has a small data cache capable of holding eight 32-bit words. Each cache block consists of one 32-bit word. When a given program is executed, the processor reads data from the following sequence of hex addresses:

200, 204, 208, 20C, 2F4, 2F0, 200, 204, 218, 21C, 24C, 2F4

This pattern is repeated four times.

(*a*) Show the contents of the cache at the end of each pass through this loop if a direct-mapped cache is used. Compute the hit rate for this example. Assume that the cache is initially empty.

(*b*) Repeat part (*a*) for an associative-mapped cache that uses the LRU replacement algorithm.

(*c*) Repeat part (*a*) for a four-way set-associative cache.

5.14 Repeat Problem 5.13, assuming that each cache block consists of two 32-bit words. For part (*c*), use a two-way set-associative cache.

5.15 How might the value of k in the interleaved memory system of Figure 5.25*b* influence block size in the design of a cache memory to be used with the system?

5.16 In many computers the cache block size is in the range of 32 to 128 bytes. What would be the main advantages and disadvantages of making the size of cache blocks larger or smaller?

5.17 Consider the effectiveness of interleaving with respect to the size of cache blocks. Using calculations similar to those in Section 5.6.2, estimate the performance improvement for block sizes of 16, 8, and 4 words. Assume that all words loaded into the cache are accessed by the processor at least once.

5.18 Assume a computer has L1 and L2 caches, as discussed in Section 5.6.3. The cache blocks consist of 8 words. Assume that the hit rate is the same for both caches and that it is equal to 0.95 for instructions and 0.90 for data. Assume also that the times needed to access an 8-word block in these caches are $C_1 = 1$ cycle and $C_2 = 10$ cycles.

(*a*) What is the average access time experienced by the processor if the main memory uses interleaving? Assume that the memory access parameters are as described in Section 5.6.1.

(*b*) What is the average access time if the main memory is not interleaved?

(*c*) What is the improvement obtained with interleaving?

5.19 Repeat Problem 5.18, assuming that a cache block consists of 4 words. Estimate an appropriate value for C_2, assuming that the L2 cache is implemented with SRAM chips.

5.20 Consider the following analogy for the concept of caching. A serviceman comes to a house to repair the heating system. He carries a toolbox that contains a number of tools that he has used recently in similar jobs. He uses these tools repeatedly, until he reaches a point where other tools are needed. It is likely that he has the required tools in his truck outside the house. But, if the needed tools are not in the truck, he must go to his shop to get them.

Suppose we argue that the toolbox, the truck, and the shop correspond to the L1 cache, the L2 cache, and the main memory of a computer. How good is this analogy? Discuss its correct and incorrect features.

5.21 A 1024 × 1024 array of 32-bit numbers is to be "normalized" as follows. For each column, the largest element is found and all elements of the column are divided by this maximum value. Assume that each page in the virtual memory consists of 4K bytes, and that 1M bytes of the main memory are allocated for storing data during this computation. Suppose that it takes 40 ms to load a page from the disk into the main memory when a page fault occurs.

(*a*) How many page faults would occur if the elements of the array are stored in column order in the virtual memory?

(*b*) How many page faults would occur if the elements are stored in row order?

(*c*) Estimate the total time needed to perform this normalization for both arrangements (*a*) and (*b*).

5.22 Consider a computer system in which the available pages in the physical memory are divided among several application programs. When all the pages allocated to a program are full and a new page is needed, the new page must replace one of the resident pages. The operating system monitors the page transfer activity and dynamically adjusts the page allocation to various programs. Suggest a suitable strategy that the operating system can use to minimize the overall rate of page transfers.

5.23 In a computer with a virtual-memory system, the execution of an instruction may be interrupted by a page fault. What state information has to be saved so that this instruction can be resumed later? Note that bringing a new page into the main memory involves a DMA transfer, which requires execution of other instructions. Is it simpler to abandon the interrupted instruction and completely reexecute it later? Can this be done?

5.24 When a program generates a reference to a page that does not reside in the physical main memory, execution of the program is suspended until the requested page is loaded into the main memory. What difficulties might arise when an instruction in one page has an operand in a different page? What capabilities must the processor have to handle this situation?

5.25 A disk unit has 24 recording surfaces. It has a total of 14,000 cylinders. There is an average of 400 sectors per track. Each sector contains 512 bytes of data.

(*a*) What is the maximum number of bytes that can be stored in this unit?

(*b*) What is the data transfer rate in bytes per second at a rotational speed of 7200 rpm?

(*c*) Using a 32-bit word, suggest a suitable scheme for specifying the disk address, assuming that there are 512 bytes per sector.

5.26 The seek time plus rotational delay in accessing a particular data block on a disk is usually much longer than the data flow period for most disk transfers. Consider a long sequence of accesses to the 3.5-inch disk given as an example in Section 5.9.1, for either Read or Write operations in which the average block being accessed is 8K bytes long.

(*a*) Assuming that the blocks are randomly located on the disk, estimate the average percentage of the total time occupied by seek operations and rotational delays.

(*b*) Repeat part (*a*) for the situation in which the disk accesses have been arranged so that in 90 percent of the cases, the next access will be to a data block on the same cylinder.

5.27 The average seek time and rotational delay in a disk system are 6 ms and 3 ms, respectively. The rate of data transfer to or from the disk is 30 Mbytes/sec and all disk accesses are for 8 Kbytes of data. Disk DMA controllers, the processor, and the main memory are all attached to a single bus. The bus data width is 32 bits, and a bus transfer to or from the main memory takes 10 nanoseconds.

(*a*) What is the maximum number of disk units that can be simultaneously transferring data to or from the main memory?

(*b*) What percentage of main memory cycles are stolen by a disk unit, on average, over a long period of time during which a sequence of independent 8K-byte transfers takes place?

5.28 Given that magnetic disks are used as the secondary storage for program and data files in a virtual-memory system, which disk parameter(s) should influence the choice of page size?

5.29 A tape drive has the following parameters:

Bit density	2000 bits/cm
Tape speed	800 cm/s
Time to reverse direction of motion	225 ms
Minimum time spent at an interrecord gap	3 ms
Average record length	4000 characters

Estimate the percentage gain in time resulting from the ability to read records in both the forward and backward directions. Assume that records are accessed at random and that on average, the distance between two records accessed in sequence is four records.

REFERENCES

1. T.C. Mowry, "Tolerating Latency through Software-Controlled Data Prefetching," *Tech. Report CSL-TR-94-628*, Stanford University, Calif., 1994.
2. J.L. Baer and T.F. Chen, "An Effective On-Chip Preloading Scheme to Reduce Data Access Penalty," *Proceedings of Supercomputing '91*, 1991, pp. 176–186.
3. J.W.C. Fu and J.H. Patel, "Stride Directed Prefetching in Scalar Processors," *Proceedings of the 24th International Symposium on Microarchitecture*, 1992, pp. 102–110.
4. D. Kroft, "Lockup-Free Instruction Fetch/Prefetch Cache Organization," *Proceedings of the 8th Annual International Symposium on Computer Architecture*, 1981, pp. 81–85.
5. D.A. Patterson, G.A. Gibson, and R.H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1988, pp. 109–166.
6. P.M. Chen, E.K. Lee, G.A. Gibson, R.H. Katz, and D.A. Patterson, "RAID: High-Performance, Reliable Secondary Storage," *ACM Computing Surveys*, vol. 26, no. 2, June 1994, pp. 145–185.
7. D.A. Patterson and J.L. Hennessy, *Computer Architecture — A Quantitative Approach*, 2nd ed., Morgan Kaufmann, San Francisco, CA, 1996.
8. A.S. Tannenbaum, *Structured Computer Organization*, 4th ed. Prentice Hall, Upper Saddle River, NJ, 1999.
9. "RAID Technology White Paper," Dell Computer Corporation, 1999.
10. A. Clements, *The Principles of Computer Hardware*, 3rd ed., Oxford University Press, 2000.