Unit 3

Characteristics of Memory Systems

The complex subject of computer memory is made more manageable if we classify memory systems according to their key characteristics.

The term **location** refer to whether memory is internal and external to the computer. Internal memory is often equated with main memory. But there are other forms of internal memory. The processor requires its own local memory, in the form of registers Cache is another form of internal memory. External memory consists of peripheral storage devices, such as disk and tape that are accessible to the processor via I/O controllers.

Another characteristic of memory is **capacity**. For internal memory, this is typically expressed in terms of bytes (1 byte 8 bits) or words. Common word lengths are 8, 16, and 32 bits. External memory capacity is typically expressed in terms of bytes.

A related concept is the **unit of transfer**. For internal memory, the unit of transfer is equal to the number of electrical lines into and out of the memory module. This may be equal to the word length, but is often larger, such as 64, 128, or 256 bits. To clarify this point, consider three related concepts for internal memory:

- **Word:** The "natural" unit of organization of memory. The size of the word is typically equal to the number of bits used to represent an integer and to the instruction length.
- **Addressable units:** In some systems, the addressable unit is the word. However, many systems allow addressing at the byte level. The relationship between the length in bits A of an address and the number N of addressable units is $2^A=N$.
- **Unit of transfer:** For main memory, this is the number of bits read out of or written into memory at a time. The unit of transfer need not equal a word or an addressable unit.

Method of accessing units of data. These include the following:

- **Sequential access:** Memory is organized into units of data, called records. Access must be made in a specific linear sequence. Stored addressing information is used to separate records and assist in the retrieval process. A shared read– write mechanism is used, and this must be moved from its current location to the desired location, passing and rejecting each intermediate record. Thus, the time to access an arbitrary record is highly variable. Tape units, are sequential access.
- **Direct access:** As with sequential access, direct access involves a shared read– write mechanism. However, individual blocks or records have a unique address based on physical location. Access is accomplished by direct access to reach a general vicinity plus sequential searching, counting, or waiting to reach the final location. Again, access time is variable. Disk units, are direct access.
- **Random access:** Each addressable location in memory has a unique, physically wired-in addressing mechanism. The time to access a given location is independent of the sequence of prior accesses and is constant. Thus, any location

can be selected at random and directly addressed and accessed. Main memory and some cache systems are random access.

- **Associative:** This is a random access type of memory that enables one to make a comparison of desired bit locations within a word for a specified match, and to do this for all words simultaneously. Thus, a word is retrieved based on a portion of its contents rather than its address. As with ordinary random-access memory, each location has its own addressing mechanism, and retrieval time is constant independent of location or prior access patterns. Cache memories may employ associative access.

Three **performance** parameters are used:

- **Access time (latency):** For random-access memory, this is the time it takes to perform a read or write operation, that is, the time from the instant that an address is presented to the memory to the instant that data have been stored or made available for use. For non-random-access memory, access time is the time it takes to position the read–write mechanism at the desired location.
- **Memory cycle time:** This concept is primarily applied to random-access memory and consists of the access time plus any additional time required before a second access can commence. This additional time may be required for transients to die out on signal lines or to regenerate data if they are read destructively. Note that memory cycle time is concerned with the system bus, not the processor.
- **Transfer rate:** This is the rate at which data can be transferred into or out of a memory unit. For random-access memory, it is equal to 1/(cycle time).

A variety of **physical types** of memory have been employed. The most common today are semiconductor memory, magnetic surface memory, used for disk and tape, and optical and magneto-optical.

Physical characteristics of data storage are important. In a volatile memory, information decays naturally or is lost when electrical power is switched off. In a non-volatile memory, information once recorded remains without deterioration until deliberately changed; no electrical power is needed to retain information. Magnetic-surface memories are nonvolatile. Semiconductor memory may be either volatile or non-volatile. Non Erasable memory cannot be altered, except by destroying the storage unit. Semiconductor memory of this type is known as *read-only memory* (ROM). Of necessity, a practical non erasable memory must also be non-volatile.

Memory Hierarchy

In computer architecture the memory hierarchy is a concept used to discuss performance issues in computer architectural design, algorithm predictions, and lower level programming constructs involving locality of reference. The memory hierarchy in computer storage separates each of its levels based on response time. Since response

time, complexity, and capacity are related, the levels may also be distinguished by their performance and controlling technologies.

Designing for high performance requires considering the restrictions of the memory hierarchy, i.e. the size, capabilities and cost of each component. Each of the various components can be viewed as part of a hierarchy of memories ($m_1, m_2, ..., m_n$) in which each member m_i is typically smaller, faster and costly than the next highest member m_{i+1} of the hierarchy. To limit waiting by higher levels, a lower level will respond by filling a buffer and then signalling to activate the transfer

There is a trade-off among the three key characteristics of memory: namely, capacity, access time, and cost. A variety of technologies are used to implement memory systems, and across this spectrum of technologies, the following relationships hold:

- Faster access time, greater cost per bit
- Greater capacity, smaller cost per bit
- Greater capacity, slower access time

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 short 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.**

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

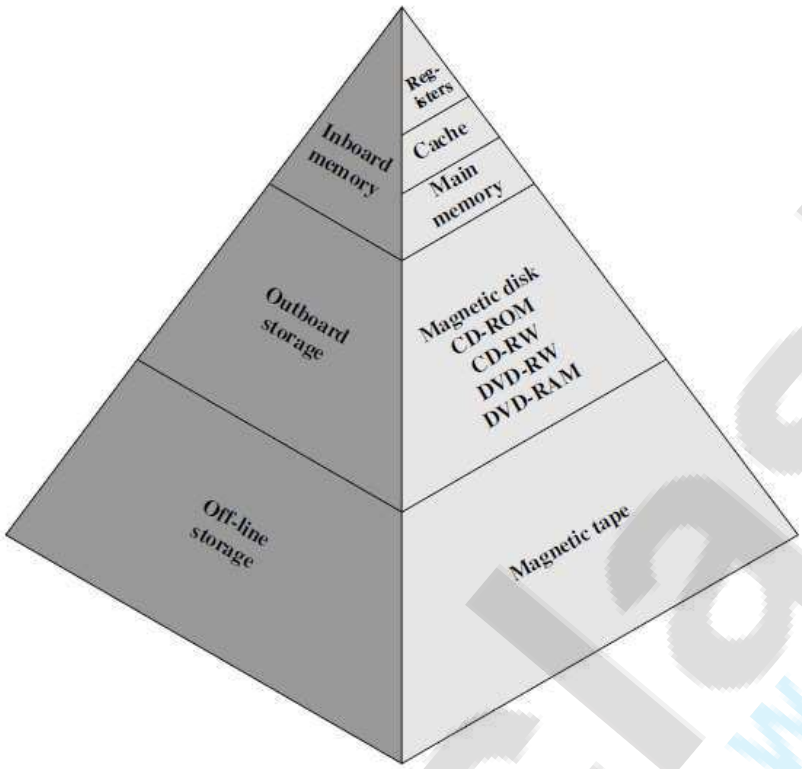


Figure 4.1 The Memory Hierarchy

The basis for the validity of condition(Decreasing frequency of access of the memory by the processor) is a principle known as locality of reference. During the course of execution of a program, memory references by the processor, for both instructions and data, tend to cluster. Programs typically contain a number of iterative loops and subroutines. Once a loop or subroutine is entered, there are repeated references to a small set of instructions. Similarly, operations on tables and arrays involve access to a clustered set of data words. Over a long period of time, the clusters in use change, but over a short period of time, the processor is primarily working with fixed clusters of memory references. Accordingly, it is possible to organize data across the hierarchy such that the percentage of accesses to each successively lower level is substantially less than that of the level above.

Internal Memory:

All semiconductor memory cells share certain properties:

- They exhibit two stable (or semistable) states, which can be used to represent binary 1 and 0.
- They are capable of being written into (at least once), to set the state.
- They are capable of being read to sense the state.

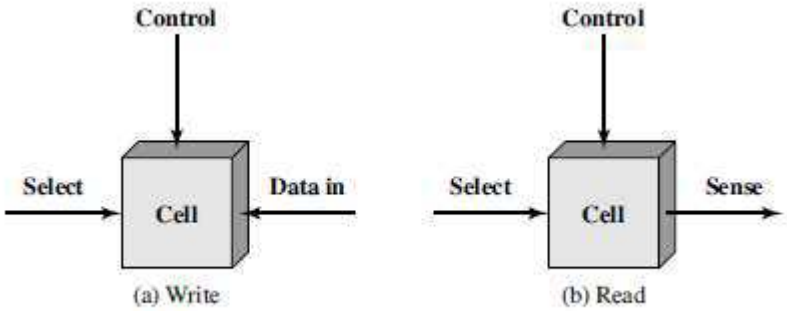


Figure 5.1 Memory Cell Operation

The most common is referred to as random-access memory (RAM). This is, of course, a misuse of the term, because all of the types listed in the table are random access. One distinguishing characteristic of RAM is that it is possible both to read data from the memory and to write new data into the memory easily and rapidly. Both the reading and writing are accomplished through the use of electrical signals. The other distinguishing characteristic of RAM is that it is volatile. A RAM must be provided with a constant power supply. If the power is interrupted, then the data are lost.Thus, RAM can be used only as temporary storage.The two traditional forms of RAM used in computers are DRAM and SRAM.

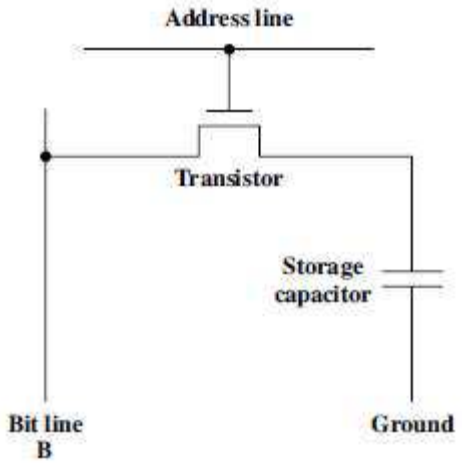
Table 5.1 Semiconductor Memory Types

Memory Type	Category	Erasure	Write Mechanism	Volatility
Random-access memory (RAM)	Read-write memory	Electrically, byte-level	Electrically	Volatile
Read-only memory (ROM)	Read-only memory	Not possible	Masks	Nonvolatile
Programmable ROM (PROM)				
Erasable PROM (EPROM)	Read-mostly memory	UV light, chip-level	Electrically	
Electrically Erasable PROM (EEPROM)		Electrically, byte-level		
Flash memory		Electrically, block-level		

RAM technology is divided into two technologies: dynamic and static.

Dynamic RAM

A dynamic RAM (DRAM) is made with cells that store data as charge on capacitors.The presence or absence of charge in a capacitor is interpreted as a binary 1 or 0. Because capacitors have a natural tendency to discharge, dynamic RAMs require periodic charge refreshing to maintain data storage. The term dynamic refers to this tendency of the stored charge to leak away, even with power continuously applied.



(a) Dynamic RAM (DRAM) cell

Figure 5.2a is a typical DRAM structure for an individual cell that stores 1 bit. The address line is activated when the bit value from this cell is to be read or written. The transistor acts as a switch that is closed (allowing current to flow) if a voltage is applied to the address line and open (no current flows) if no voltage is present on the address line.

For the write operation, a voltage signal is applied to the bit line; a high voltage represents 1, and a low voltage represents 0. A signal is then applied to the address line, allowing a charge to be transferred to the capacitor.

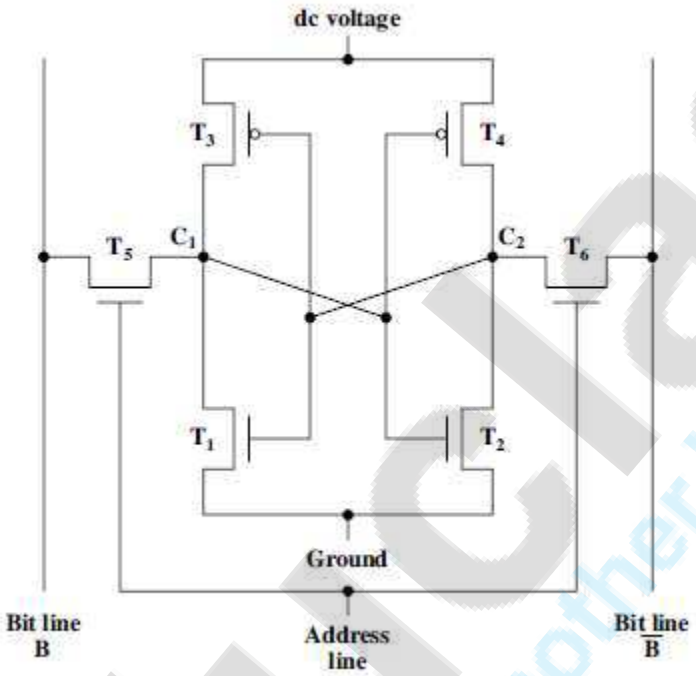
For the read operation, when the address line is selected, the transistor turns on and the charge stored on the capacitor is fed out onto a bit line and to a sense amplifier. The sense amplifier compares the capacitor voltage to a reference value and determines if the cell contains a logic 1 or a logic 0. The readout from the cell discharges the capacitor, which must be restored to complete the operation.

Static RAM

In contrast, a static RAM (SRAM) is a digital device that uses the same logic elements used in the processor. In a SRAM, binary values are stored using traditional flip-flop logic-gate configurations. A static RAM will hold its data as long as power is supplied to it.

Figure b is a typical SRAM structure for an individual cell. Four transistors (T1,T2,T3,T4) are cross connected in an arrangement that produces a stable logic state. In logic state 1, point C1 is high and point C2 is low; in this state,T1 and T4 are off and T2 and T3 are on.1 In logic state 0, point C1 is low and point C2 is high; in this state,T1 and T4 are on and T2 and T3 are off. Both states are stable as long as the direct current (dc) voltage is applied. Unlike the DRAM, no refresh is needed to retain data.

As in the DRAM, the SRAM address line is used to open or close a switch. The address line controls two transistors (T5 and T6).When a signal is applied to this line, the two transistors are switched on, allowing a read or write operation. For a write operation, the desired bit value is applied to line B, while its complement is applied to line . This forces the four transistors (T1, T2, T3, T4) into the proper state. For a read operation, the bit value is read from line B.



(b) Static RAM (SRAM) cell

DRAM	SRAM
Dynamic random-access memory is a type of random-access memory that stores each bit of data in a separate capacitor within an integrated circuit.	Static random-access memory is a type of semiconductor memory that uses bistable latching circuitry to store each bit.
Dynamic memory cell is simpler and smaller than a static memory cell.	Static memory cell is complex as compared to dynamic memory cell
DRAM is more dense (smaller cells more cells per unit area) and less expensive than a corresponding SRAM.	SRAM is sparse (large cells less cells per unit area) and more expensive than a corresponding DRAM.
DRAM requires the supporting refresh circuitry.	SRAM do not requires Supporting refresh circuit.

DRAMs tend to be favored for large memory requirements.	SRAMs tend to be favored for small memory requirements.
DRAMs are generally somewhat slower than SRAMs.	SRAMs are generally somewhat faster than DRAMs.
SRAM is used for cache memory (both on and off chip)	DRAM is used for main memory.

Interleaved Memory

Main memory is composed of a collection of DRAM memory chips. A number of chips can be grouped together to form a memory bank. It is possible to organize the memory banks in a way known as interleaved memory. Each bank is independently able to service a memory read or write request, so that a system with K banks can service K requests simultaneously, increasing memory read or write rates by a factor of K. If consecutive words of memory are stored in different banks, then the transfer of a block of memory is speeded up.

Associative Memory

A memory whose storage locations are identified by their contents, or by a part of their contents, rather than by their names or positions. This is a random access type of memory that enables one to make a comparison of desired bit locations within a word for a specified match, and to do this for all words simultaneously. Thus, a word is retrieved based on a portion of its contents rather than its address. As with ordinary random-access memory, each location has its own addressing mechanism, and retrieval time is constant independent of location or prior access patterns. Cache memories may employ associative access.

Cache Memory

A relatively small fast memory interposed between a larger, slower memory and the logic that accesses the larger memory.The cache holds recently accessed data, and is designed to speed up subsequent access to the same data.

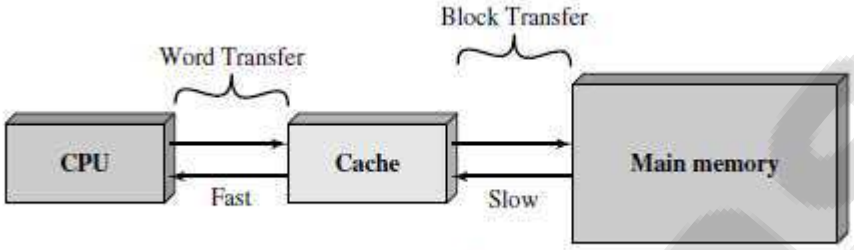
A special buffer storage, smaller and faster than main storage, that is used to hold a copy of instructions and data in main storage that are likely to be needed next by the processor and that have been obtained automatically from main storage.

A portion of main memory can be used as a buffer to hold data temporarily that is to be read out to disk. Such a technique, sometimes referred to as a disk cache improves performance in two ways:

- Disk writes are clustered. Instead of many small transfers of data, we have a few large transfers of data. This improves disk performance and minimizes processor involvement.

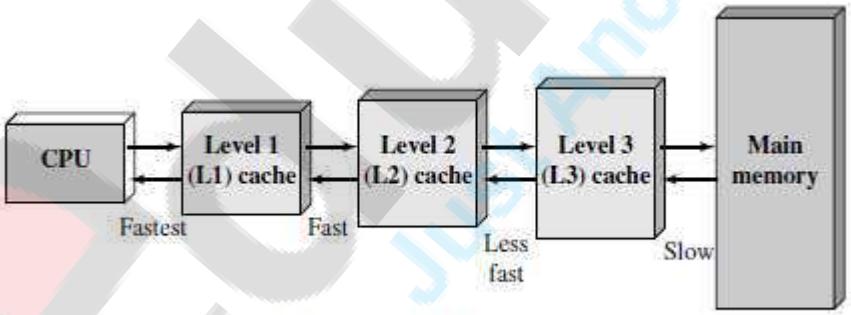
- Some data destined for write-out may be referenced by a program before the next dump to disk. In that case, the data are retrieved rapidly from the software cache rather than slowly from the disk.

Cache memory is intended to give memory speed approaching that of the fastest memories available, and at the same time provide a large memory size at the price of less expensive types of semiconductor memories. The concept is illustrated in Figure.



(a) Single cache

There is a relatively large and slow main memory together with a smaller, faster cache memory. The cache contains a copy of portions of main memory. When the processor attempts to read a word of memory, a check is made to determine if the word is in the cache. If so, the word is delivered to the processor. If not, a block of main memory, consisting of some fixed number of words, is read into the cache and then the word is delivered to the processor. Because of the phenomenon of **locality of reference**, when a block of data is fetched into the cache to satisfy a single memory reference, it is likely that there will be future references to that same memory location or to other words in the block.



(b) Three-level cache organization

Figure (b) depicts the use of multiple levels of cache. The L2 cache is slower and typically larger than the L1 cache, and the L3 cache is slower and typically larger than the L2 cache.

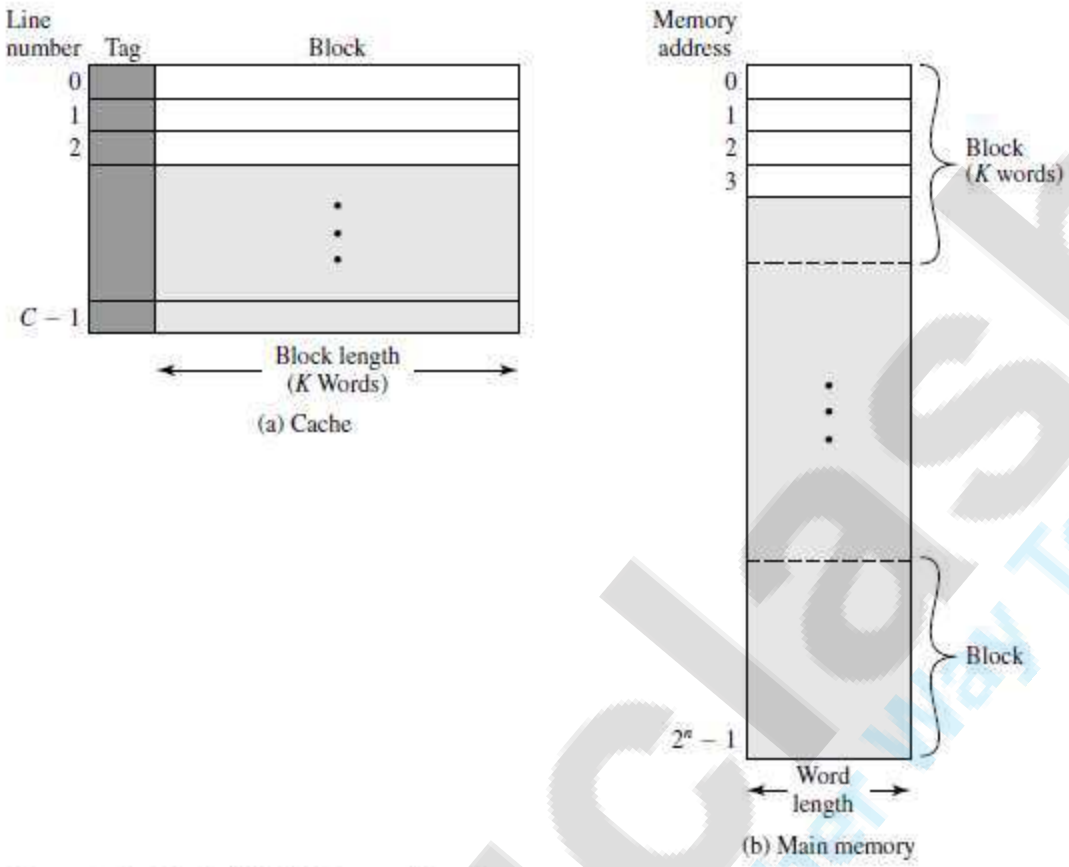


Figure 4.4 Cache/Main Memory Structure

Figure 4.4 depicts the structure of a cache/main-memory system. Main memory consists of up to 2^n addressable words, with each word having a unique n-bit address. For mapping purposes, this memory is considered to consist of a number of fixed length blocks of K words each.That is, there are $M=2^n/K$ blocks in main memory. The cache consists of m blocks, called lines. Each line contains K words, plus a tag of a few bits. Each line also includes control bits (not shown), such as a bit to indicate whether the line has been modified since being loaded into the cache.The length of a line, not including tag and control bits, is the line size. The line size may be as small as 32 bits, with each "word" being a single byte; in this case the line size is 4 bytes. The number of lines is considerably less than the number of main memory blocks($m<<M$). At any time, some subset of the blocks of memory resides in lines in the cache. If a word in a block of memory is read, that block is transferred to one of the lines of the cache. Because there are more blocks than lines, an individual line cannot be uniquely and permanently dedicated to a particular block.Thus, each line includes a tag that identifies which particular block is currently being stored. The tag is usually a portion of the main memory address.

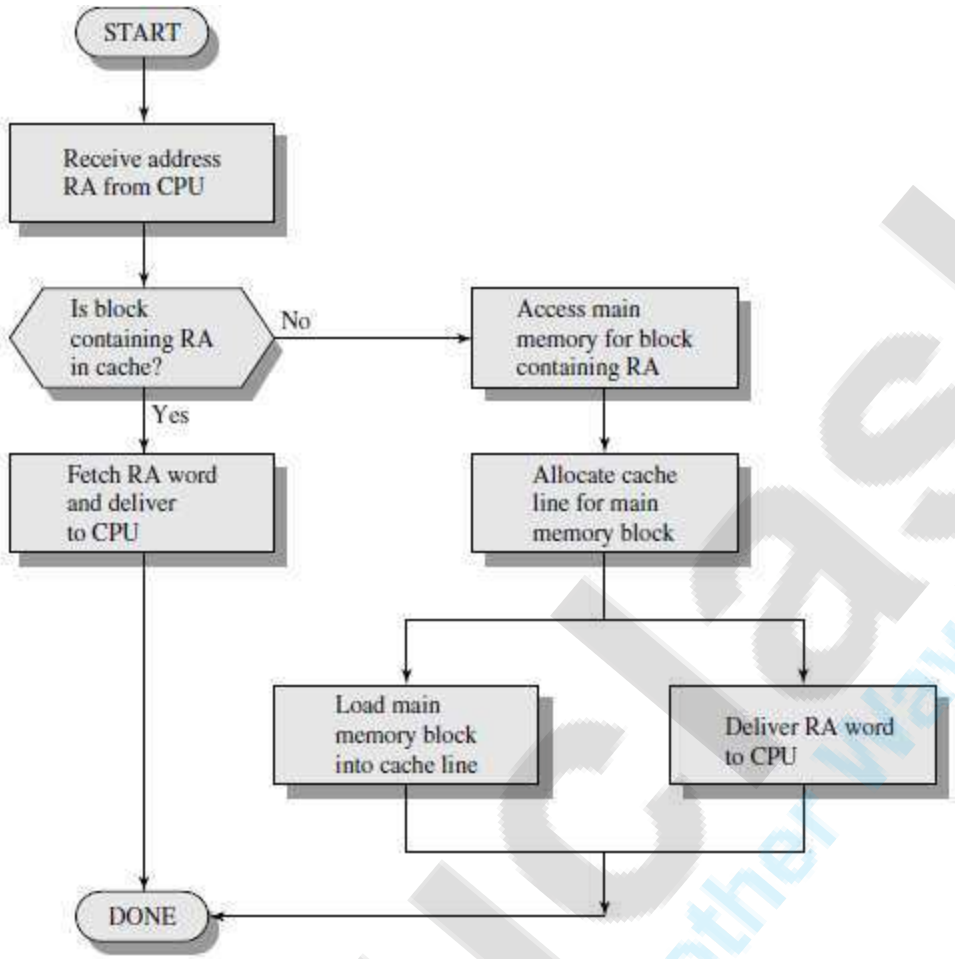


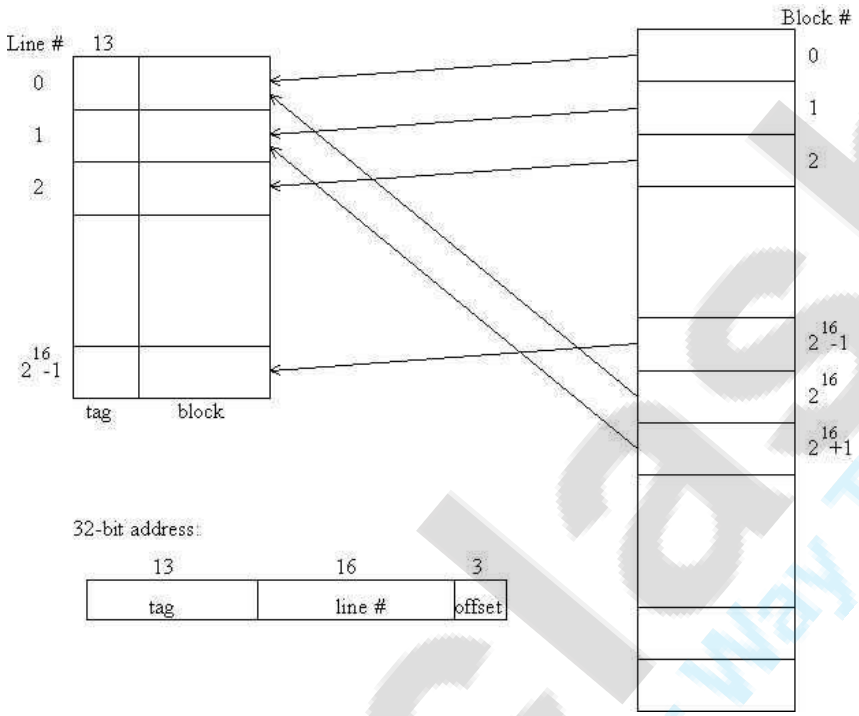
Figure 4.5 Cache Read Operation

Memory Mapping Techniques

Following are the Memory Mapping Techniques

- Direct Mapping
- Associative Mapping
- Set-associative Mapping

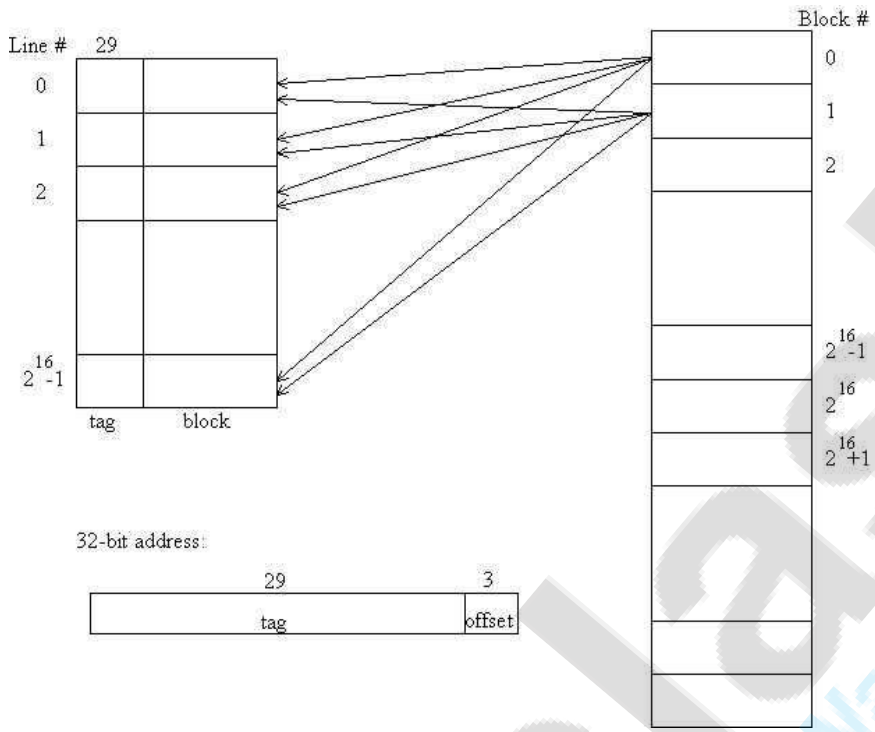
Direct Mapping



When CPU generates a memory request,

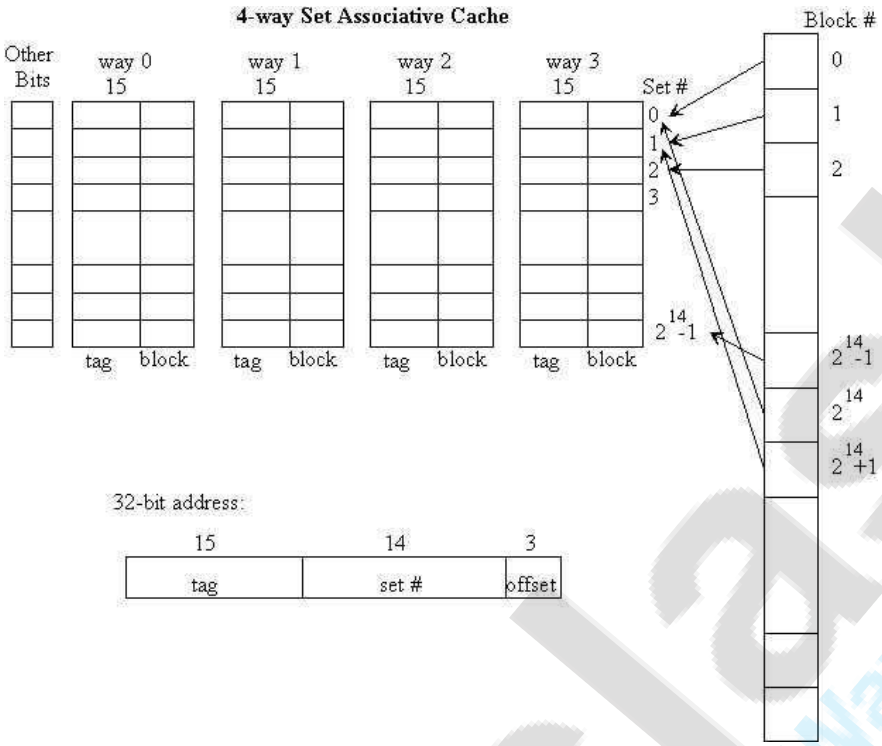
- Index field is used for the address to access the cache.
- Tag field of the CPU address is compared with the tag in the word read from the cache.
- If two tags match, there is a hit and the desired data word is in cache.
- If there is no match, there is a miss and the required word is read from main memory.
- It is then stored in the cache together with the new tag , replacing the previous value.
- The disadvantage is that, the hit ratio will drop if two or more words whose address have the same index but different tags are accessed repeatedly.

Associative Mapping



- Fastest and flexible cache organization.
- Stores both the address and data of the memory word.
- Permits any location in cache to store any word from main memory.
- If no match occurs, the main memory is accessed for the word.
- The address-data pair is then transferred to the associate cache memory.
- If cache is full, an address-data pair must be displaced to make room for a pair that is needed and not present in the cache.
- Drawback: Address occupies more memory.

Set-associative Mapping



- Cache is divided into a number of sets.
- Each word of cache can store two or more words of memory under the same index address.
- When a miss occurs and the set is full, it is necessary to replace one of the tag-data items with a new value.

Replacement Algorithms

Once the cache has been filled, when a new block is brought into the cache, one of the existing blocks must be replaced. For direct mapping, there is only one possible line for any particular block, and no choice is possible. For the associative and set associative techniques, a replacement algorithm is needed. To achieve high speed, such an algorithm must be implemented in hardware. A number of algorithms have been tried. We mention four of the most common.

- Probably the most effective is **least recently used (LRU)**: Replace that block in the set that has been in the cache longest with no reference to it. For two-way set associative, this is easily implemented. Each line includes a USE bit.When a line is referenced, its USE bit is set to 1 and the USE bit of the other line in that set is set to 0.When a block is to be read into the set, the line whose USE bit is 0 is used. Because we are assuming that more recently used memory locations are more likely to be referenced, LRU should give the best hit ratio. LRU is also relatively easy to implement for a fully associative cache. The cache mechanism maintains a separate list of indexes to all the lines in the cache.When a line is referenced, it moves to the front of the list. For replacement, the line at the back of the list is

used. Because of its simplicity of implementation, LRU is the most popular replacement algorithm.

- Another possibility is **first-in-first-out (FIFO)**: Replace that block in the set that has been in the cache longest. FIFO is easily implemented as a round-robin or circular buffer technique.
- Still another possibility is **least frequently used (LFU)**: Replace that block in the set that has experienced the fewest references. LFU could be implemented by associating a counter with each line.
- A technique not based on usage (i.e., not LRU, LFU, FIFO, or some variant) is to pick a line at random from among the candidate lines. Simulation studies have shown that **random replacement** provides only slightly inferior performance to an algorithm based on usage.
- **Optimal Replacement Algorithm-** The theoretically optimal page replacement algorithm is an algorithm that works as follows: when a page needs to be swapped in, the operating system swaps out the page whose next use will occur farthest in the future.

Cache Performance

Cache Coherence.

The use of caches introduces some new design considerations. Because each local cache contains an image of a portion of memory, if a word is altered in one cache, it could conceivably invalidate a word in another cache. To prevent this, the other processors must be alerted that an update has taken place. This problem is known as the cache coherence problem and is typically addressed in hardware rather than by the operating system.

Explain cache coherence strategies in single and multiprocessor systems.

Virtual Memory

With the use of paging, truly effective multiprogramming systems came into being. Furthermore, the simple tactic of breaking a process up into pages led to the development of another important concept: virtual memory. To understand virtual memory. That refinement is demand paging, which simply means that each page of a process is brought in only when it is needed, that is, on demand.

Consider a large process, consisting of a long program plus a number of arrays of data. Over any short period of time, execution may be confined to a small section of the program (e.g., a subroutine), and perhaps only one or two arrays of data are being used.This is the principle of locality. It would clearly be wasteful to load in dozens of pages for that process when only a few pages will be used before the program is suspended.We can make better use of memory by loading in just a few pages.Then, if the program

branches to an instruction on a page not in main memory, or if the program references data on a page not in memory, a **page fault** is triggered.This tells the OS to bring in the desired page. Thus, at any one time, only a few pages of any given process are in memory, and therefore more processes can be maintained in memory. Furthermore, time is saved because unused pages are not swapped in and out of memory. However, the OS must be clever about how it manages this scheme. When it brings one page in, it must throw another page out; this is known as **page replacement**. If it throws out a page just before it is about to be used, then it will just have to go get that page again almost immediately. Too much of this leads to a condition known as **thrashing**: *the processor spends most of its time swapping pages rather than executing instructions*. In essence, the OS tries to guess, based on recent history, which pages are least likely to be used in the near future.

With demand paging, it is not necessary to load an entire process into main memory. This fact has a remarkable consequence: *It is possible for a process to be larger than all of main memory*. One of the most fundamental restrictions in programming has been lifted. Without demand paging, a programmer must be acutely aware of how much memory is available. If the program being written is too large, the programmer must devise ways to structure the program into pieces that can be loaded one at a time.With demand paging, that job is left to the OS and the hardware. As far as the programmer is concerned, he or she is dealing with a huge memory, the size associated with disk storage. Because a process executes only in main memory, that memory is referred to as real memory. But a programmer or user perceives a much larger memory—that which is allocated on the disk.This latter is therefore referred to as virtual memory. **Virtual memory allows for very effective multiprogramming and relieves the user of the unnecessarily tight constraints of main memory.**

External Memory :

Magnetic Discs

A magnetic disk is a storage device that uses a magnetization process to write, rewrite and access data. It is covered with a magnetic coating and stores data in the form of tracks, spots and sectors. Hard disks, zip disks and floppy disks are common examples of magnetic disks.

A disk is a circular platter constructed of nonmagnetic material, called the substrate, coated with a magnetizable material. Traditionally, the substrate has been an aluminum or aluminum alloy material. More recently, glass substrates have been introduced. The glass substrate has a number of benefits, including the following:

- Improvement in the uniformity of the magnetic film surface to increase disk reliability
- A significant reduction in overall surface defects to help reduce read-write errors
- Ability to support lower fly heights (described subsequently)

- Better stiffness to reduce disk dynamics
- Greater ability to withstand shock and damage

Optical Memory

Table 6.5 Optical Disk Products

CD Compact Disk. A nonerasable disk that stores digitized audio information. The standard system uses 12-cm disks and can record more than 60 minutes of uninterrupted playing time.
CD-ROM Compact Disk Read-Only Memory. A nonerasable disk used for storing computer data. The standard system uses 12-cm disks and can hold more than 650 Mbytes.
CD-R CD Recordable. Similar to a CD-ROM. The user can write to the disk only once.
CD-RW CD Rewritable. Similar to a CD-ROM. The user can erase and rewrite to the disk multiple times.
DVD Digital Versatile Disk. A technology for producing digitized, compressed representation of video information, as well as large volumes of other digital data. Both 8 and 12 cm diameters are used, with a double-sided capacity of up to 17 Gbytes. The basic DVD is read-only (DVD-ROM).
DVD-R DVD Recordable. Similar to a DVD-ROM. The user can write to the disk only once. Only one-sided disks can be used.
DVD-RW DVD Rewritable. Similar to a DVD-ROM. The user can erase and rewrite to the disk multiple times. Only one-sided disks can be used.
Blu-Ray DVD High definition video disk. Provides considerably greater data storage density than DVD, using a 405-nm (blue-violet) laser. A single layer on a single side can store 25 Gbytes.

RAID

RAID (originally redundant array of inexpensive disks, now commonly array of independent disks) is a data storage virtualization technology that combines multiple physical disk drive components into a single logical unit for the purposes of data redundancy, performance improvement, or both

In the case of disk storage, this leads to the development of arrays of disks that operate independently and in parallel.With multiple disks, separate I/O requests can be handled

in parallel, as long as the data required reside on separate disks. Further, a single I/O request can be executed in parallel if the block of data to be accessed is distributed across multiple disks.

Data is distributed across the drives in one of several ways, referred to as RAID levels, depending on the required level of redundancy and performance. The different schemas, or data distribution layouts, are named by the word RAID followed by a number, for example RAID 0 or RAID 1. Each schema, or RAID level, provides a different balance among the key goals: reliability, availability, performance, and capacity. RAID levels greater than RAID 0 provide protection against unrecoverable sector read errors, as well as against failures of whole physical drives.

The RAID scheme consists of seven levels,3 zero through six. These levels do not imply a hierarchical relationship but designate different design architectures that share three common characteristics:

1. RAID is a set of physical disk drives viewed by the operating system as a single logical drive.
2. Data are distributed across the physical drives of an array in a scheme known as striping, described subsequently.
3. Redundant disk capacity is used to store parity information, which guarantees data recoverability in case of a disk failure.

The details of the second and third characteristics differ for the different RAID levels. RAID 0 and RAID 1 do not support the third characteristic.

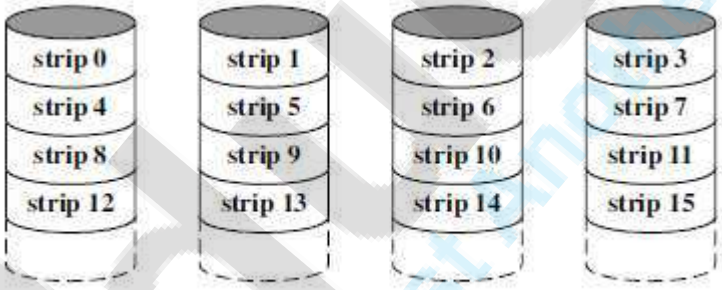
Raid Levels

Table 6.3 RAID Levels

Category	Level	Description	Disks Required	Data Availability	Large I/O Data Transfer Capacity	Small I/O Request Rate
Striping	0	Nonredundant	N	Lower than single disk	Very high	Very high for both read and write
Mirroring	1	Mirrored	$2N$	Higher than RAID 2, 3, 4, or 5; lower than RAID 6	Higher than single disk for read; similar to single disk for write	Up to twice that of a single disk for read; similar to single disk for write
Parallel access	2	Redundant via Hamming code	$N + m$	Much higher than single disk; comparable to RAID 3, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
	3	Bit-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
Independent access	4	Block-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 5	Similar to RAID 0 for read; significantly lower than single disk for write	Similar to RAID 0 for read; significantly lower than single disk for write
	5	Block-interleaved distributed parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 4	Similar to RAID 0 for read; lower than single disk for write	Similar to RAID 0 for read; generally lower than single disk for write
	6	Block-interleaved dual distributed parity	$N + 2$	Highest of all listed alternatives	Similar to RAID 0 for read; lower than RAID 5 for write	Similar to RAID 0 for read; significantly lower than RAID 5 for write

N = number of data disks; m proportional to log N

RAID Level 0



(a) RAID 0 (Nonredundant)

RAID level 0 is not a true member of the RAID family because it does not include redundancy to improve performance. However, there are a few applications, such as some on supercomputers in which performance and capacity are primary concerns and low cost is more important than improved reliability.

For RAID 0, the user and system data are distributed across all of the disks in the array. This has a notable advantage over the use of a single large disk: If two different I/O requests are pending for two different blocks of data, then there is a good chance that the requested blocks are on different disks.Thus, the two requests can be issued in parallel, reducing the I/O queuing time.

But RAID 0, as with all of the RAID levels, goes further than simply distributing the data across a disk array:The data are striped across the available disks. This is best understood by considering Figure.

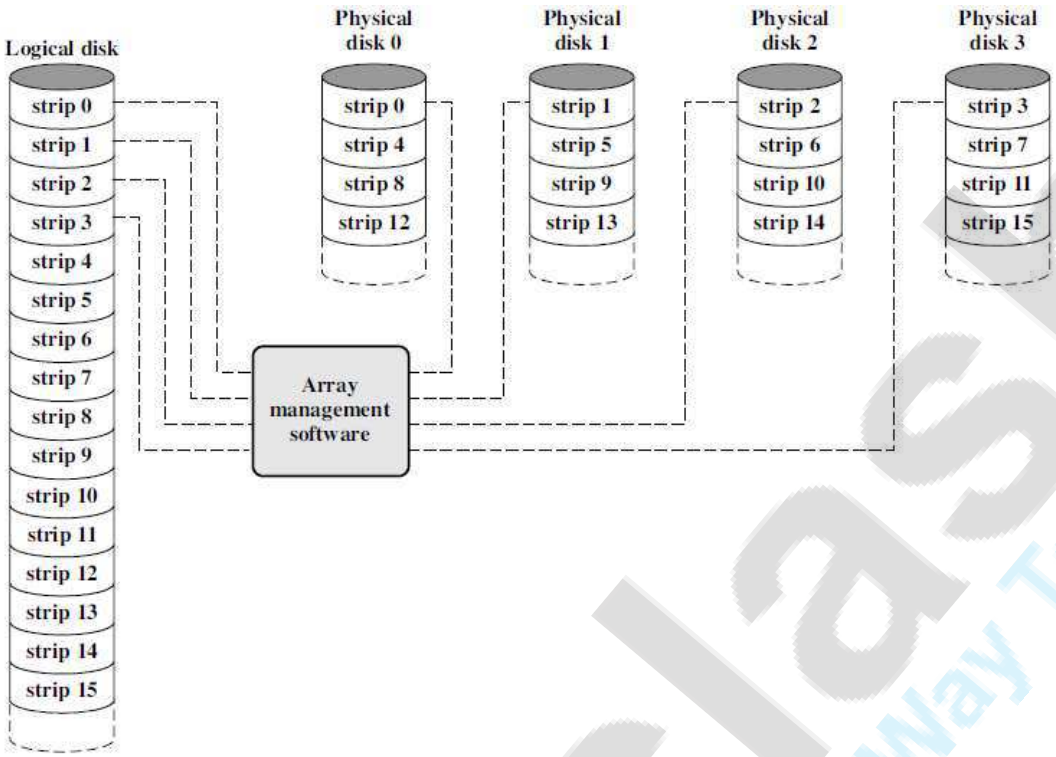
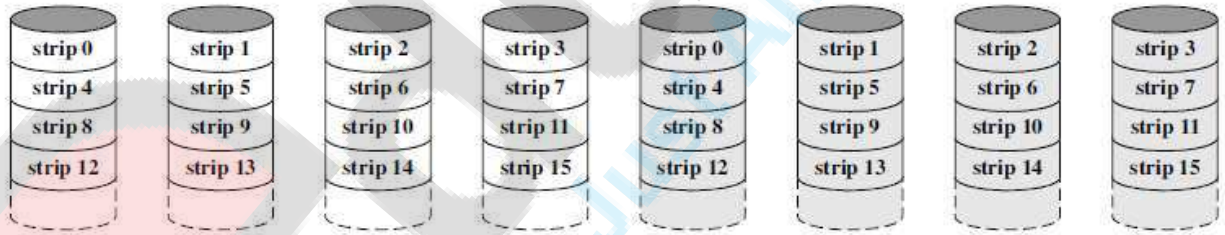


Figure 6.9 Data Mapping for a RAID Level 0 Array

All of the user and system data are viewed as being stored on a logical disk. A set of logically consecutive strips that maps exactly one strip to each array member is referred to as a stripe.

RAID Level 1



(b) RAID 1 (Mirrored)

RAID 1 differs from RAID levels 2 through 6 in the way in which redundancy is achieved. In these other RAID schemes, some form of parity calculation is used to introduce redundancy, whereas in RAID 1, redundancy is achieved by the simple expedient of duplicating all the data.As Figure 6.8b shows, data striping is used, as in RAID 0. But in this case, each logical strip is mapped to two separate physical disks so that every disk in the array has a mirror disk that contains the same data. RAID 1 can also be implemented without data striping, though this is less common. There are a number of positive aspects to the RAID 1 organization:

1. A read request can be serviced by either of the two disks that contains the requested data, whichever one involves the minimum seek time plus rotational latency.
2. A write request requires that both corresponding strips be updated, but this can be done in parallel.Thus, the write performance is dictated by the slower of the two writes (i.e., the one that involves the larger seek time plus rotational latency). However, there is no "write penalty" with RAID 1. RAID levels 2 through 6 involve the use of parity bits. Therefore, when a single strip is updated, the array management software must first compute and update the parity bits as well as updating the actual strip in question.
3. Recovery from a failure is simple.When a drive fails, the data may still be accessed from the second drive.

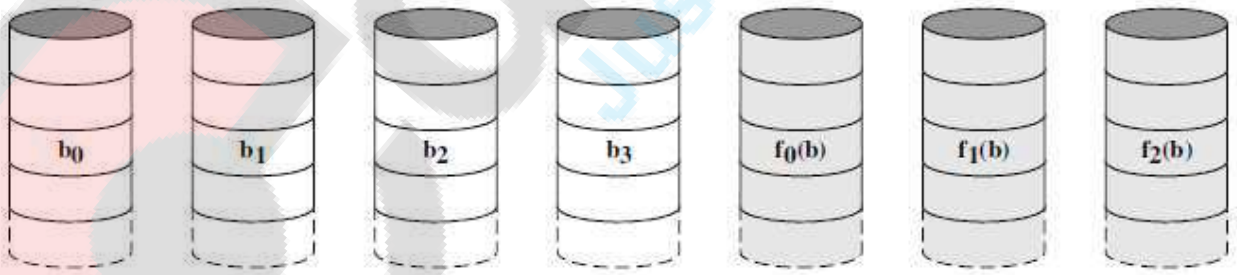
Disadvantage

- RAID 1 is the costly; it requires twice the disk space of the logical disk that it supports. Because of that, a RAID 1 configuration is likely to be limited to drives that store system software and data and other highly critical files.
- The main disadvantage is that the effective storage capacity is only half of the total drive capacity because all data get written twice.

Advantages

- RAID 1 offers excellent read speed and a write-speed that is comparable to that of a single drive.
- In case a drive fails, data do not have to be rebuild, they just have to be copied to the replacement drive.
- RAID 1 is a very simple technology

RAID Level 2



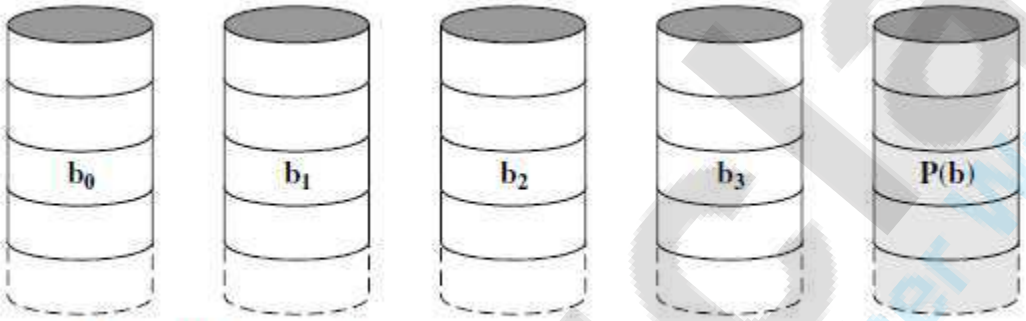
(c) RAID 2 (Redundancy through Hamming code)

RAID levels 2 and 3 make use of a parallel access technique. In a parallel access array, all member disks participate in the execution of every I/O request. Typically, the spindles of the individual drives are synchronized so that each disk head is in the same position on each disk at any given time.

In the case of RAID 2 and 3, the strips are very small, often as small as a single byte or word. With RAID 2, an error-correcting code is calculated across corresponding bits on

each data disk, and the bits of the code are stored in the corresponding bit positions on multiple parity disks. Typically, a Hamming code is used, which is able to correct single-bit errors and detect double-bit errors. Although RAID 2 requires fewer disks than RAID 1, it is still rather costly. The number of redundant disks is proportional to the log of the number of data disks. On a single read, all disks are simultaneously accessed. The requested data and the associated error-correcting code are delivered to the array controller. If there is a single-bit error, the controller can recognize and correct the error instantly, so that the read access time is not slowed. On a single write, all data disks and parity disks must be accessed for the write operation. RAID 2 would only be an effective choice in an environment in which many disk errors occur. Given the high reliability of individual disks and disk drives, RAID 2 is overkill and is not implemented.

RAID Level 3



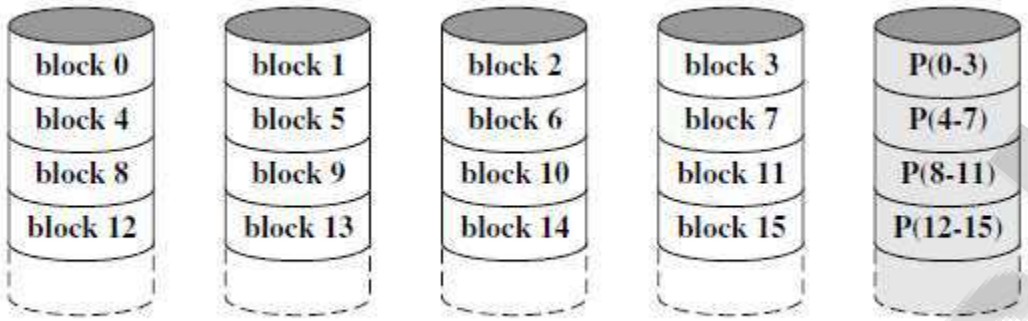
(d) RAID 3 (Bit-interleaved parity)

RAID 3 is organized in a similar fashion to RAID 2.The difference is that RAID 3 requires only a single redundant disk, no matter how large the disk array. RAID 3 employs parallel access, with data distributed in small strips. Instead of an error-correcting code, a simple parity bit is computed for the set of individual bits in the same position on all of the data disks.

REDUNDANCY In the event of a drive failure, the parity drive is accessed and data is reconstructed from the remaining devices. Once the failed drive is replaced, the missing data can be restored on the new drive and operation resumed.

PERFORMANCE Because data are striped in very small strips, RAID 3 can achieve very high data transfer rates.

RAID Level 4



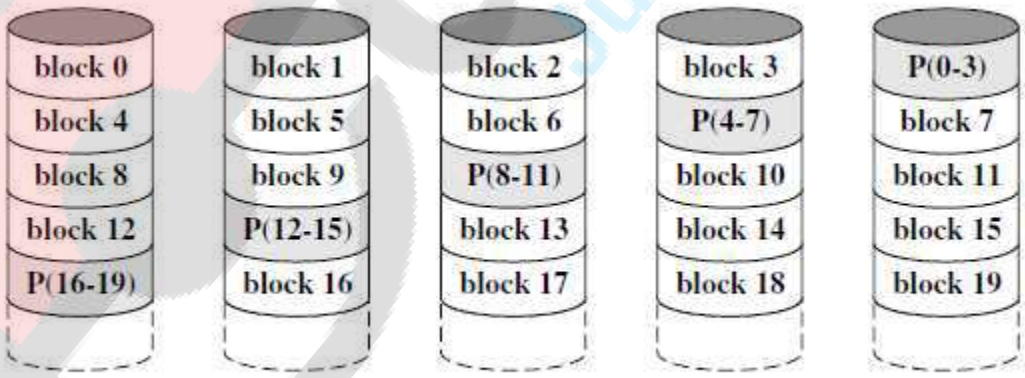
(e) RAID 4 (Block-level parity)

RAID levels 4 through 6 make use of an independent access technique. In an independent access array, each member disk operates independently, so that separate I/O requests can be satisfied in parallel. Because of this, independent access arrays are more suitable for applications that require high I/O request rates and are relatively less suited for applications that require high data transfer rates.

The strips are relatively large. With RAID 4, a bit-by-bit parity strip is calculated across corresponding strips on each data disk, and the parity bits are stored in the corresponding strip on the parity disk.

RAID 4 involves a write penalty when an I/O write request of small size is performed. Each time that a write occurs, the array management software must update not only the user data but also the corresponding parity bits.To calculate the new parity, the array management software must read the old user strip and the old parity strip. Then it can update these two strips with the new data and the newly calculated parity. Thus, each strip write involves two reads and two writes. every write operation must involve the parity disk, which therefore can become a bottleneck.

RAID Level 5

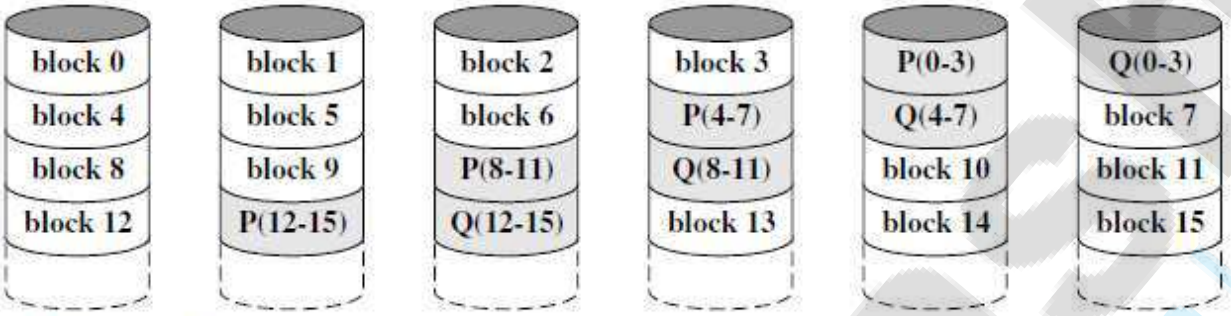


(f) RAID 5 (Block-level distributed parity)

RAID 5 is organized in a similar fashion to RAID 4. The strips are relatively large. The difference is that RAID 5 distributes the parity strips across all disks. A typical allocation

is a round-robin scheme, as illustrated in Figure. For an n-disk array, the parity strip is on a different disk for the first n stripes, and the pattern then repeats. The distribution of parity strips across all drives avoids the potential I/O bottleneck found in RAID 4.

RAID Level 6



(g) RAID 6 (Dual redundancy)

In the RAID 6 scheme, two different parity calculations are carried out and stored in separate blocks on different disks. Thus, a RAID 6 array whose user data require N disks consists of N 2 disks. Figure illustrates the scheme. P and Q are two different data check algorithms. One of the two is the exclusive-OR calculation used in RAID 4 and 5. But the other is an independent data check algorithm.This makes it possible to regenerate data even if two disks containing user data fail. The advantage of RAID 6 is that it provides extremely high data availability. Three disks would have to fail within the MTTR (mean time to repair) interval to cause data to be lost. On the other hand, RAID 6 incurs a substantial write penalty, because each write affects two parity blocks.

Table 6.4 RAID Comparison

Level	Advantages	Disadvantages	Applications
0	I/O performance is greatly improved by spreading the I/O load across many channels and drives No parity calculation overhead is involved Very simple design Easy to implement	The failure of just one drive will result in all data in an array being lost	Video production and editing Image Editing Pre-press applications Any application requiring high bandwidth
1	100% redundancy of data means no rebuild is necessary in case of a disk failure, just a copy to the replacement disk Under certain circumstances, RAID 1 can sustain multiple simultaneous drive failures Simplest RAID storage subsystem design	Highest disk overhead of all RAID types (100%)—inefficient	Accounting Payroll Financial Any application requiring very high availability
2	Extremely high data transfer rates possible The higher the data transfer rate required, the better the ratio of data disks to ECC disks Relatively simple controller design compared to RAID levels 3, 4 & 5	Very high ratio of ECC disks to data disks with smaller word sizes—inefficient Entry level cost very high—requires very high transfer rate requirement to justify	No commercial implementations exist/ not commercially viable
3	Very high read data transfer rate Very high write data transfer rate Disk failure has an insignificant impact on throughput Low ratio of ECC (parity) disks to data disks means high efficiency	Transaction rate equal to that of a single disk drive at best (if spindles are synchronized) Controller design is fairly complex	Video production and live streaming Image editing Video editing Prepress applications Any application requiring high throughput
4	Very high Read data transaction rate Low ratio of ECC (parity) disks to data disks means high efficiency	Quite complex controller design Worst write transaction rate and Write aggregate transfer rate Difficult and inefficient data rebuild in the event of disk failure	No commercial implementations exist/ not commercially viable

Level	Advantages	Disadvantages	Applications
5	Highest Read data transaction rate Low ratio of ECC (parity) disks to data disks means high efficiency Good aggregate transfer rate	Most complex controller design Difficult to rebuild in the event of a disk failure (as compared to RAID level 1)	File and application servers Database servers Web, e-mail, and news servers Intranet servers Most versatile RAID level
6	Provides for an extremely high data fault tolerance and can sustain multiple simultaneous drive failures	More complex controller design Controller overhead to compute parity addresses is extremely high	Perfect solution for mission critical applications