UNIT- V

MEMORY AND I/O

Memory concepts and Hierarchy - Memory Management - Cache Memories : Mapping and Replacement Techniques - Virtual Memory - DMA - I/O - Accessing I/O : Parallel and Serial Interface - Interrupt I/O - Interconnection Standards : USB, SATA.

Memory Concepts and Hierarchy.

Memory Concepts:

* Programs and the data they operate on are held in the memory of the computer.
* The memory is designed to store and retrieve data in word-length quantities.
  - word-length = number of bits stored or retrieved in one memory access.

*Size:

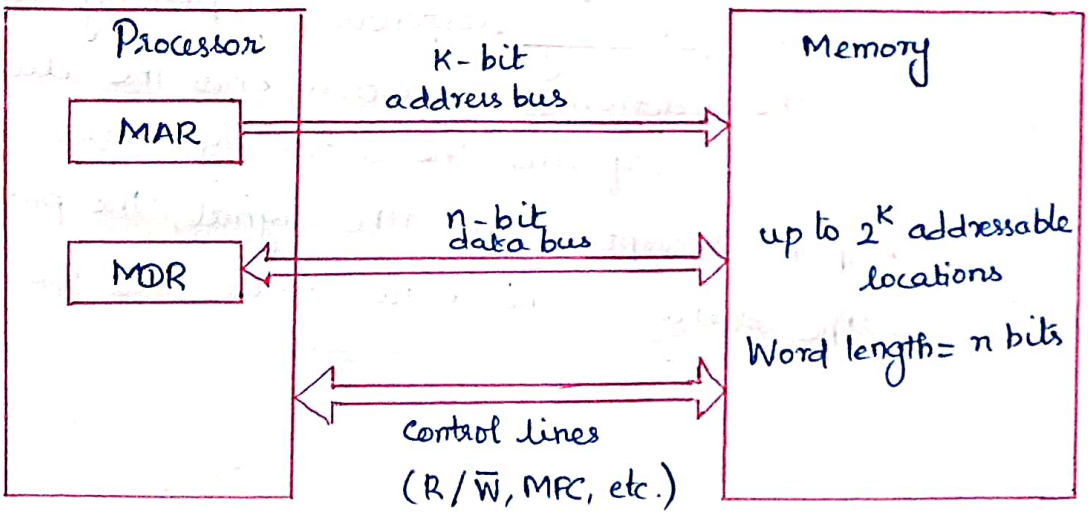
* The maximum size of the memory that can be used in any computer is determined by the addressing scheme.

Ex:

* a 16-bit computer that generates 16-bit addresses is capable of addressing upto $2^{16} = 64K$ memory locations.
* 32-bit addresses:
  $2^{32} = 4G$ (giga) memory locations
* 40-bit addresses:
  $2^{40} = 1T$ (tera) locations.

→ The number of locations represents the size of the address space of the computer.

Connection of the memory to the Processor



* View the memory unit as a black box.
- Data transfer between the memory and the processor takes place through the use of two processor registers
  - i) MAR (Memory Address Register)
  - ii) MDR (Memory Data Register)

MAR – K bits long, MDR – n bits long
→ memory unit upto 2^K addressable locations.

* During a memory cycle, n bits of data are transferred between the memory and the processor.
- Transfer takes place over the processor bus. which has
  - K address lines
  - n data lines
- Bus includes the control lines for coordinating data transfers
  - * Read / write ($R/\overline{W}$)
  - * Memory function Completed (MFC)

Two operations:

(i) Read:

* Data is retrieved from the memory.
- The processor reads data from the memory by
  - loading the address of the required memory location into the MAR register
  - setting the $R/\overline{W}$ line to 1.
- The memory responds by placing the data from the addressed location onto the data lines
  - confirms the action by asserting the MFC signal.

* Upon receipt of the MFC signal, the processor loads the data on the data lines into the MDR register.

(ii) Write:

* Data is stored into the memory

- The processor writes data into a memory location by
  - loading the address of the location into MAR
  and
  - loading the data into MDR.
  - setting the R/$\bar{W}$ line to 0.

* If read or write operations involve consecutive address locations in the main memory, then a "block transfer" operation can be performed in which the only address sent to the memory is the one that identifies the first location.

* clock:

* Memory accesses may be synchronized using a clock, or they may be controlled using special signals that control transfers on the bus.

* Speed:
  - The time that elapses between the initiation of an operation and the completion of that operation

* Memory Access Time:
  - The time between the Read and MFC signals.

* Memory cycle Time:
  - Minimum time delay required between the initiation of two successive memory operations
- cycle time is longer than the access time.

* RAM:

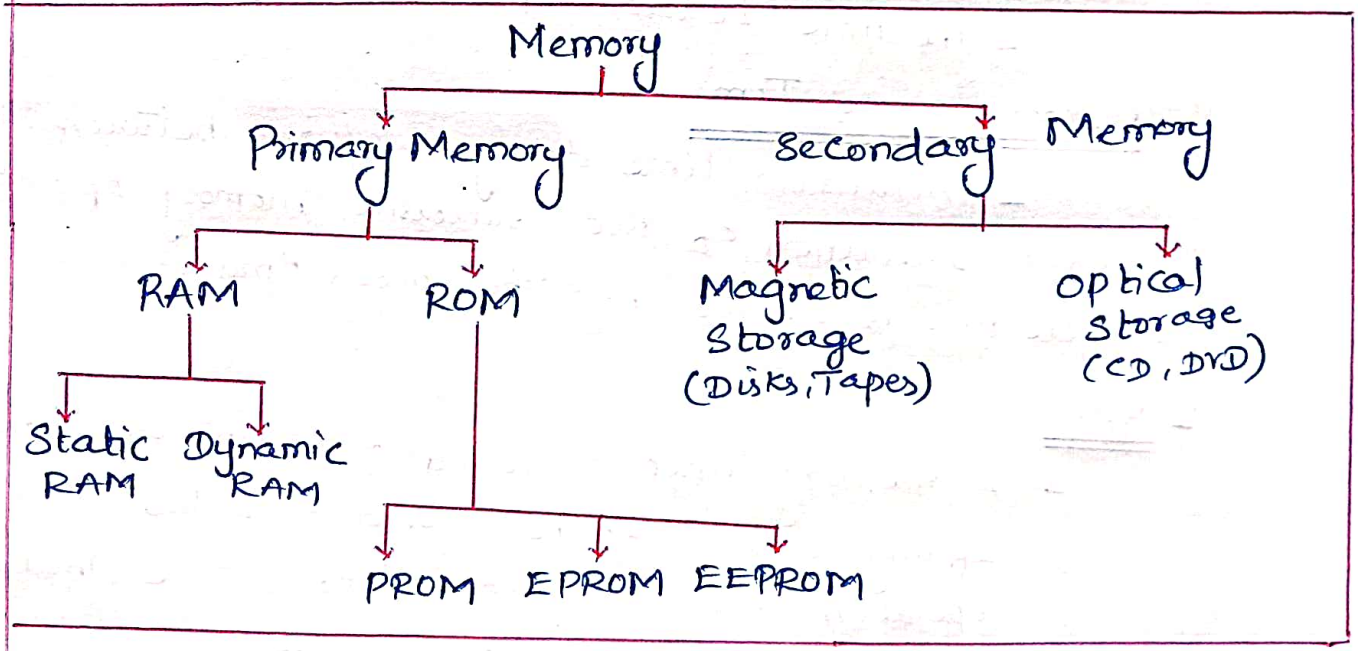
* A memory unit is called random-access memory if any location can be accessed for a Read or write operation in some fixed amount of time that is independent of the location's address.

* **Cache Memory:**
  * A small, fast memory that is inserted between the larger, slower main memory and the processor.
  - holds the currently active segments of a program and their data.
  * Used to reduce the memory access time.

* **Virtual Memory:** - to increase the size of the physical memory
  * Data may be stored in physical memory locations that have addresses different from those specified by the program.

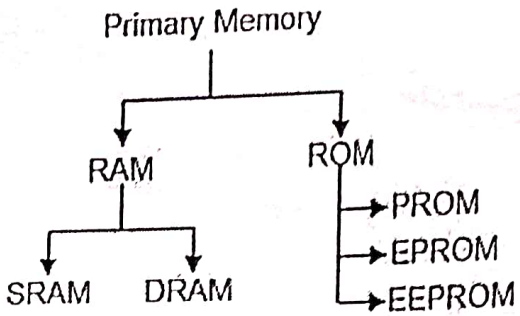
  Virtual or logical address:
  - An address generated by the processor.
  * The virtual address space is mapped onto the physical memory where data are actually stored.

* **Memory management Unit:**
  - The special memory control circuit is used to implement mapping function.
  - The mapping function can be changed during program execution according to system requirements.



. MEMORY TECHNOLOGIES

1) Types of Memories



Volatile Memory(RAM)

- Memory that loses its contents when the computer is switched off or if the power is lost is called as **volatile memory.**
- Random Access Memory(**RAM**) is a volatile Memory. It is also called as **main memory, Primary memory or system memory.**
- In RAM, the informations are accessed and stored **randomly**. It is possible to perform both read and write operations in RAM.
- Types of RAM - i) **DRAM**(Dynamic Random Access Memory)

  ii) **SRAM**(Static Random Access Memory)

Non-Volatile Memory(ROM)

- Memory that does not lose its contents when the computer is switched off or if the power is lost is called as **non- volatile memory.**
- No need to refresh the memory content periodically.
- Read Only Memory(**ROM**) is a non-volatile Memory. It is possible to perform only read operations in ROM.
- **Types of ROM** - i) **PROM**(Programmable Read-Only Memory)

  ii) **EPROM**(Electrically Programmable Read-Only Memory)

  iii) **EEPROM**(Electrically Erasable Programmable Read-Only Memory)

2) Primary Memory technologies

- There are four primary technologies used. They are

  I) Static Random Access Memory (SRAM).

  II) Dynamic Random Access Memory (DRAM).

  III) ROM and Flash Memory.

  IV) Magnetic Disk.

.2.1) SRAM (Static Random Access Memory)

- Used for cache Memories.
- Volatile memory and it holds the data as long as the power is ON.
- No need to refresh at regular intervals.
- Faster and more expensive than Dynamic RAM(DRAM).
- SRAM use **6 to 8 transistors** per bit.

2.2) DRAM (Dynamic Random Access Memory)

- Used for Main Memory.
- Need to be refresed at regular intervals.
- Very sensitive, if power off, it will loss all data.
- Stores each bit in a storage cell consisting of a capacitor and a transistor.
- **Types:** 1. Asynchronous DRAM 2. Synchronous DRAM 3. Rambus Memory.

a. Asynchronous DRAM

- DRAM's stores **all the charge on a capacitor** and must be periodically refreshed. That is why this memory is called dynamic.
- Asynchronous DRAM is faster than synchronous DRAM in sequential operations .
- In DRAM to refresh the cell, we read its contents and write it back. The charge can be kept for several milliseconds.

b. Synchronous DRAM

- In synchronous DRAM, the memory is synchronized with the processor's clock speed.
- The fastest version is called Double Data Rate (DDR). The latest version of this technology is called DDR4.

c. Rambus DRAM

- Rambus Dynamic Random Access memory in short called as Rambus memory is the fastest type of computer memory available. Standard RDRAM can transfer data over 1 GHZ.
- RDRAM is used for **video memory on graphics** accelerator cards.

Difference between SRAM and DRAM

SRAM	DRAM
Information is stored in **Flip Flop**	Information is stored in **capacitor.**
Used in **cache Memory**	Used in **Main Memory**
Information will be stored as long as the power is ON	Information may be lost, if power loss.
No refreshing is needed	**Refreshing** is needed
Less packaging density	**High packaging density**
More complex hardware	**Less complex** hardware
More expensive	**Less expensive**

2.2.3) ROM and Flash Memory

- Read Only Memory (ROM) is a non-volatile Memory. It is possible to perform only read operations in ROM. No need to refresh the memory content periodically.
- Flash Memory is Erasable and rewritable
- **Types of ROM** - i) **PROM**(Programmable Read-Only Memory)
  ii) **EPROM**(Electrically Programmable Read-Only Memory)
  iii) **EEPROM**(Electrically Erasable Programmable Read-Only Memory)

i) PROM(Programmable Read Only Memory)

- The data in PROM is permanent and cannot be changed. Permanent data and programs are stored in PROM .
- PROM are faster and less expensive.

ii) EPROM (Erasable Programmable Read Only Memory)

- EPROM (erasable programmable read-only memory) is programmable read-only memory (programmable ROM) that can be erased and re-used.

Advantage: Its content can be erased and re-programmed.

Disadvantage: When erasing, entire EPROM chip content is erased.

iii) EEPROM (Electrically Erasable Programmable Read Only Memory)

- EPROM (erasable programmable read-only memory) is programmable read-only memory (programmable ROM) that can be erased and re-used using a electrical signal.

Disadvantage: Different voltages are needed for erasing, reading and writing the data.

Flash Memory

- Flash memory is a type of EEPROM. But still there are some major difference between EEPROM and flash memory.
- **In EEPROM,** it is possible to read and write the contents of a single cell.
- **In Flash Memory,** it is possible to read the contents of a single cell, but during writing operations the entire block of cells must be written. The previous contents of the clock of cells must be erased before writing.
- There are two popular high capacity flash memory devices are Flash cards and Flash drivers.

Advantage : i) Flash memory devices have greater density.
ii) Higher capacity and lower cost per bit.
iii) Consumes less power in their operation.
iv) Flash memory usages in various devices.

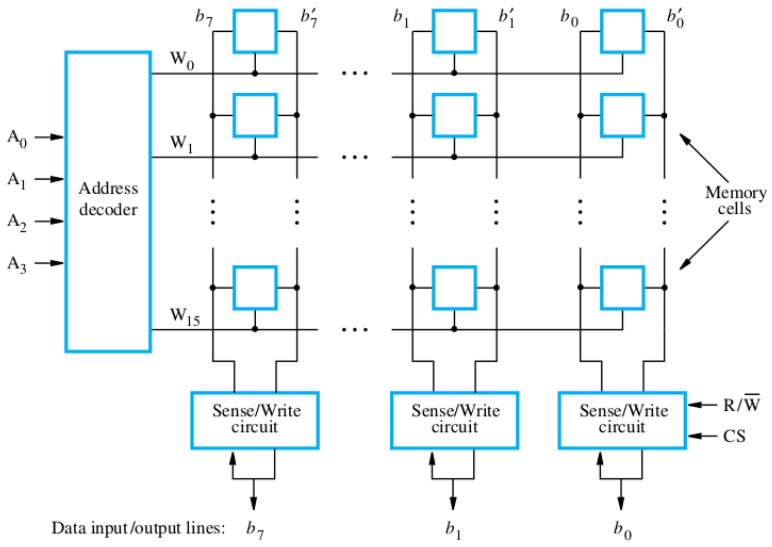


Figure 8.2 Organization of bit cells in a memory chip.

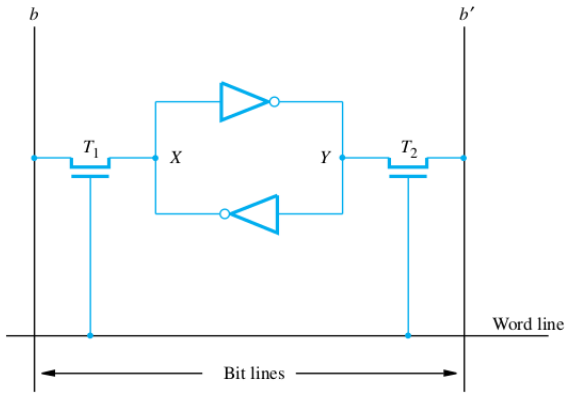


Figure 8.4 A static RAM cell.

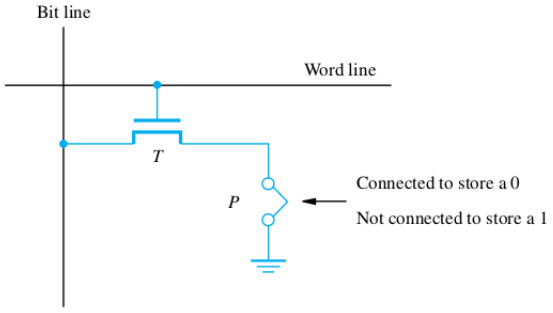


Figure 8.11 A ROM cell.

2.4) Disk Memory

- Provides huge amount of cost effective storage.
- Holds enormous amount of data (i.e. hundreds to thousands of gigabytes).
- Takes millisecond to read information from a disk.
- Hundred thousand times longer than DRAM and a million times longer than from SRAM.

2.4.1) Performance Measures of Disks

The main measures of the qualities of a disk are

1. Capacity
2. Access Time
3. Data Transfer Rate
4. Reliability

i) Capacity

- The storage capacity of a single disk ranges from 10MB to 10GB

ii) Access time

- Time from when a read or write request is isused when data transfer begins.
- The time for repositioning the arm is called **seek time**, and it increases with the distance the arm must move.
- Average seek time is the average of the seek time;
- It is measured over a sequence of (uniformly distributed) random requests, and it is about one third of the worst-case seek time.

iii) Data transfer rate

- Rate at which data can be retrieved from or stored to the disk.

iv) Reliability

- Measured by the mean time to failure
- The typical mean time to failure of disks ranges from 30,000 to 800,000 hours

MEMORY HIERACHY

- Multiple levels of memories is called **memory hierarchy.**
- A memory hierarchy is classified based on **SPEED, SIZE and COST.**
- The **faster** memories are more expensive than the slow memories and they are smaller.
- As the distance from the processor increases, the size of the memories and the access time both increases.
- The memory hierarchy is

  i) Processor Register

  ii) Primary Cache (or) Level 1 Cache

  iii) Secondary Cache

  iv) Main Memory

  v) Secondary Memory

i) Processor Register

- The **speed** of the data in the processor register is **very fast** compared to other memories.
- The **size** of processor register is **very small** compared to other memories.
- The **cost** of processor register is **very high** compared to other memories .

ii) Primary Cache (Or) Level 1 Cache

- A primary cache is placed between the processor register and secondary cache (L2 cache). It is referred to as Level 1 cache(L1). It is larger than the primary cache. It is usually implemented using SRAM chips.
- The **speed** of the data in the primary cache is faster than L2 cache but lesser than registers.
- The **size** of primary cache is smaller than L2 cache but higher than registers.
- The **cost** of primary cache is higher than L2 cache but lesser than registers.
- Primary cache holds the copies of instruction and data that are needed for current execution.

iii) Secondary Cache (Or) Level 2 Cache

- A secondary cache is placed between the **primary cache(Level 1 cache) and Main Memory.** It is referred to as **Level 2 cache(L2).**
- It is larger than the primary cache.
- The cost and the speed is lesser than primary memory.
- It is usually implemented using SRAM chips.

iv) Main Memory

- A main memory is placed between the secondary cache and secondary Memory.
- It is larger than the secondary cache. The cost and the speed is lesser than secondary cache but higher than secondary memory.

- This is large memory and is implemented using dynamic memory components like

  a) Single in-line memory module(SIMM),

  b) Dual in-line memory module(DIMM)

  c) Rambus in-line memory module(RIMM)

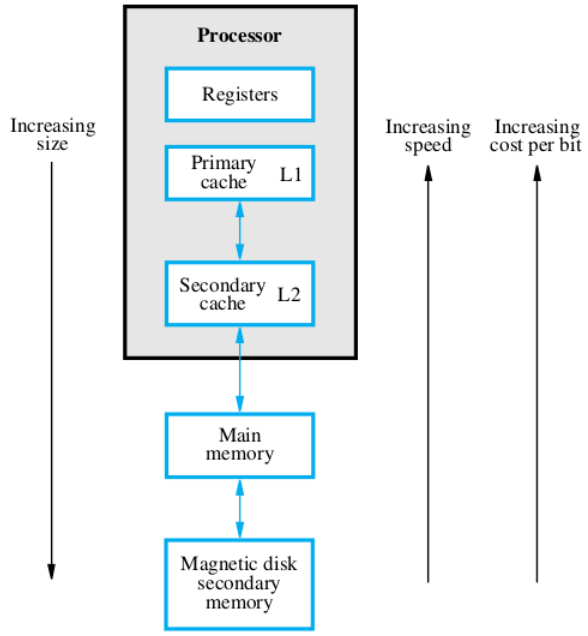


Figure 8.14 Memory hierarchy.

v) Secondary Memory

- Magnetic disk storage units are called as secondary memory.
- The size of secondary memory is very high, the cost is lower and the speed is very slow compared to other memories.

Characteristics of Memory

Memory	Speed	cost	Size
Processor Register	Higher	Higher	very Low
Primary cache (L1)	High	High	Low
Secondary Cache (L2)	Low	Low	Higher than Primary cache
Main Memory	Lower than Secondary cache	Lower than Secondary Cache	High
Secondary Memory	very Low	very Low	Higher

MEMORY MANAGEMENT

- The task of subdivision is carried out dynamically by the OS and is known as memory management.
- Uniprogramming system:
  - main memory is divided into two parts: one part for the OS (resident monitor) and one part for the program currently being executed.
- Multiprogramming system:
  - the "user" part of memory is subdivided to accommodate multiple processes.

i) Swapping

- Three types of queues:
  1. the long-term queue of requests for new processes,
  2. the short-term queue of processes ready to use the processor, and
  3. the various I/O queues of processes that are not ready to use the processor.
- I/O activities are much slower than computation and therefore the processor in a uniprogramming system is idle most of the time.

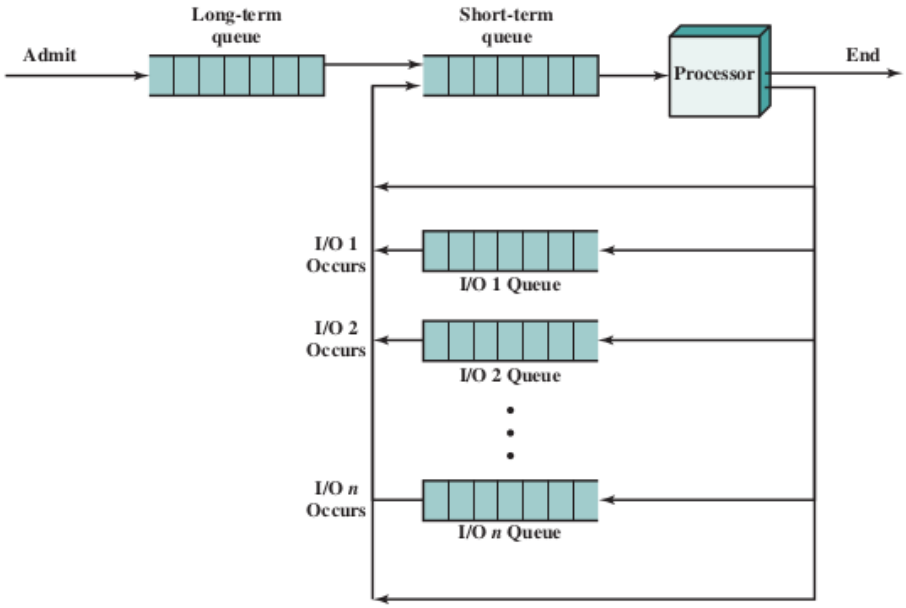
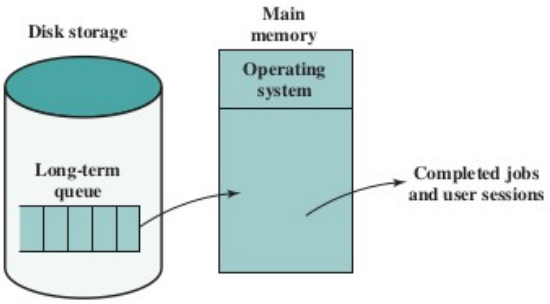


Figure 8.11 Queuing Diagram Representation of Processor Scheduling

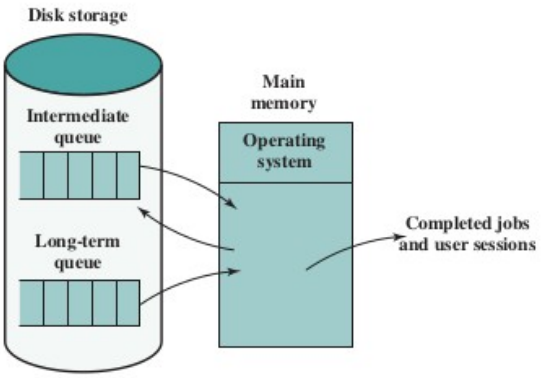
- Memory holds multiple processes and that the processor can move to another process when one process is waiting.
- But the processor is so much faster than I/O that it will be common for all the processes in memory to be waiting on I/O.
- Thus, even with multiprogramming, a processor could be idle most.

Solution is swapping

- a long-term queue of process requests, typically stored on disk.
- These are brought in, one at a time, as space becomes available.
- As processes are completed, they are moved out of main memory.
- Now the situation will arise that none of the processes in memory are in the ready state (e.g., all are waiting on an I/O operation).



(a) Simple job scheduling



(b) Swapping

- The processor swaps one of these processes back out to disk into an intermediate queue.
- This is a queue of existing processes that have been temporarily kicked out of memory.
- The OS then brings in another process from the intermediate queue, honors a new process request from the long-term queue. Execution then continues with the newly arrived process.

ii) Partitioning

- The simplest scheme for partitioning available memory is to use fixed-size partitons.
- Even with the use of unequal fixed-size partitions, there will be wasted time
- In most cases, a process will not require exactly as much memory as provided by the partition.
    - For example,
        - a process that requires 3M bytes of memory would be placed in the 4M partition, wasting 1M that could be used by another process.

It leads situation in which there are a lot of small holes in memory.
As time goes on, mem becomes more and more fragmented, and memory utilization declines.

One technique for overcoming this problem is compaction.
A process in memory consists of instructions plus data. The instructions will contain addresses for memory locations of two types:
*Addresses of data items
* Addresses of instructions, used for branching instructions

A logical address is expressed as a location relative to the beginning of the program Instructions in the program contain only logical addresses.
A physical address is an actual location in main memory. When the processor executes a process, it automatically converts from logical to physical address by adding the current starting location of the process, called its base address)

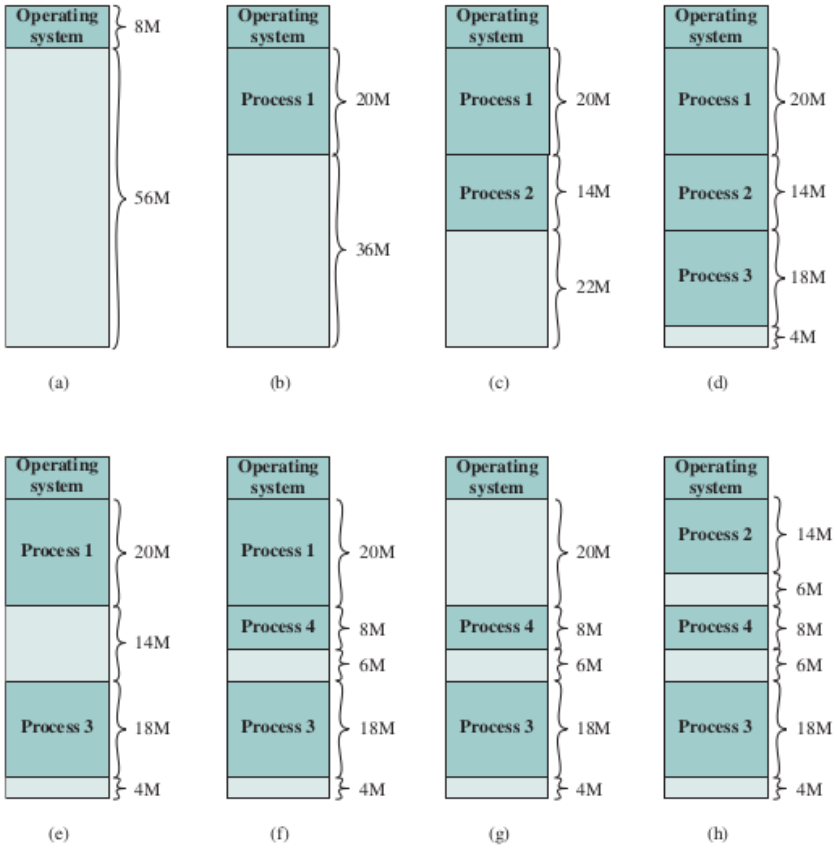


Figure 8.14 The Effect of Dynamic Partitioning

iii) Paging

- unequal fixed-size and variable-size partitions are inefficient in the use of memory
- Suppose, however that memory is partitioned into equal fixed-size chunks that are relatively small, and that each process is also divided into small fixed-size chunks of some size.
- The chunks of a program, known as **pages**, could be as signed to available chunks of memory, known as **frames** or page frames

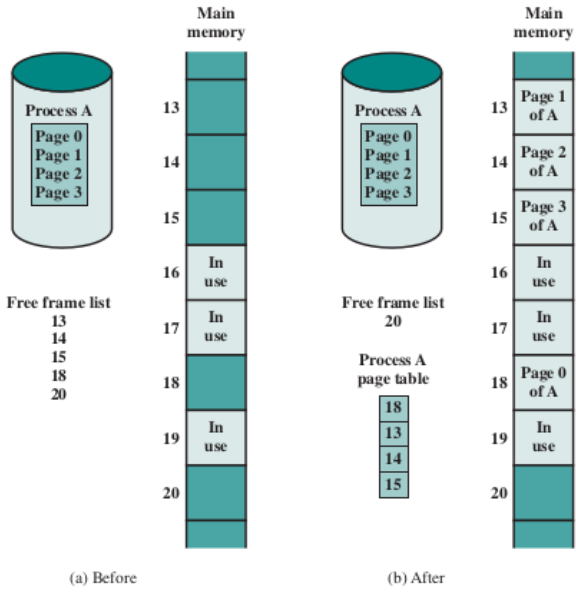


Figure 8.15 Allocation of Free Frames

- At a given point in time, some of the frames in memory are in use and some are free,

- The list of freeframes is maintained by the OS. Process A, stored on disk, consists of four When it comes time to load this process, the OS finds four free frames and loads the four pages of the process A into the four frames page

Example, that there are not sufficient unused contiguous frames to hold the process. Does this prevent the OS from loading A?
The answer is no, because once again use the concept of logical address.

- A simple base address will no longer suffice. Rather, the OS maintains a page table for each process.
- The page table shows the frame location for each page of the process.
- Within the program, each logical address consists of a page number and a relative address within the page.

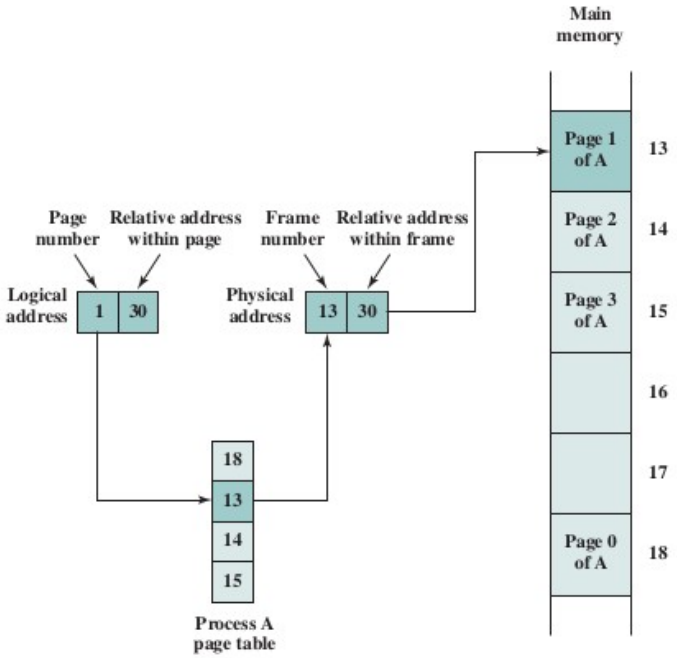


Figure 8.16 Logical and Physical Addresses

- 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.
- Thus, at any one time, only a few pages of any given process are in mem and therefore more processes can be maintained in memory, When it brings one pag it must throw another page out; this is known as page replacement).

iv) Segmentation

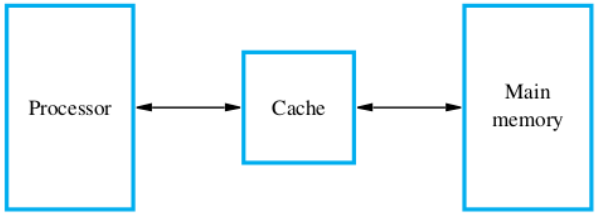
- Segmentation allows the programmer to view memory as consisting of mul tiple address spaces or segments.
- Segments are of variable, indeed dynamic, size.
- The programmer or the OS will assign programs and data to different segments.
- There may be a number of program segments for various types of programs as well as a number of data segments.

- Each segment may be assigned access and usage rights. Memory references consist of a (segment number, offset) form of address.
- This organization has a number of advantages to the programmer over a segmented address space:
    1. It simplifies the handling of growing data structures. If the programmer does not know ahead of time how large a particular data structure will become, it not necessary to guess. The data structure can be assigned its own segment and the OS will expand or shrink the segment as needed.
    2. It allows programs to be altered and recompiled independently without requiring that an entire set of programs be relinked and reloaded. Again, this is accomplished using multiple segments.
    3. It lends itself to sharing among processes. A programmer can place a utility program or a useful table of data in a segment that can be addressed by other processes
    4. It lends itself to protection. Because a segment can be constructed to contain well-defined set of programs or data, the programmer or a system administra tor can assign access privileges in a convenient fashion

Mapping function and replacement techniques.

* Cache memory
* Mapping function
  * Direct
  * Associative
  * Set - associative
* Replacement algorithm

Cache memory



* The Speed of the main memory is very Low in comparison with the speed of modern processors.
* The processor cannot spend much of its time waiting to access instruction and data in main memory.
* Hence, it is important to devise a scheme that reduces the time needed to access the necessary information.
* An efficient solution is to use a fast cache memory which essentially makes the main memory appear to the processor to be faster than it really is.

Types of cache Memory:

i) Primary cache

→ referred as L1 cache.

→ Located in processor chip

ii) Secondary cache

→ referred as L2 cache

→ placed b/w L1 cache & MM

Advantages of cache:

→ Faster than Main Memory

→ Less access time

→ can be executed in short time

→ store data for temporary use.

Disadvantages of cache:

→ Limited capacity

→ very expensive

* The effectiveness of the cache mechanism is based on a property of computer programs called locality of reference.

* Two types of locality of reference.
  * Temporal locality of reference (Based on time)
  * Spatial locality of reference (Based on space)

a) * Temporal aspect of the locality of reference suggest that whenever an information item (instruction or data) is first needed, this item should be brought into the cache where it will hopefully remain until it is needed again.

b) * The Spatial aspect suggests that instead of fetching just one item from the main memory to the cache, it is useful to fetch several items that reside at adjacent addresses.

* We will use the term block to refer a set of contiguous address locations of some size
* Another term that is often used to refer to a cache block is cache line.

* Processor issues Read and write request using addresses that refer to location in the memory.

* The cache control circuit determines whether

the requested word currently exist in the cache

* If the requested word is in the cache, it is a read or write hit.
* The data present in the cache memory called cache hit.

Read hit.

The data is obtained from the cache hit.

Write hit

* Cache memory hold a copy the contents of the main memory.
* Write hit can be done in two ways.
    * Write through protocol.
    * Write back or copy back protocol.

Write through protocol.

The cache location and the main memory location are updated simultaneously.

Write back or copy back protocol.

* The second technique is to update only the cache location and to mark it as updated with an associated flag bit, often called directly or modified bit.
* The main memory location of the word is updated later, when the block containing this marked word is to be removed from the cache to make for a new block.

* Cache miss.

The data is not present in the cache memory called cache miss.

* Read miss

* Read operation is not in the called called Read miss.
* Block of words containing this required word is transferred from the memory.
* After the block is transferred, the desired word is forwarded to the processor.
* The desired word may also be forwarded to processor as soon as it is transferred without waiting for the entire block to be transferred

* Write miss

* Write operation is not in the cache called write miss.
* Write Operation can be done in two ways.
    * Write through protocol.
    * Write back protocol.

a) Write through protocol.

Write through protocol is used then contents of the main memory are updated directly.

b) Write back protocol.

Write back protocol is used block containing addressed word is first brought into the cache is overwritten.

Mapping function

The mapping functions are used to map a particular block of main memory to a particular block of cache.

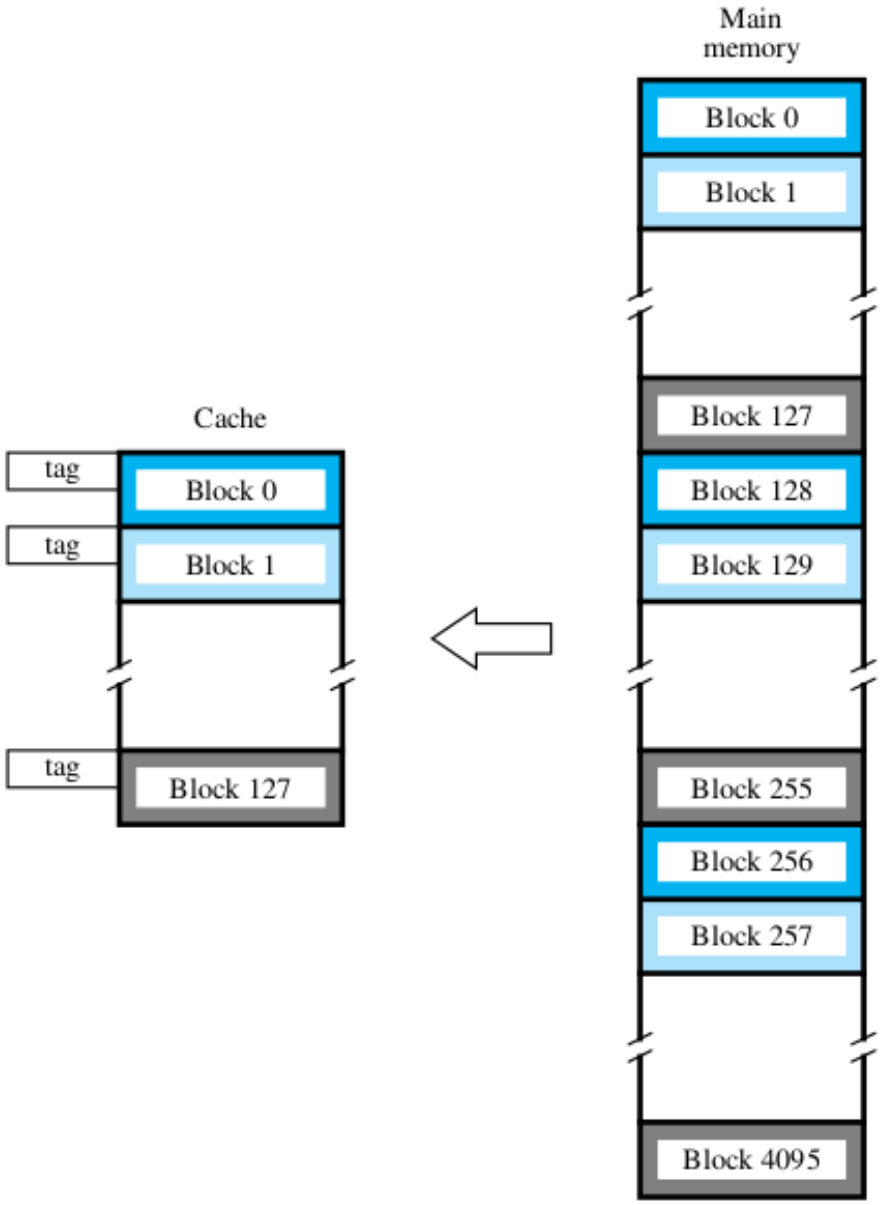
This mapping function is used to transfer the block from main memory to cache memory.

Types of mapping

* Direct mapping
* Associative mapping
* Set associative mapping.

a) Direct mapping

* The simplest way to determine cache location in which to store memory blocks is the direct mapping technique.
* Cache consisting of 128 blocks of 16 words each, for a total of 2048 (2K) words, and assume that the main memory is addressable by a 16-bit address.
* The main memory has 64.



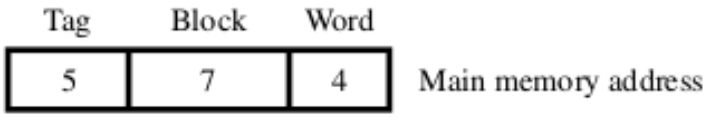
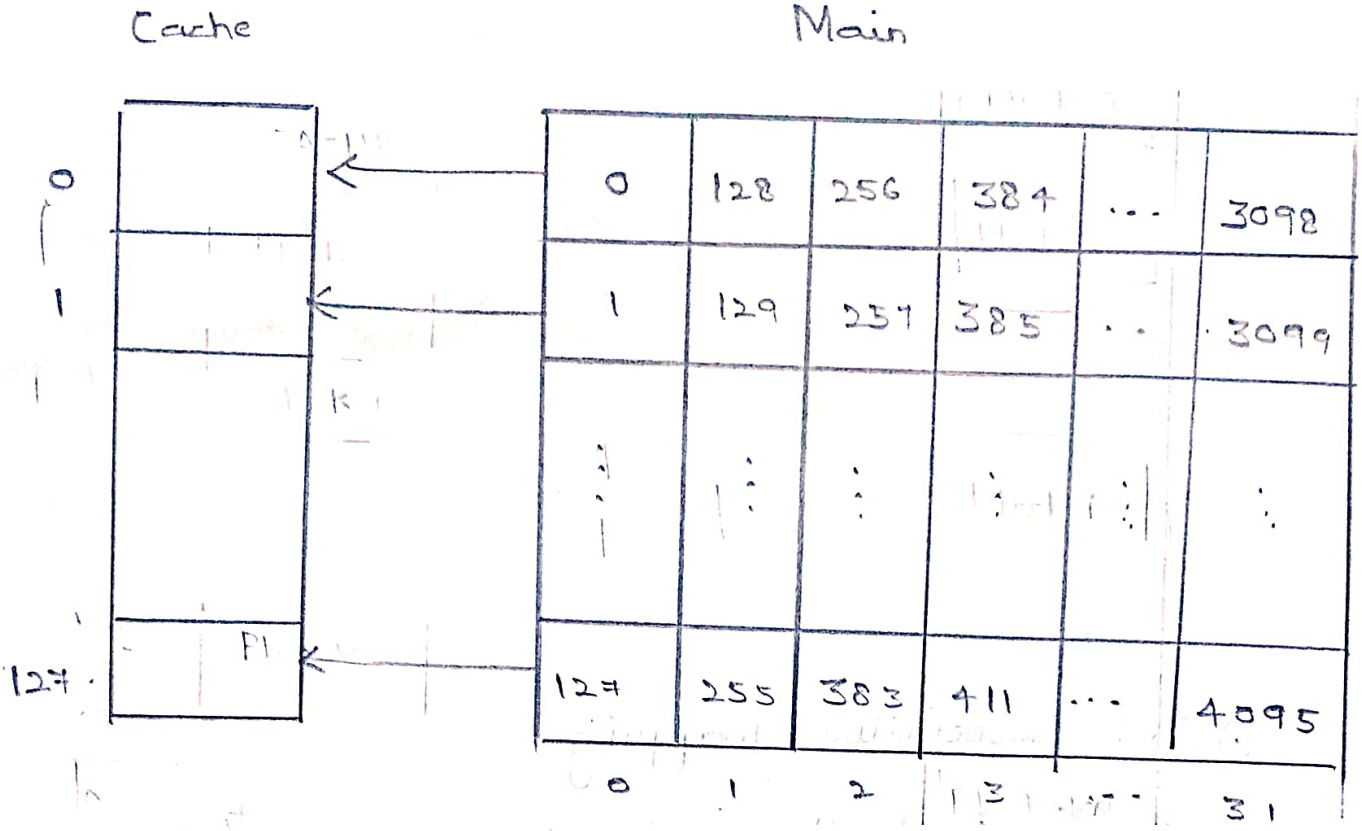


Figure Direct-mapped cache.



b) Associative mapping.

* The tag bits of an address received from the processor are compared to the tag bits of each block of the cache to see if the desired block is present called associative mapping function.

* In this mapping function, any block of the main memory can potentially reside in any cache block position.

* This is much more flexible mapping method.

* The cost of an associative cache is higher than the cost of a direct-mapped cache.

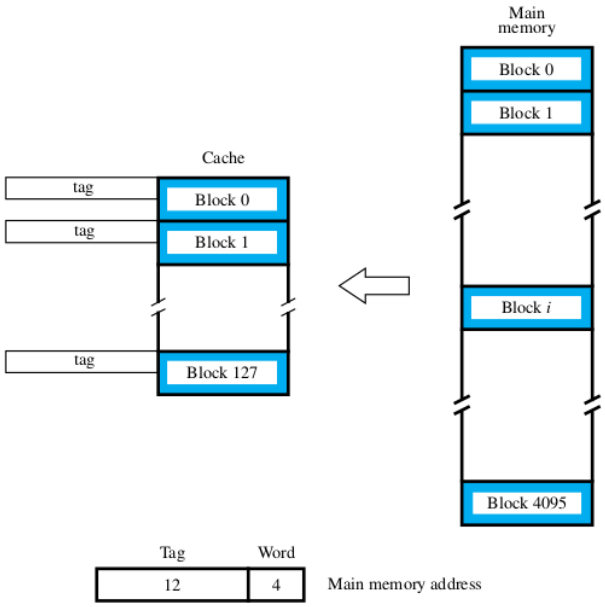
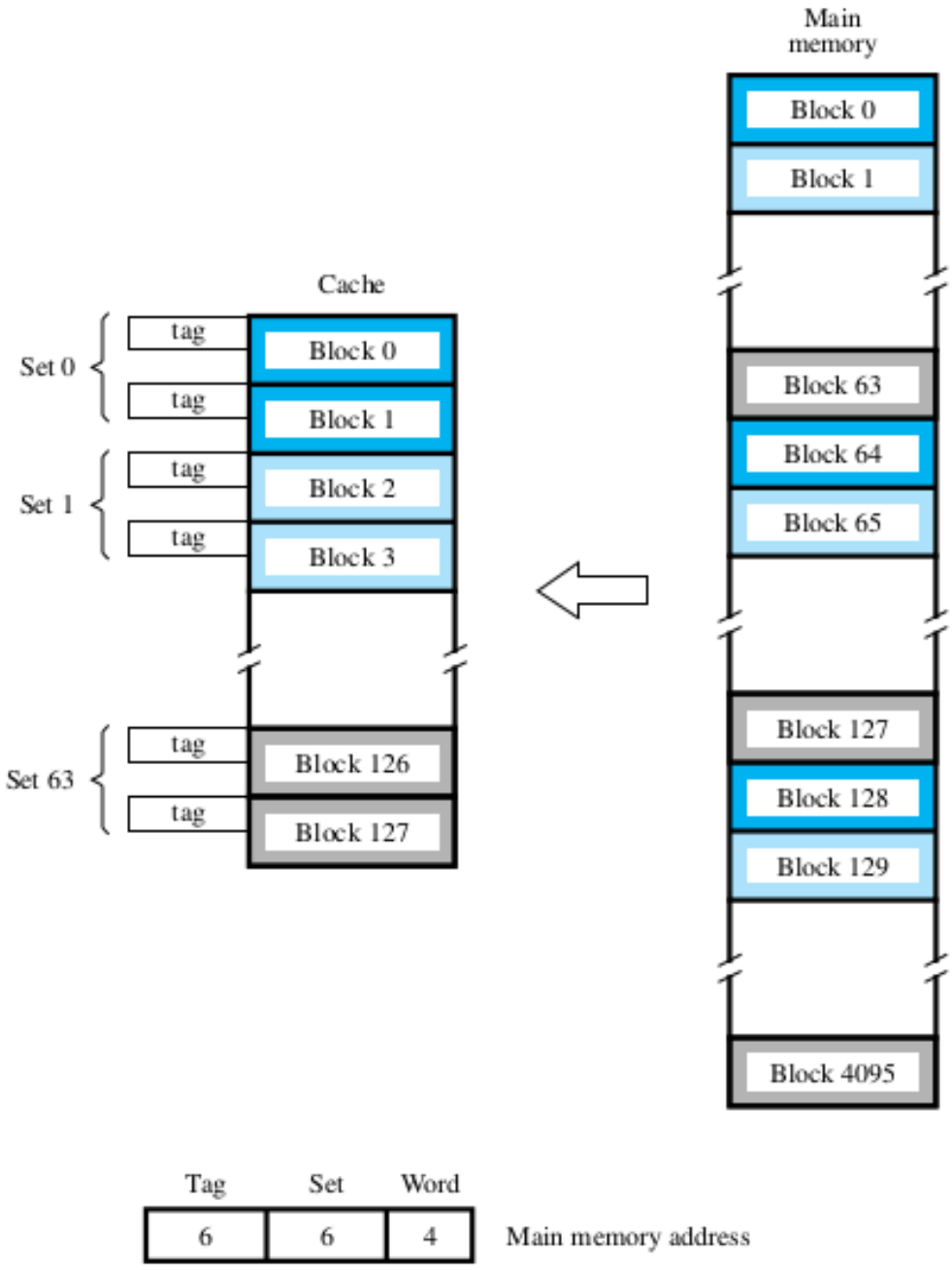


Figure Associative-mapped cache.

c) Set Associative mapping

* A combination of the direct and associative mapping techniques can be used.
* Blocks of the cache are grouped into sets, and the mapping allows a block of the main memory to reside in any block of a specific set.
* In this case, memory blocks 0, 64, 128 ... 4032 map into cache set 0, and they can occupy either of the two block position within the set.
* For example, in main memory block 0 can be stored in any place of set 0 in cache memory.



Set-associative-mapped cache with two blocks per set.

Replacement Algorithm

* When the cache is full, and a block of words needs to be transferred from the main memory, some blocks of words in the cache must be replaced.
* This is determined by replacement algorithm.

(1) * In a direct mapped cache, the position of each block is predetermined, hence no replacement strategy exists.

(2) * In associative and set-associative caches there exists some flexibility.

* When a new block is to be brought into the cache and all the positions that it may occur are full, the cache controller must decide which of the old blocks to overwrite.
* The property of locality of reference in programs gives a clue to a reasonable strategy.
* When a block is to be overwritten, it is sensible to overwrite the one that has gone the longest time without being referenced called the Least recently used block, and the technique is called the LRU replacement algorithm.

* To use the LRU algorithm, the cache controller must track reference to all blocks as computation proceeds.

* Suppose it is required to track the LRU block of a four block set in a set associative cache, 2 bit counter can be used.

1) * When a hit occurs, counter of the block that is referenced is set to 0.

* Counters with values originally lower than the referenced one are incremented by one, and all the other remain unchanged.

2) a) * When a miss occurs and set is not full, counter associated with new block loaded from main memory is set to 0, and values of all other counters are increased by 1.

b) * When a miss occurs and set is full, block with the counter value 3 is removed, new block put in its place, and its counter is set to 0.

* The LRU algorithm has been used extensively. It lead to a poor performance in some cases.

* Performance of the LRU algorithm can be improved by introducing a small amount of randomness in deciding which block to replace.

* Several other replacement algorithm are also used in practice.

* An intuitively reasonable rule would be to remove the "oldest" block from a full set when a new block must be brought in.

* However, because this algorithm does not take into account the recent pattern of access to blocks in the cache, it is generally not as effective as the LRU algorithm in choosing the best blocks to remove.

* The simplest algorithm is to randomly choose the block to be overwritten.

* The simple algorithm has been found to be quite effective in practice.

Virtual memory

Definition

* It is a technique that uses main memory as a 'cache' for secondary storage.

* The term virtual memory refers to something which appears to be present but actually it is not

* The virtual memory technique allows users to use more memory for a program than the real memory of a computer.

Physical Address → An address in main memory.

Protection:

* It is a set of mechanisms for ensuring that multiple processes sharing the processor, memory or I/O devices cannot interface, intentionally or unintentionally with one another by reading or writing each other's data. The mechanism also isolate the operating system from a user process.

Page Fault:

- An event that occurs when an accessed page is not present in main memory.

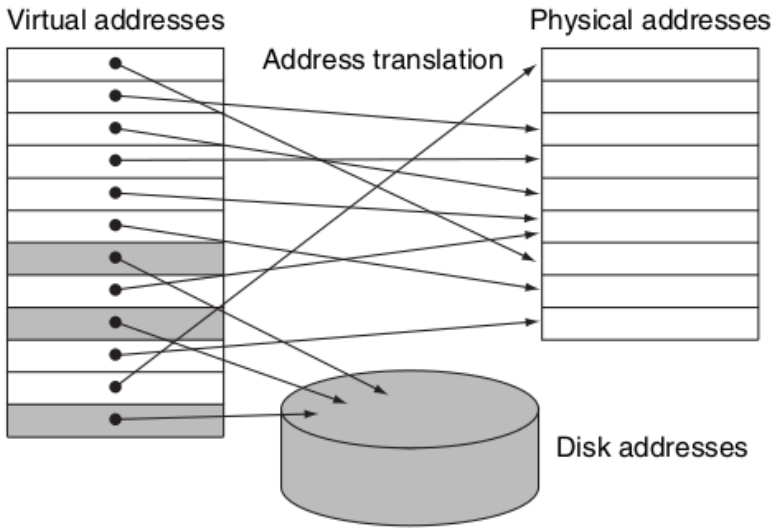


FIGURE In virtual memory, blocks of memory (called *pages*) are mapped from one set of addresses (called *virtual addresses*) to another set (called *physical addresses*). The processor generates virtual addresses while the memory is accessed using physical addresses. Both the virtual memory and the physical memory are broken into pages, so that a virtual page is mapped to a physical page. Of course, it is also possible for a virtual page to be absent from main memory and not be mapped to a physical address; in that case, the page resides on disk. Physical pages can be shared by having two virtual addresses point to the same physical address. This capability is used to allow two different programs to share data or code.

first word for typical page sizes, leads to several key decisions in designing virtual memory systems:

- Pages should be large enough to try to amortize the high access time. Sizes from 4 KiB to 16 KiB are typical today. New desktop and server systems are being developed to support 32 KiB and 64 KiB pages, but new embedded systems are going in the other direction, to 1 KiB pages.
- Organizations that reduce the page fault rate are attractive. The primary technique used here is to allow fully associative placement of pages in memory.
- Page faults can be handled in software because the overhead will be small compared to the disk access time. In addition, software can afford to use clever algorithms for choosing how to place pages because even small reductions in the miss rate will pay for the cost of such algorithms.
- Write-through will not work for virtual memory, since writes take too long. Instead, virtual memory systems use write-back.

Virtual address : * An address that corresponds to a location in virtual space and is translated by address mapping to a physical address when memory is accessed.

Address translation :

* It is also called address mapping. The process by which a virtual address is mapped to an address used to access memory.

* In virtual memory, the address is broken into a virtual page number and a page offset. Figure shows the translation of the virtual page number to a physical page number

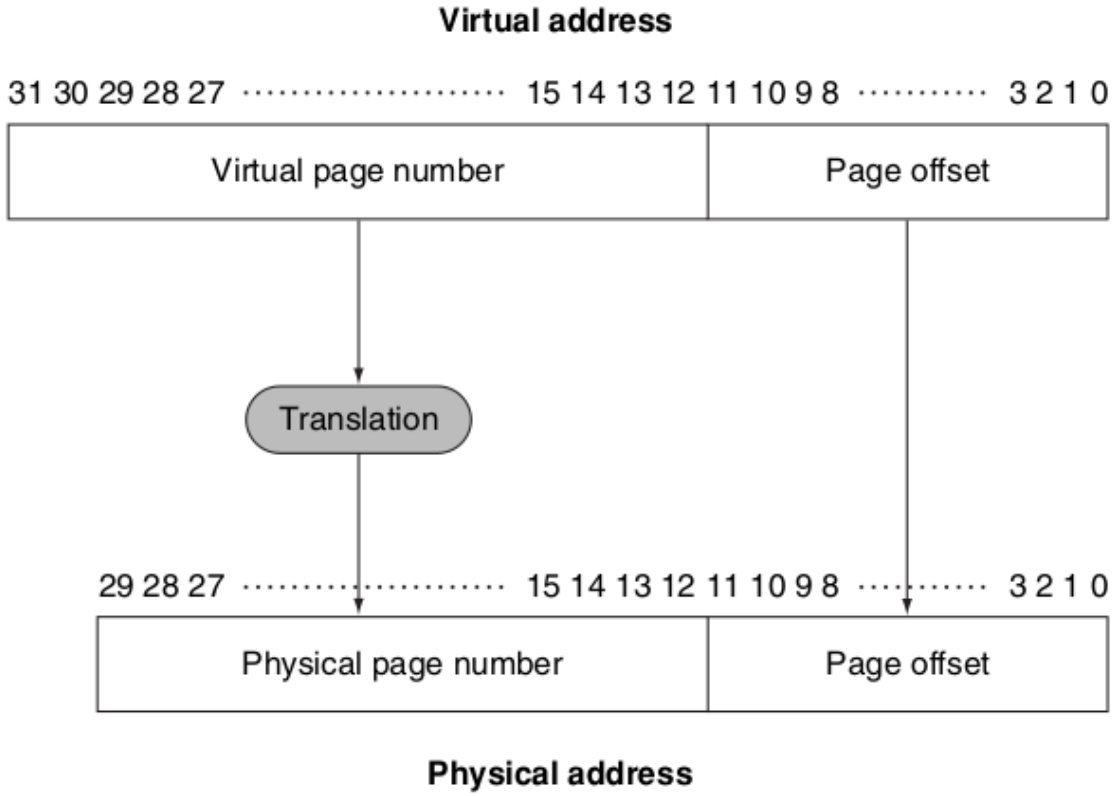


FIGURE **Mapping from a virtual to a physical address.** The page size is 2^{12} = 4 KiB. The number of physical pages allowed in memory is 2^{18}, since the physical page number has 18 bits in it. Thus, main memory can have at most 1 GiB, while the virtual address space is 4 GiB.

gives the starting address of the page table. In this figure, the page size is 2^{12} bytes, or 4 KiB. The virtual address space is 2^{32} bytes, or 4 GiB, and the physical address space is 2^{30} bytes, which allows main memory of up to 1 GiB. The number of entries in the page table is 2^{20}, or 1 million entries. The valid bit for each entry indicates whether the mapping is legal. If it is off, then the page is not present in memory. Although the page table entry shown here need only be 19 bits wide, it would typically be rounded up to 32 bits for ease of indexing. The extra bits would be used to store additional information that needs to be kept on a per-page basis, such as protection.

* The physical page number constitude the upper portion of the physical address while the page off set, which is not charged, constitudes the lower portion.

* The number of bits in the page set field determines the page size. The number of page addressable with the virtual address need not match the number of pages addressable with the physical address.

Pages:

* Both the virtual memory and the physical memory are broken into pages, So that a virtual page is mapped to a physical address

* Physical pages can be shared by having two virtual address point to the same Physical address. The capability is used to allow two different programs to share data or code.

Segmentation

*It is a variable size address mapping scheme in which an address consist of two parts: a segment number, which is mapped to a physical address and segment off set.

Page table

* It is the table containing the virtual to physical address translations in a virtual memory system.

* The table which is stored in memory is typically indexed by the virtual page number; each entry in the table contain physical page number for the virtual page if the page is currently in memory.

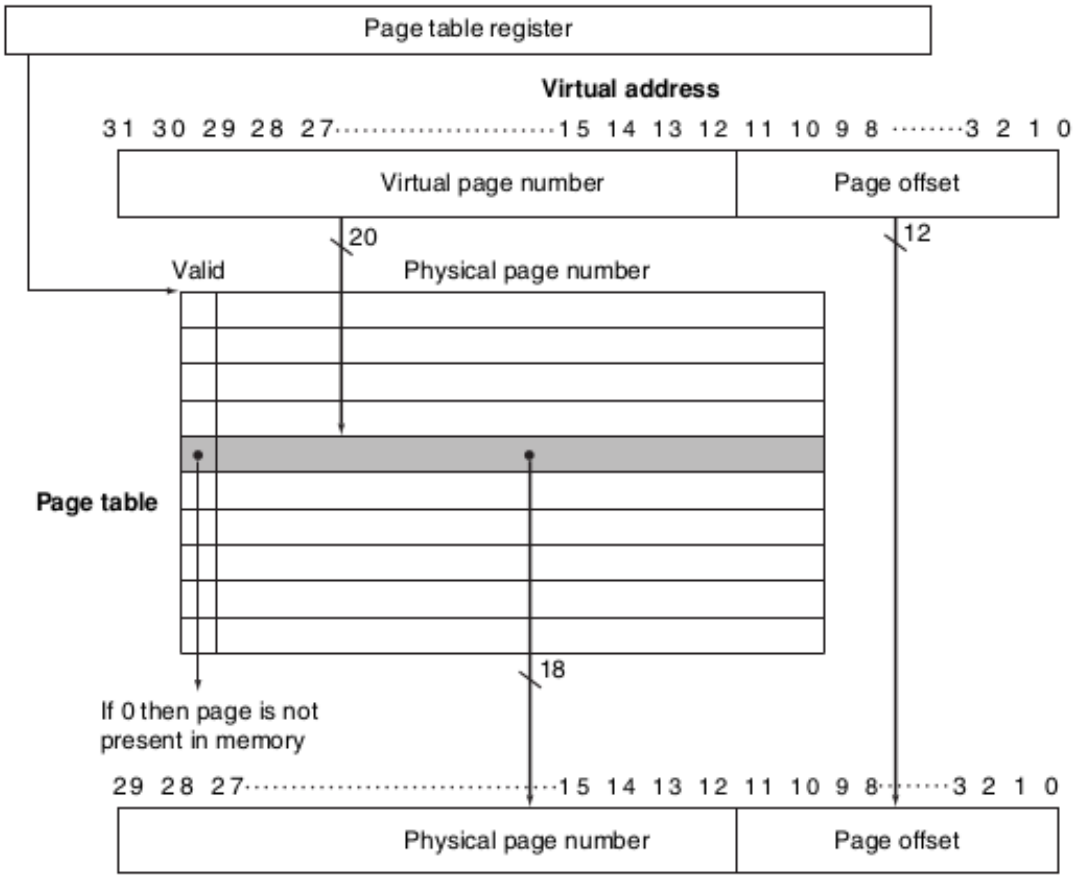


FIGURE number main m

KiB. The it. Thus,

FIGURE The page table is indexed with the virtual page number to obtain the corresponding portion of the physical address. We assume a 32-bit address. The page table pointer gives the starting address of the page table. In this figure, the page size is 2^{12} bytes, or 4 KiB. The virtual address space is 2^{32} bytes, or 4 GiB, and the physical address space is 2^{30} bytes, which allows main memory of up to 1 GiB. The number of entries in the page table is 2^{20}, or 1 million entries. The valid bit for each entry indicates whether the mapping is legal. If it is off, then the page is not present in memory. Although the page table entry shown here need only be 19 bits wide, it would typically be rounded up to 32 bits for ease of indexing. The extra bits would be used to store additional information that needs to be kept on a per-page basis, such as protection.

* The difficulty in using fully associative placement in locating an entry, since it can be anywhere in the upper level of the hierarchy. A full search is impractical

* In virtual memory systems, we locate pages by using a table that indexes the memory; this structure is called a page table and it resides in memory

Page table register

* To indicate the location of the page table in memory, the hardware includes a register that points to the start of page table; we call this the page table register.

* When a page fault occurs, if all th Pages in main memory are in use, the operating system must choose a page to replace

* Using the past to predicate the future, operating system follow the least recently used (LRU) replacement scheme. The operating system searches for the least recently used page, assuming that a page that has not been used in long time

time is less likely to be needed than a more recently accessed page.

Swap space :

* It is the space on the disk reserved for the full virtual memory space of a process

Reference bit

* It is also called use bit. A field that is set whenever a page is accessed and that is used to implement LRU or other replacement schemes

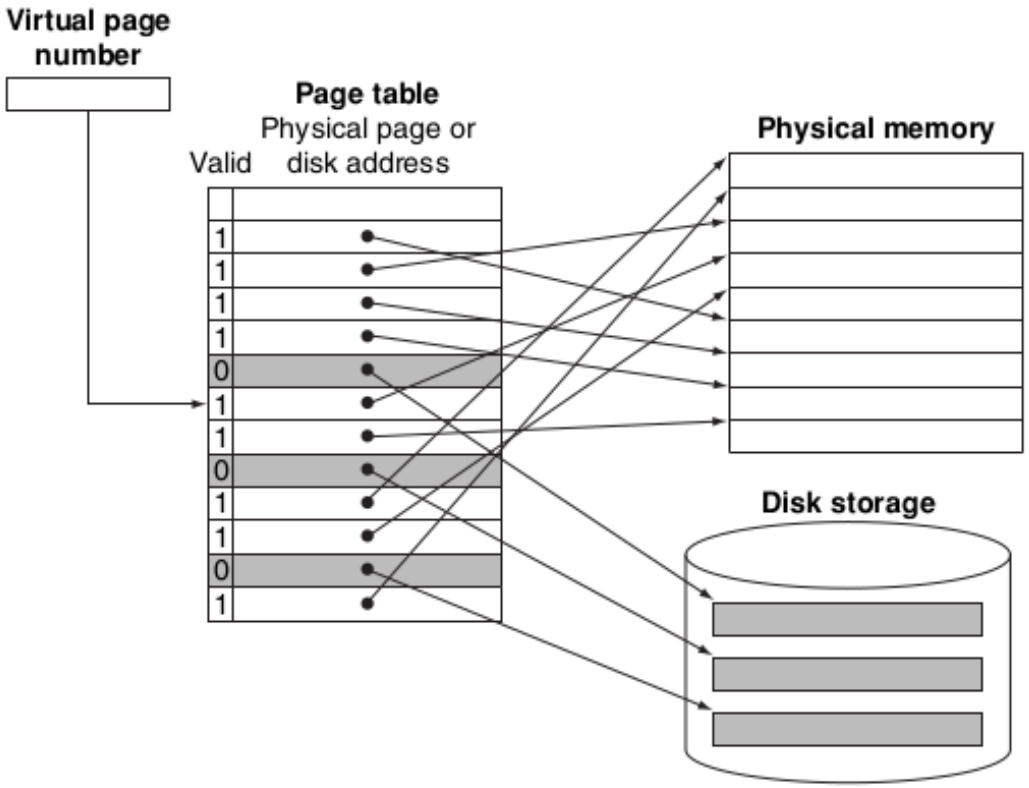


FIGURE The page table maps each page in virtual memory to either a page in main memory or a page stored on disk, which is the next level in the hierarchy. The virtual page number is used to index the page table. If the valid bit is on, the page table supplies the physical page number (i.e., the starting address of the page in memory) corresponding to the virtual page. If the valid bit is off, the page currently resides only on disk, at a specified disk address. In many systems, the table of physical page addresses and disk page addresses, while logically one table, is stored in two separate data structures. Dual tables are justified in part because we must keep the disk addresses of all the pages, even if they are currently in main memory. Remember that the pages in main memory and the pages on disk are the same size.

Translation - lookaside Buffer (TLB)

* Translation lookaside Buffer (TLB) a cache that keeps track of recently used address mappings to try to avoid an access to the page table

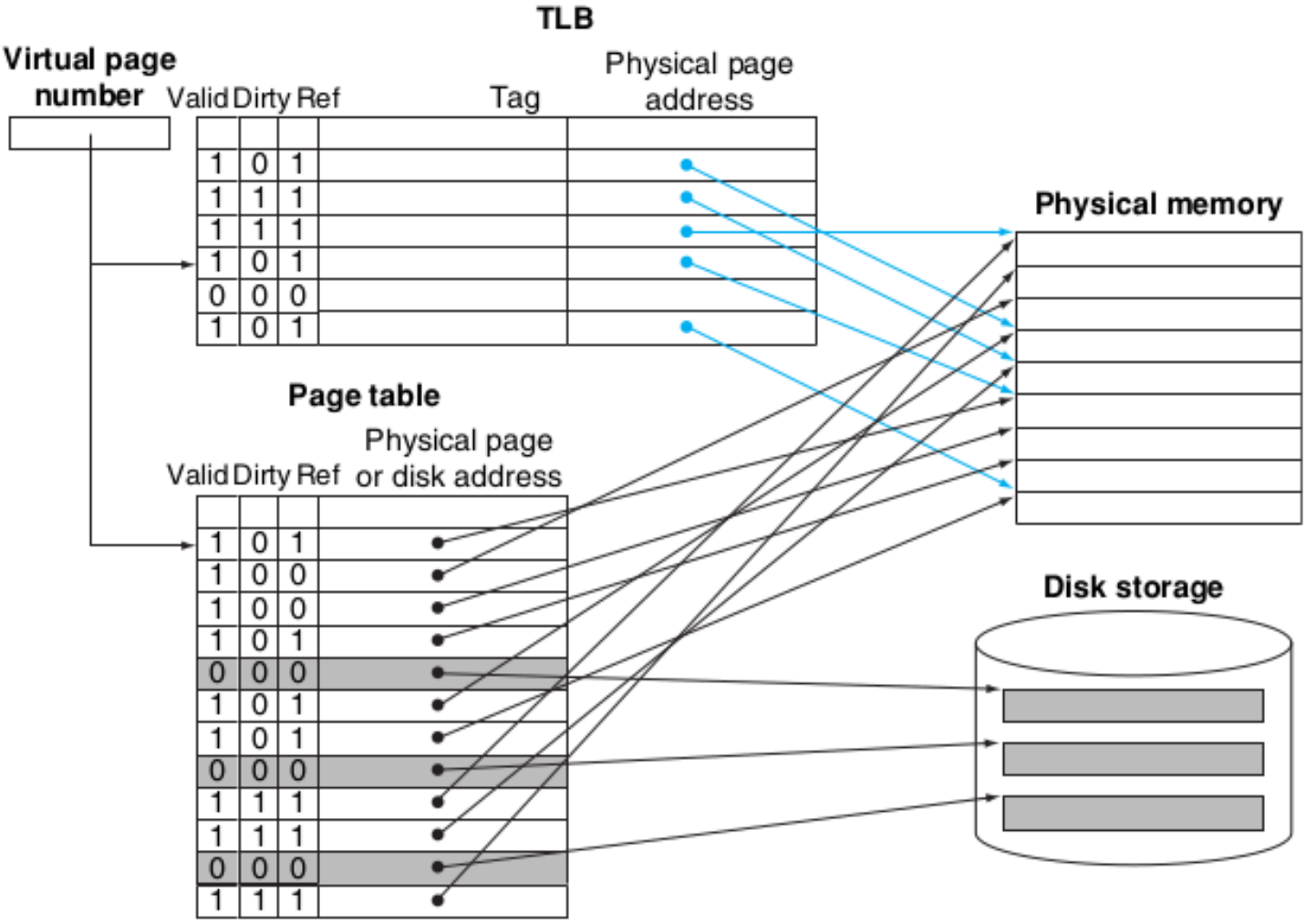


FIGURE 5.29 The TLB acts as a cache of the page table for the entries that map to physical pages only. The TLB contains a subset of the virtual-to-physical page mappings that are in the page table. The TLB mappings are shown in color. Because the TLB is a cache, it must have a tag field. If there is no matching entry in the TLB for a page, the page table must be examined. The page table either supplies a physical page number for the page (which can then be used to build a TLB entry) or indicates that the page resides on disk, in which case a page fault occurs. Since the page table has an entry for every virtual page, no tag field is needed; in other words, unlike a TLB, a page table is *not* a cache.

* Because we access the TLB instead of the page table on every reference, the TLB will need to include other status bits, such as the dirty and the reference bits.

Reference Bit:

* If we get a hit, the physical page number is used to form the address, and the corresponding reference bit is turned on.

Dirty bit:

* If the processor is performing a write, the dirty bit is also turned on. If a miss in the TLB occurs, we must determine whether it is a page fault or purely a TLB miss.

* If the page exist in memory, then the TLB miss indicates only that the translation is missing. In such case, the processor can handle the TLB miss by loading the translation from the page table into the TLB and then trying the reference again.

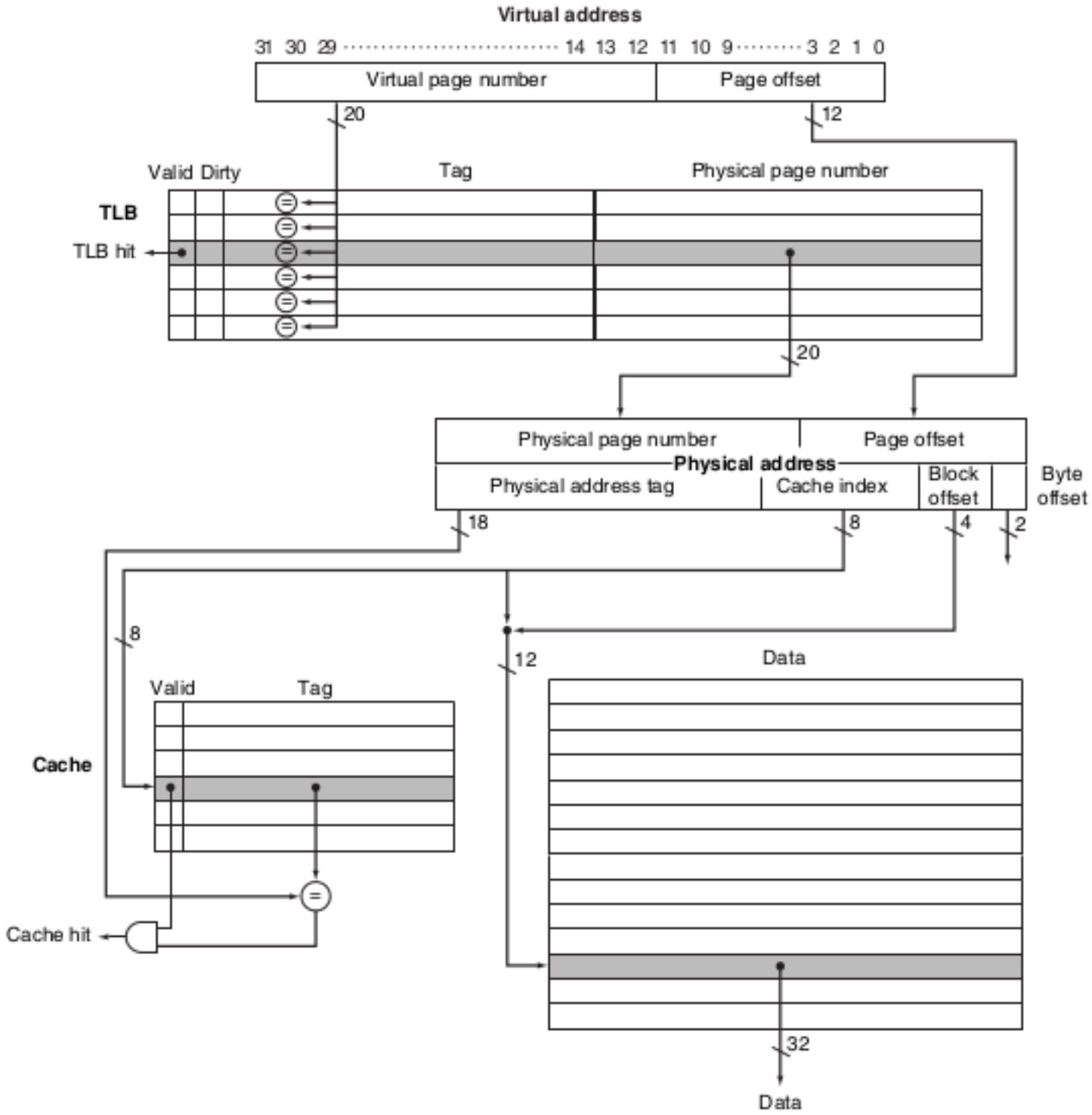
True page fault:

* If the page is not present in memory, then the TLB miss indicates a true page fault.

* In this case, the processor invokes the operating system using an exception Because the TLB has many fewer entries than the no.of pages in main memory

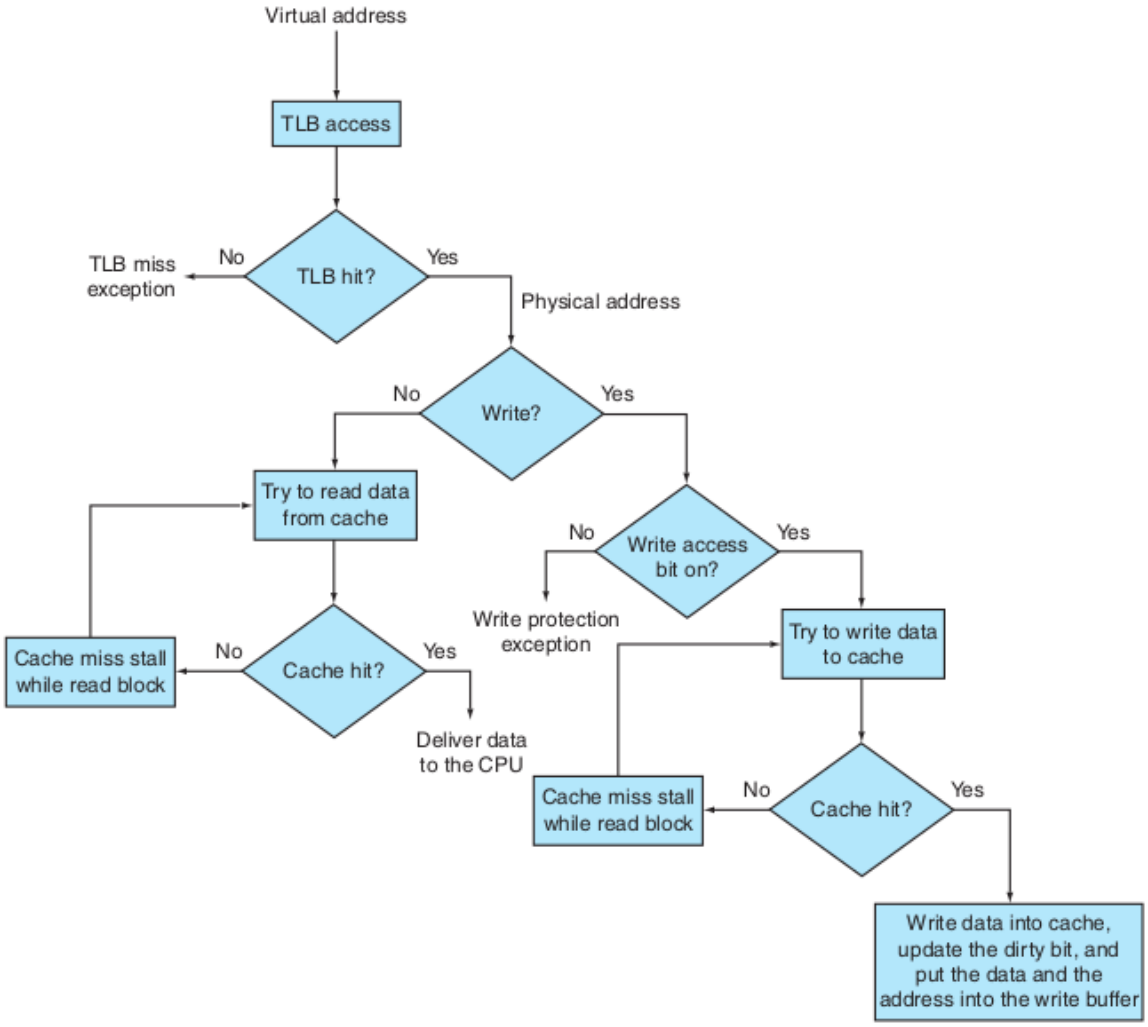
* After, a TLB miss occurs and the missing translation has been retrieved from the page table, we will need to select a TLB entry to replace

* Some systems use other techniques to approximate the reference and dirty bits eliminating the need to write into the TLB except to load a new table entry on a miss


Virtual address
31 30 29 ····· 14 13 12 11 10 9 ····· 3 2 1 0
Virtual page number
Page offset
20
12
Valid Dirty
Tag
Physical page number
TLB
TLB hit
20
Physical page number
Page offset
Physical address
Physical address tag
Cache index
Block offset
Byte offset
18
8
4
2
8
12
Data
Valid
Tag
Cache
Cache hit
32
Data

* virtually addressed cache a cache that is accessed with a virtual address rather than a physical address

* Physically addressed cache a cache that is addressed by a physical address



TLB	Page table	Cache	Possible? If so, under what circumstance?
Hit	Hit	Miss	Possible, although the page table is never really checked if TLB hits.
Miss	Hit	Hit	TLB misses, but entry found in page table; after retry, data is found in cache.
Miss	Hit	Miss	TLB misses, but entry found in page table; after retry, data misses in cache.
Miss	Miss	Miss	TLB misses and is followed by a page fault; after retry, data must miss in cache.
Hit	Miss	Miss	Impossible: cannot have a translation in TLB if page is not present in memory.
Hit	Miss	Hit	Impossible: cannot have a translation in TLB if page is not present in memory.
Miss	Miss	Hit	Impossible: data cannot be allowed in cache if the page is not in memory.

FIGURE The possible combinations of events in the TLB, virtual memory system, and cache. Three of these combinations are impossible, and one is possible (TLB hit, virtual memory hit, cache miss) but never detected.

DIRECT MEMORY ACCESS (DMA)

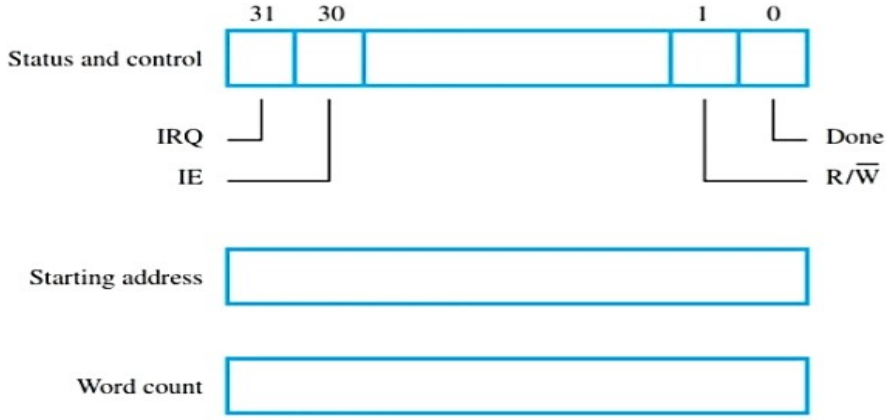
- A direct memory access (DMA) is an operation in which data is copied (transported) from one resource to another resource in a computer system without the involvement of the CPU.
- To copy data from HDD to pen drive, CPU is not necessary
- CPU is general purpose processor and has lot of work has to be done by CPU in a computer system.
- DMA is speed up the memory operations
- DMA reduces the CPU interaction in data transfer, so CPU utilization will be high in the system

DMA Controller:

- The unit that controls DMA transfers is referred to as a DMA Controller
- DMAC is a controller (a chip) specially designed for Data transfer Invented by Intel.
- In DMA process, DMAC will take over the control of the Bus and become master of it until the transfer is completed or CPU revoke the grant of master of bus
  - I/O device request the DMAC (DRQ) to transfer data
  - DMAC request CPU to grant bus to use (HLQ - Hold request)
  - CPU grants the access to use bus by DMAC (HLDA - Hold Acknowledgement)

- To initiate the transfer of a block of words, the processor sends to the DMA controller the starting address, the number of words in the block, and the direction of the transfer.
- The DMA controller then proceeds to perform the requested operation.
- When the entire block has been transferred, it informs the processor by raising an interrupt signal.

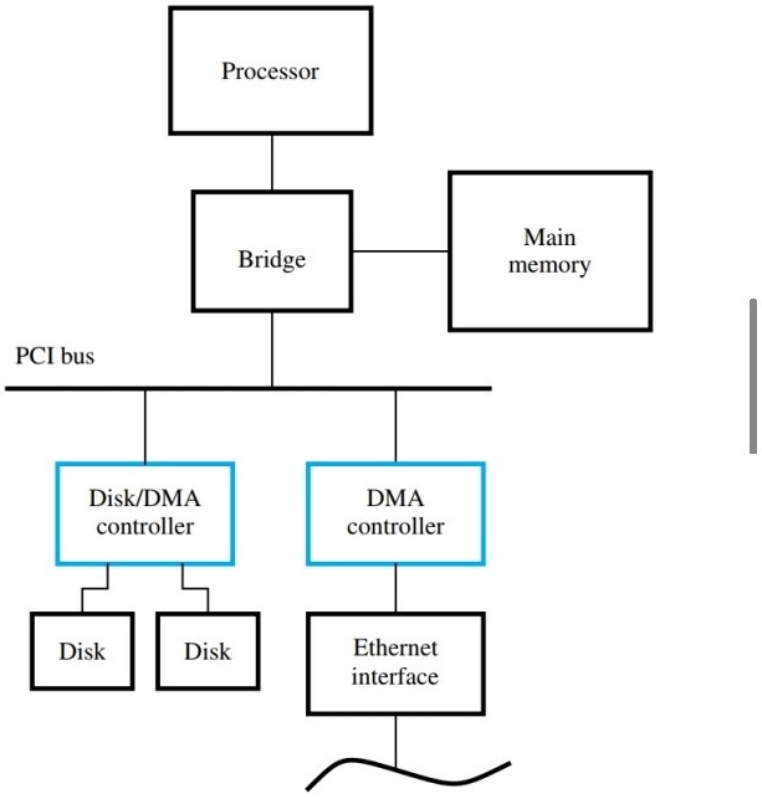
Registers in DMAC:



- DMA controller registers that are accessed by the processor to initiate the data transfer.
- Two registers are used for storing the starting address and the word count.
- The third register contains status and control flags.
- The R/W bit determines the direction of the transfer.
  - When this bit is set to 1, the controller performs a Read operation.
  - Otherwise, it performs a Write operation.
- Additional information is also transferred as may be required by I/O device.
- When the controller has completed transferring a block of data, it sets the Done flag to 1.
- Bit 30 is the Interrupt-enable flag, IE.

- When this flag is set to 1, it causes the controller to raise an interrupt after it has completed transferring a block of data.

- Finally, the controller sets the IRQ bit to 1 when it has requested an interrupt.



Use of DMA controllers in a computer system.

- One DMA controller connects a high-speed Ethernet to the computer's I/O bus.
- The disk controller, which controls two disks, also has DMA capability and provides two DMA channels. It can perform two independent DMA operations, as if each disk had its own DMA controller.

- To start DMA transfer of block of data from the main Memory to one of the disks, an OS routine write the address and word count info into the registers of the disk controllers.
- The DMA controller proceeds independently to implement the specified operation.
- When the transfer is completed, this fact is recorded in the status and control register of the DMA channel by setting the Done bit.
- At the same time, if the IE bit is set, the controller sends an interrupt request to the processor and sets the IRQ bit.
- The status register may also be used to record other information, such as whether the transfer took place currently or errors occurred.

The DMA Controller Transfers the Data in Three Modes:

1. Burst Mode
2. Cycle Stealing Mode
3. Transparent Mode

Burst Mode:

- Once the DMA controller gains the charge of the system bus, then it releases the system bus only after completion of data transfer.
- Till then the CPU has to wait for the system buses.

Cycle Stealing Mode:

- The DMA controller forces the CPU to stop its operation and relinquish the control over the bus for a short term to DMA controller.
- After the transfer of every byte, the DMA controller releases the bus and then again requests for the system bus.
- In this way, the DMA controller steals the clock cycle for transferring every byte.

Transparent Mode:

- The DMA controller takes the charge of system bus only if the processor does not require the system bus.

Storage Buffer:

- Most DMA controllers incorporate a data storage buffer.
- In the case of the network interface, the DMA controller reads a block of data from the main memory and stores it into its input buffer.
- This transfer takes place using burst mode at a speed appropriate to the memory and the computer bus.

  Then, the data in the buffer are transmitted over the network at the speed of the network.

Advantages:

- Transferring the data without the involvement of the processor will speed up the read-write task.
- DMA reduces the clock cycle requires to read or write a block of data.
- Implementing DMA also reduces the overhead of the processor.

Disadvantages:

- As it is a hardware unit, it would cost to implement a DMA controller in the system.
- Cache coherence problem can occur while using DMA controller.

Arbitration:

- A conflict may arise if both the processor and a DMA controller or two DMA controllers try to use the bus at the same time to access the main memory.
- To resolve these conflicts, an arbitration procedure is implemented on the bus.

Bus Arbitration

→A device that initiates data transfers on the bus at any given time is called a bus master.

→Bus arbitration is a process by which next device becomes the bus controller by transferring bus mastership to another bus

Bus arbitration schemes usually try to balance two factors:

▪ Bus priority: the highest priority device should be serviced first

• Fairness: Even the lowest priority device should never be completely locked out from the bus

Types of Bus Arbitration:

i)Centralized Arbitration

ii)Distributed Arbitration

Centralized Arbitration:

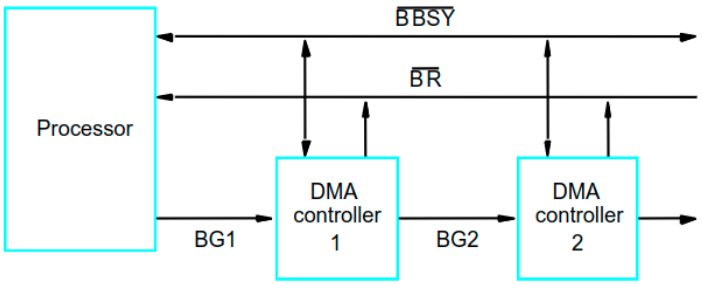
- A single bus arbiter performs the required arbitration.

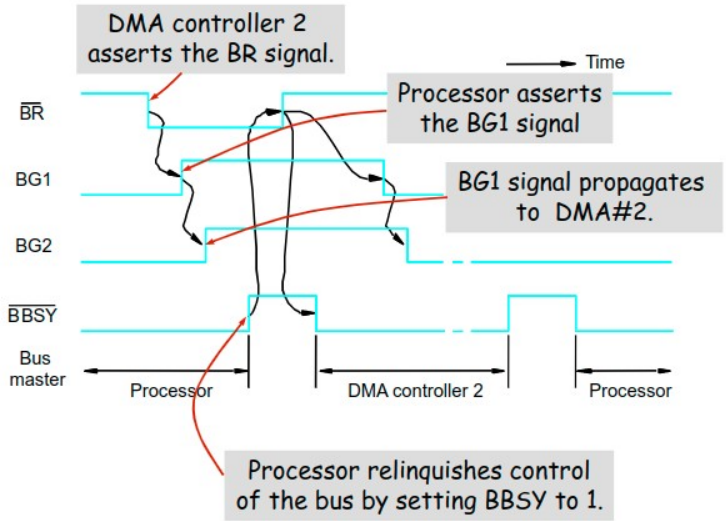
- The bus arbiter may be the processor or a separate controller connected to the bus.
- There are three different arbitration schemes that use the centralized bus arbitration approach
  1. Daisy Chaining Method
  2. Centralized Bus Arbitration Polling or Fixed Priority or Rotating Priority Method
  3. Independent Request Method

- A DMA controller indicates that it needs to become the bus master by activating the Bus-Request line, BR.
- When Bus-Request is activated, the processor activates the Bus-Grant signal, BGI, indicating to the DMA controllers.
- This signal is connected to all DMA controllers using a daisychain arrangement.

- If DMA controller 1 is requesting the bus, it blocks the propagation of the grant signal to other devices.
- Otherwise, it passes the grant downstream by asserting BG2.

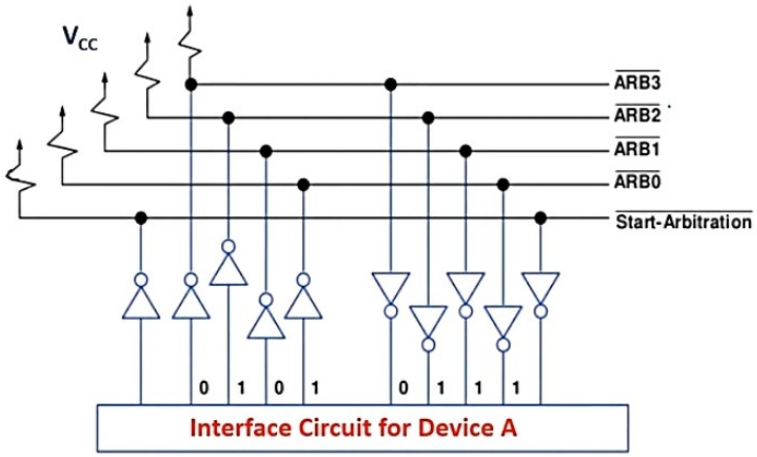
- The current bus master indicates to all devices that it is using the bus by activating another open-collector line called BusBusy BBSY.
- During its tenure as the bus master, it may perform one or more data transfer operations. After it releases the bus, the processor resumes bus mastership.
- The arbiter circuit ensures that only one request is granted at any given time, according to a predefined priority scheme. Alternatively, a rotating priority scheme may be used to give all devices an equal chance of being serviced.





Distributed arbitration:

- All devices waiting to use the bus share the responsibility of carrying out the arbitration process
- Arbitration process does not depend on a central arbiter and hence distributed arbitration has higher reliability.
- Each device is assigned a 4-bit ID number All the devices are connected using 5 lines, 4 arbitration lines to transmit the ID, and one line for the Start-Arbitration signal
- A winner is selected as a result of the interaction among the signals transmitted over these lines by all contenders. The net outcome is that the code on the four lines represents the request that has the highest ID number.
- if the input to one driver is equal to one and the input to another driver connected to the same bus line is equal to 0 the bus will be in the low-voltage state. In other words, the connection performs an OR function in which logic I wins.

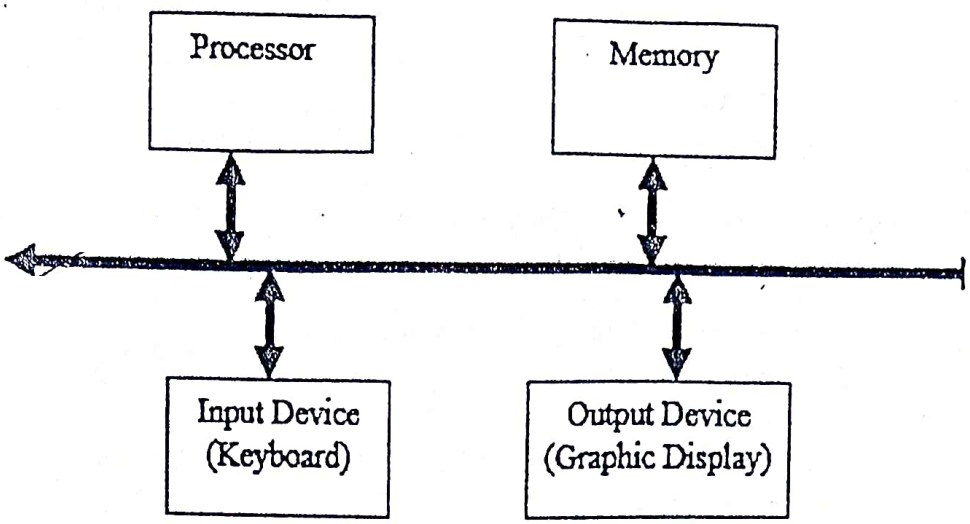


- Assume that two devices, A and B, having ID numbers 5 and 6. respectively. are requesting the use of the bus.
- Device A transmits the pattern 0101, and device B transmits the pattern 0110.
- The code seen by both devices is 0111. Each device compare the pattern on the arbitration lines to its own ID, starting from the most significant be If it detects a difference at any bit position, it disables its drivers at that bit position and for all lower-order bits.
- It does so by placing a 0 at the input of these drivers.
- In the case of our example, device A detects a difference on line ARB1. Hence, it disable its drivers on lines ARBI and ARBO.
- This causes the pattern on the arbitration lines to change to 0110, which means that B has won the contention.
- Note that, since the code on the priority lines is 0111 for a short period, device B may temporarily disable driver on line ARBO.
- However, it will enable this driver again once it sees a 0 on line ARBI resulting from the action by device A.

- Decentralized arbitration has the advantage of offering higher reliability, because operation of the bus is not dependent on any single device.
- Many schemes have been proposed and used in practice to implement distributed arbitration.

Input/Output System

- Important components of any computer system are
  - CPU
  - Memory
  - I/O devices (Peripherals).
- CPU fetches instructions from memory, process them and stores the result in memory
- The other components of the computer system (I/O devices) called the **Input/output system**
- Main function of I/O System
  - To transfer information between CPU and the outside world
- I/O devices cannot directly connected to the system bus because of the following reasons
  - Each I/O devices have different methods of operation
    - So it is difficult to incorporate the logic within CPU
  - Data transfer rate of I/O device is much slower than that of the memory
    - So it is not possible to use high speed system bus to communicate directly with I/O devices
  - I/O device used in computer system have different data formats and word length than that of CPU.
- To overcome the difficulties, to use modules in between the system bus and Peripherals called I/O module or I/O system or I/O interface.



Constraint Types

Two primary types of specification while designing a I/O system

1. Latency or response constraints
2. Bandwidth or throughput constraints

- Latency constraints
  - Total elapsed time to accomplish an input or output operation
- Bandwidth constraints
  - Amount of information communicated across an interconnect to the processor/memory (I/O device) per unit time

Accessing I/O devices

- I/O devices can be connected to a computer through a single bus which enables the exchange of information.
- The bus consists of three sets of lines used to carry address, data, and control signals.
- Each I/O device is assigned a unique set of addresses.
- When the processor places a particular address on the address lines, the device that recognizes this address responds to the commands issued on the control lines.
- The processor requests either a read or a write operation, and the requested data are transferred over the data lines.

Single Bus Structure

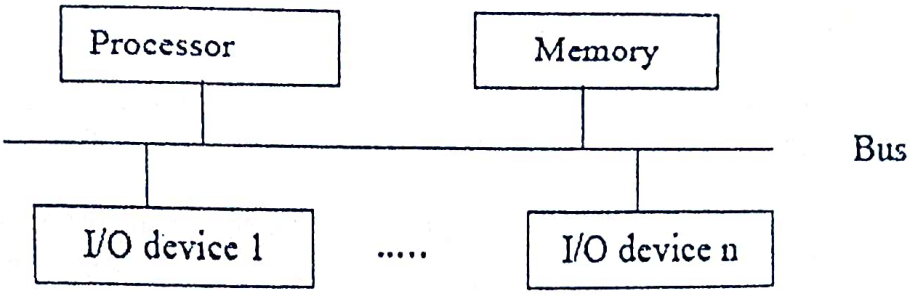


Fig 5.1 Single Bus Structure

Memory-mapped I/O

- When I/O devices & memory share the same address space, the arrangement is called memory mapped I/O.
- With memory-mapped I/O, any machine instruction that can access memory can be used to transfer data to or from an I/O device.

Program controlled I/O

- Program controlled I/O is one in which the processor repeatedly checks a status flag to achieve the required synchronization between Processor & I/O device.
- The processor polls the device.
- There are 2 mechanisms to handle I/O operations. They are,
  - **Interrupt** → Synchronization is achieved by having the I/O device send a special signal over the bus whenever it is ready for a data transfer operation
  - **DMA** (Direct Memory Access) → It is a technique used for high speed I/O device. Here, the input device transfers data directly to or from the memory without continuous involvement by the processor.

DATA TRANSFER (I/O) TECHNIQUES

- To transfer data, the system requires external devices and processor.

Different types of I/O data transfer

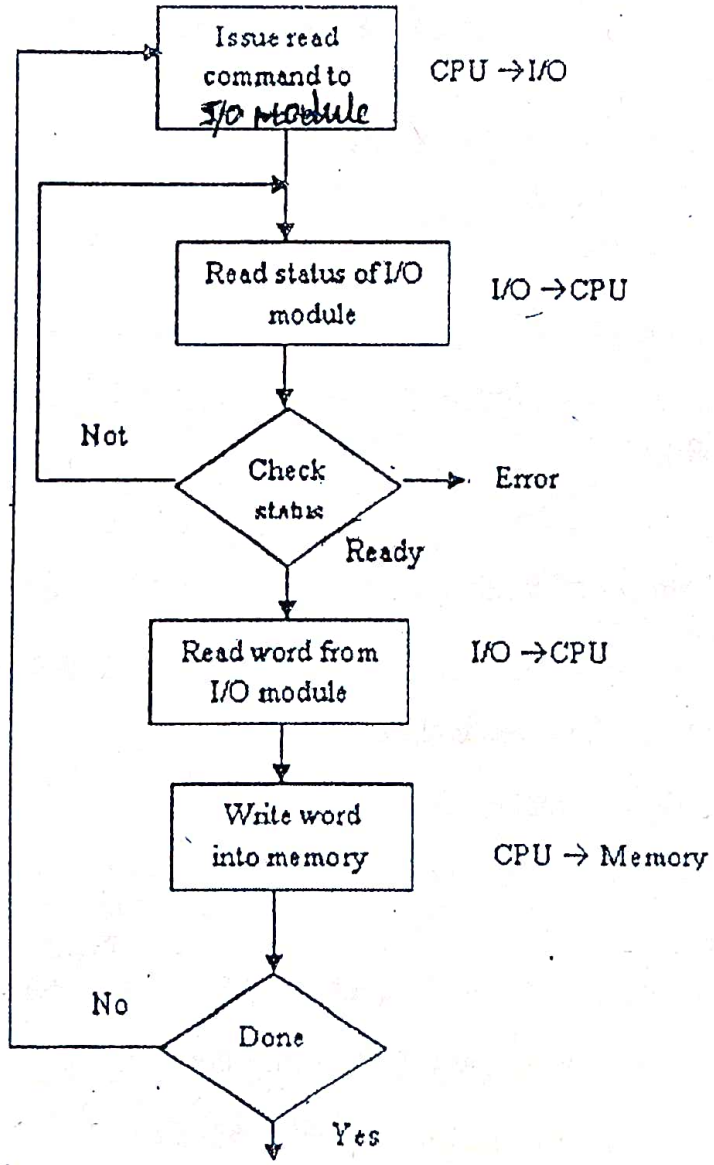
1. Programmed I/O
2. Interrupt
3. Direct Memory Access (DMA)

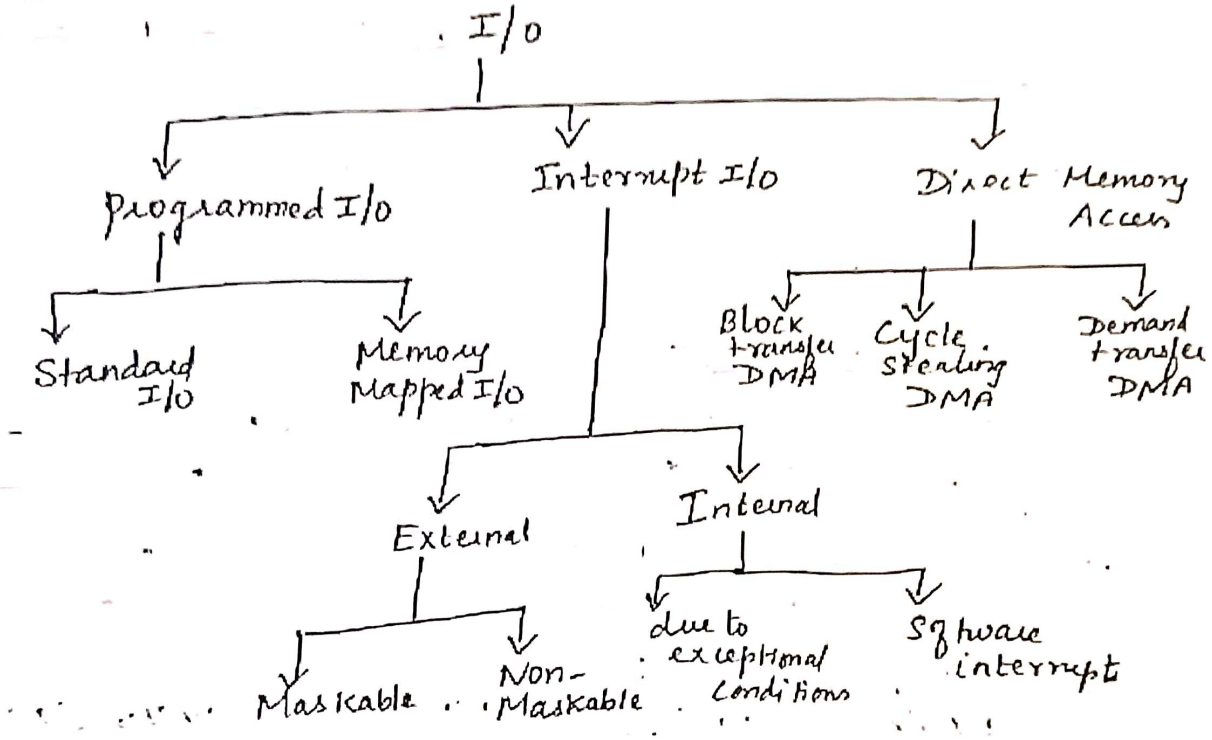
Programmed I/O

- I/O operations are directly controlled by the processor, then the system is said to be Programmed I/O

- Data item transfer is initiated by an instruction in the program
- Transferring data under program control requires the constant monitoring of the peripheral by the CPU
- Transfer a data between I/O device and memory or between I/O device and the processor
- I/O device does not have direct access to memory
- Processor executes a program that initiates, direct, and terminate I/O operations
- Process includes
    - Sensing device status
    - Sending read and write commands
    - Transferring the data
- Processor periodically checks the status of the I/O system until the operation is completed.

Flowchart





5.6.3 Basic operation

- CPU requests I/O operation
- I/O module performs read operation
- Check the status bits periodically
- Read the data from I/O module
- Transfer the data to memory
- Check if all the data words are transferred if not go to step 1 otherwise continue next operation

PARALLEL AND SERIAL INTERFACE

Interface Circuits:

- The I/O interface of a device consists of the circuitry needed to connect that device to the bus. On one side of the interface are the bus lines for address, data, and control.
- On the other side are the connections needed to transfer data between the interface and the I/O device.
- This side is called a *port*, and it can be either a parallel or a serial port.

An I/O interface does the following:

1. Provides a register for temporary storage of data
2. Includes a status register containing status information that can be accessed by the processor
3. Includes a control register that holds the information governing the behaviour of the interface
4. Contains address-decoding circuitry to determine when it is being addressed by the processor
5. Generates the required timing signals
6. Performs any format conversion that may be necessary to transfer data between the processor and the I/O device, such as parallel-to-serial conversion in the case of a serial port

Parallel Interface:

- A typical keyboard consists of mechanical switches that are normally open.
- When a key is pressed, its switch closes and establishes a path for an electrical signal.
- This signal is detected by an encoder circuit that generates the ASCII code for the corresponding character.

- A difficulty with such mechanical pushbutton switches is that the contacts *bounce* when a key is pressed, resulting in the electrical connection being made then broken several times before the switch settles in the closed position.
- Although bouncing may last only one or two milliseconds, this is long enough for the computer to erroneously interpret a single pressing of a key as the key being pressed and released several times.
- The effect of bouncing can be eliminated using a simple debouncing circuit.

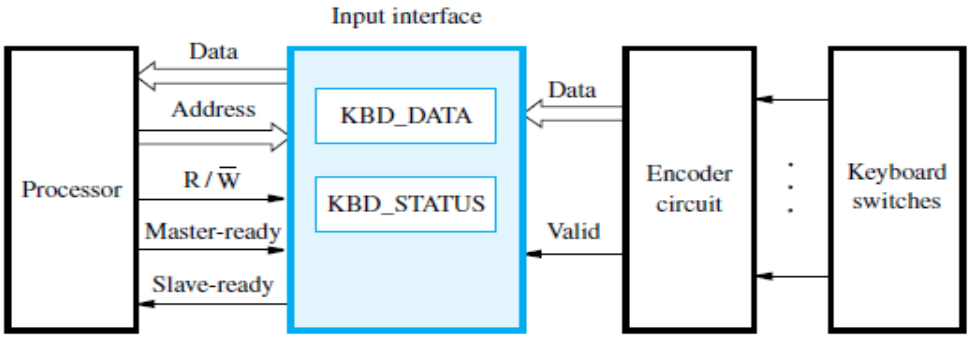
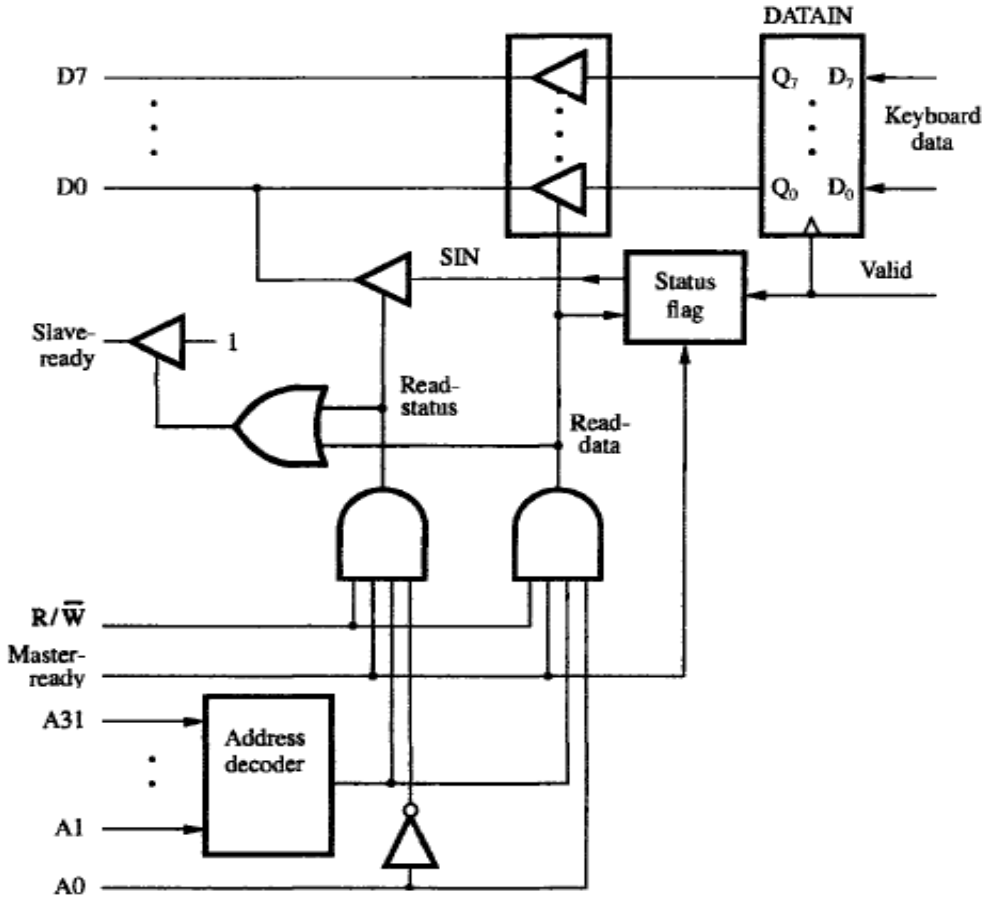


Figure 7.10: Keyboard to Processor connection

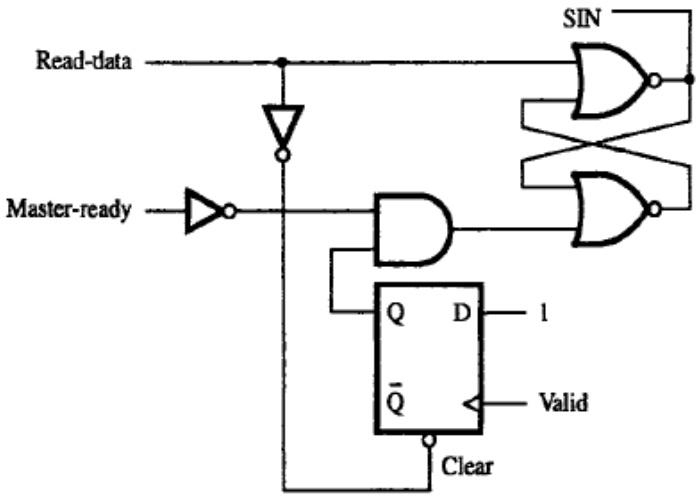
- The output of the encoder consists of one byte of data representing the encoded character and one control signal called Valid.
- When a key is pressed, the Valid signal changes from 0 to 1, causing the ASCII code of the corresponding character to be loaded into the KBD_DATA register and the status flag KIN to be set to 1.



Input-interface-circuit

- The interface circuit connected to an asynchronous bus on which transfers are controlled by the handshake signals Master-ready and Slave-ready.

Implementation of the status flag circuit



- The KIN flag is the output of a NOR latch connected
- A flip-flop is set to 1 by the rising edge on the Valid signal line.
- This event changes the state of the NOR latch to set KIN to 1, but only when Master-ready is low.

- The reason for this additional condition is to ensure that KIN does not change state while being read by the processor.
- Both the flip-flop and the latch are reset to 0 when Read-data becomes equal to 1, indicating that KBD_DATA is being read.

- A designer using modern computer aided design tools would specify these functions using a hardware description language such as VHDL or Verilog.
- The resulting circuits would depend on the technology used and may or may not be the same as the circuits shown in these figures.

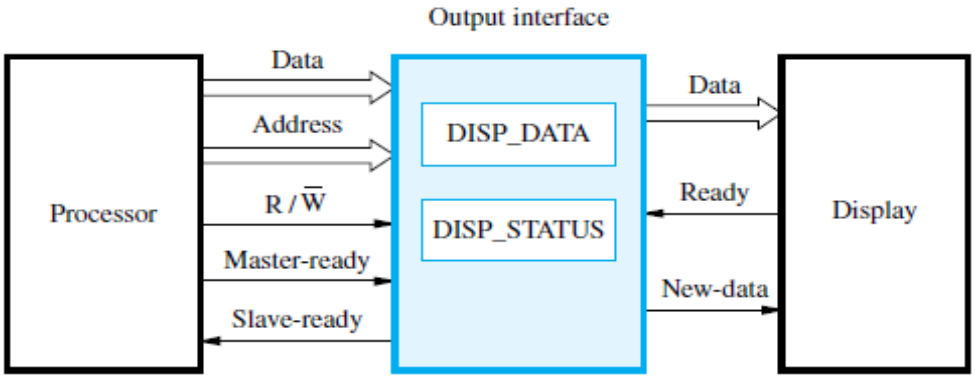
Output Interface:

- used to connect an output device such as a display.
- Assume that the display uses two handshake signals, New-data and Ready, in a manner similar to the handshake between the bus signals Master-ready and Slave-ready.

- When the display is ready to accept a character, it asserts its Ready signal, which causes the DOUT flag in the DISP_STATUS register to be set to 1.
- When the I/O routine checks DOUT and finds it equal to 1, it sends a character to DISP_DATA.
- This clears the DOUT flag to 0 and sets the New-data signal to 1.

- In response, the display returns Ready to 0 and accepts and displays the character in DISP_DATA.

- When it is ready to receive another character, it asserts Ready again, and the cycle repeats.



Display to processor connection.

Serial Interface:

- A serial interface is used to connect the processor to I/O devices that transmit data one bit at a time.
- Data are transferred in a bit-serial fashion on the device side and in a bit-parallel fashion on the processor side.
- The transformation between the parallel and serial formats is achieved with shift registers that have parallel access capability.

- The input shift register accepts bit-serial input from the I/O device.
- When all 8 bits of data have been received, the contents of this shift register are loaded in parallel into the DATAIN register.

- Similarly, output data in the DATAOUT register are transferred to the output shift register, from which the bits are shifted out and sent to the I/O device.

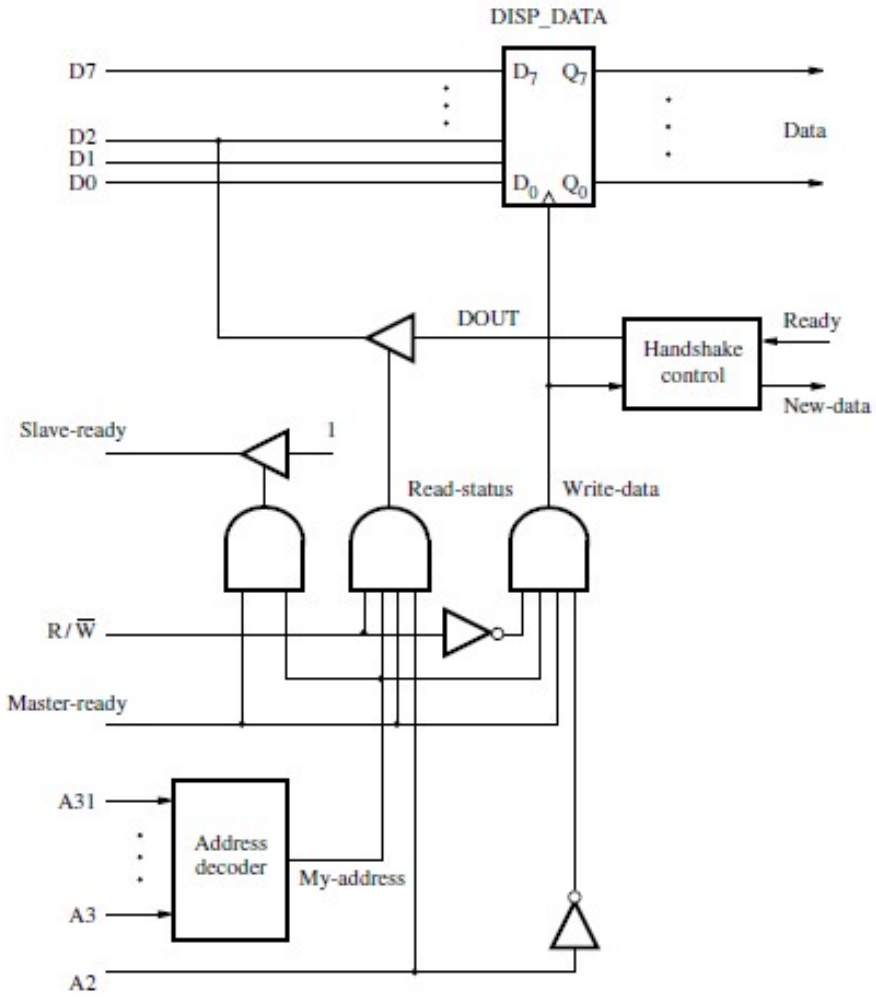


Figure 7.14 An output interface circuit.

- Two status flags, which refers to as SIN and SOUT, are maintained by the Status and control block.
- The SIN flag is set to 1 when new data are loaded into DATAIN from the shift register, and cleared to 0 when these data are read by the processor.
- The SOUT flag indicates whether the DATAOUT register is available.
- It is cleared to 0 when the processor writes new data into DATAOUT and set to 1 when data are transferred from DATAOUT to the output shift register.

- The double buffering used in the input and output paths
- It is possible to implement DATAIN and DATAOUT themselves as shift registers, thus obviating the need for separate shift registers.
- After receiving one character from the serial line, the interface would not be able to start receiving the next character until the processor reads the contents of DATAIN.
- Thus, a pause would be needed between two characters to give the processor time to read the input data.

- With double buffering, the transfer of the second character can begin as soon as the first character is loaded from the shift register into the DATAIN register.
- Thus, provided the processor reads the contents of DATAIN before the serial transfer of the second character is completed, the interface can receive a continuous stream of input data over the serial line.
- An analogous situation occurs in the output path of the interface.

- During serial transmission, the receiver needs to know when to shift each bit into its input shift register.
- Since there is no separate line to carry a clock signal from the transmitter to the receiver, the timing information needed must be embedded into the transmitted data using an encoding scheme.

- There are two basic approaches.
- The first is known as asynchronous transmission, because the receiver uses a clock that is not synchronized with the transmitter clock.
- In the second approach, the receiver is able to generate a clock that is synchronized with the transmitter clock

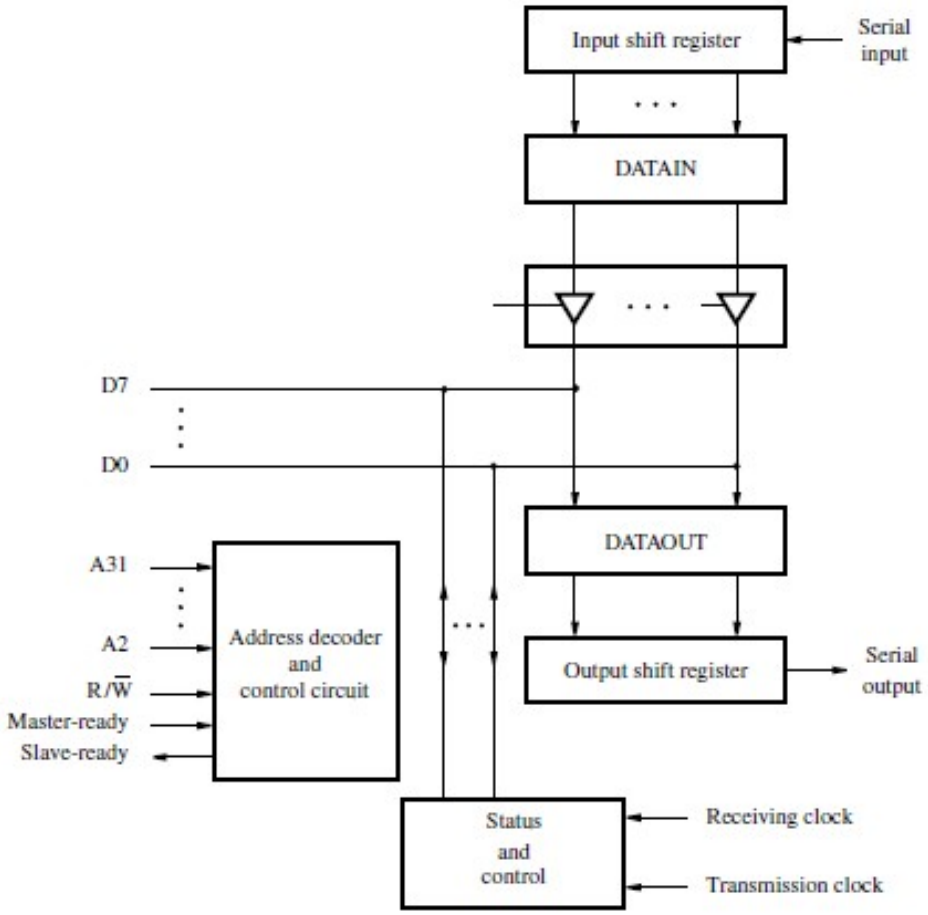


Figure 7.15 A serial interface.

Asynchronous Transmission:

- This approach uses a technique called *start-stop* transmission.

- Data are organized in small groups of 6 to 8 bits, with a well-defined beginning and end. In a typical arrangement, alphanumeric characters encoded in 8 bits are transmitted
- The line connecting the transmitter and the receiver is in the 1 state when idle.
- A character is transmitted as a 0 bit, referred to as the Start bit, followed by 8 data bits and 1 or 2 Stop bits.
- The Stop bits have a logic value of 1.
- The 1-to-0 transition at the beginning of the Start bit alerts the receiver that data transmission is about to begin.
- Using its own clock, the receiver determines the position of the next 8 bits, which it loads into its input register.
- The Stop bits following the transmitted character, which are equal to 1, ensure that the Start bit of the next character will be recognized.
- When transmission stops, the line remains in the 1 state until another character is transmitted.

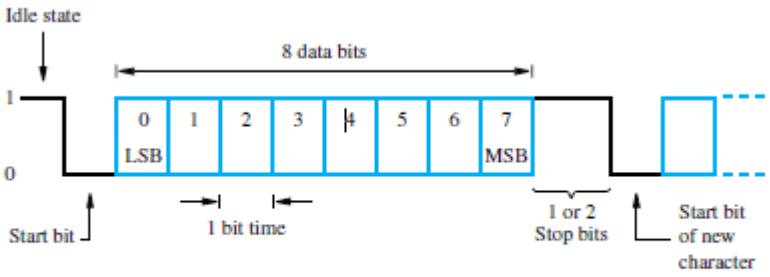


Figure 7.16 Asynchronous serial character transmission.

Synchronous Transmission:

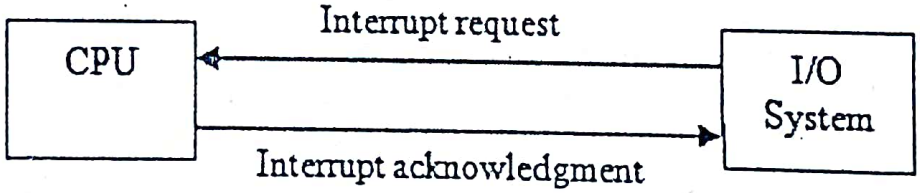
- In the start-stop scheme, the position of the 1-to-0 transition at the beginning of the start bit is the key to obtaining correct timing information.
- This scheme is useful only where the speed of transmission is sufficiently low and the conditions on the transmission link are such that the square waveforms shown in the figure maintain their shape.
- For higher speed a more reliable method is needed for the receiver to recover the timing information.

- Encoded data are usually transmitted in large blocks consisting of several hundreds or several thousands of bits.
- The beginning and end of each block are marked by appropriate codes, and data within a block are organized according to an agreed upon set of rules. Synchronous transmission enables very high data transfer rates

INTERRUPT

Definition

- External event that affects the normal flow of instruction execution generated by the external hardware devices such as keyboard, mouse, disk drives, scanner, and printer.
- Example
    - Computer system should give response to devices such as keyboard, mouse when they request for service
- If a device wants to notify the processor about the completion of operation by sending a hardware signal called interrupt
- Interrupt Request line is used to alert the processor
- Interrupt Service Routine
    - The processor provides the requested service called the Interrupt Service Routine (ISR).
- Processor acknowledges the interrupt by using interrupt acknowledgement signal

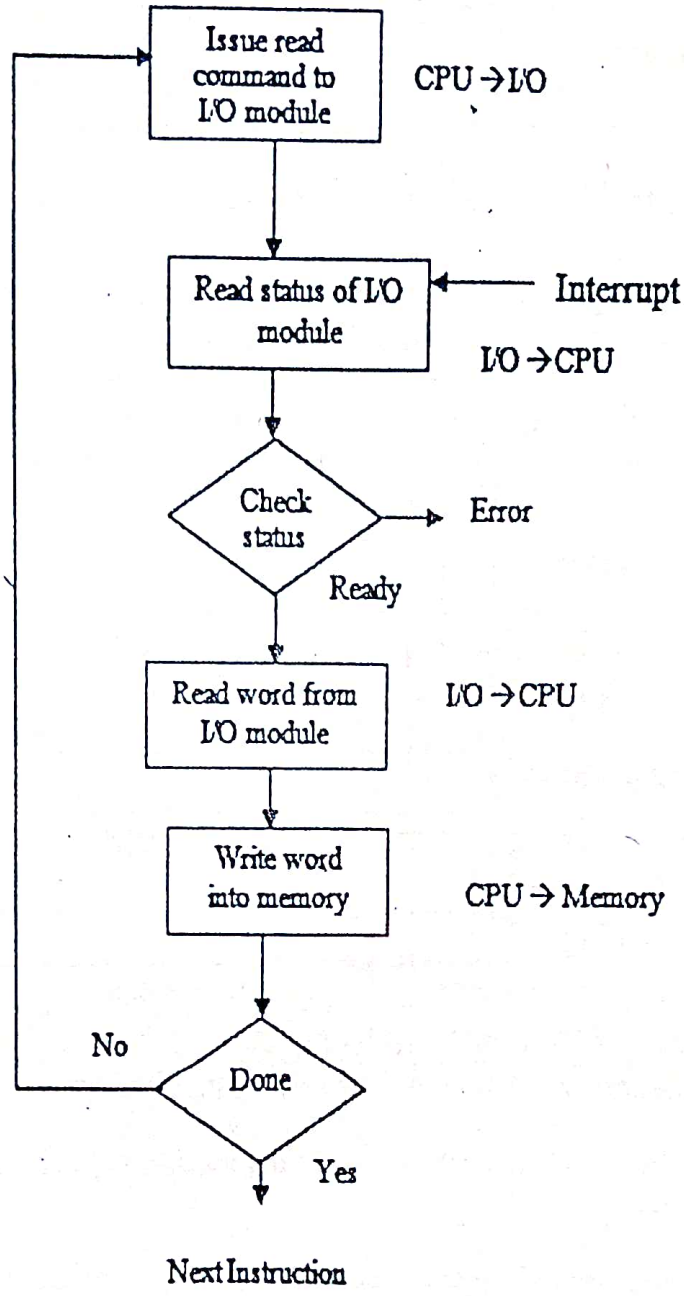


- I/O system gains control of the bus
- I/O system sends interrupt request
- The processor acknowledges the interrupt request

.1 Types of interrupt

- Hardware interrupt
    - Interrupt caused by an external signal is referred as Hardware interrupt
- Software interrupt
    - Interrupt caused by special instruction are called Software interrupt

.2 Flowchart



.3 Basic operation

.3.1 I/O module view point

- I/O module receives a READ command form the processor
- I/O module reads data from desired peripheral into data register
- I/O module interrupts the processor
- I/O module waits until data is requested by the processor
- I/O module places data on the data bus when requested

3.2 Processor view point

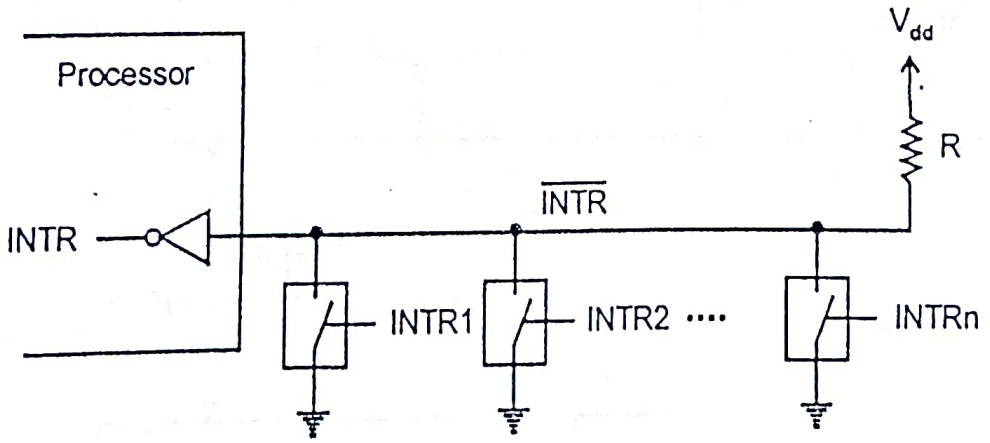
- The processor issues a READ command
- The processor performs some other useful work
- The processor checks for interrupts at the end of the instruction cycle
- The processor saves the current context when interrupted by the I/O module
- The processor read the data from the I/O module and stores it in memory
- The processor the restores the saved context and resumes execution

.4 Interrupt hardware

- I/O device request an interrupt by activating a bus line called interrupt-request
- A single interrupt request line may be used to serve n devices
- All devices are connected to interrupt request line via switches to ground
- If all interrupt request lines of I/O devices are inactive, then interrupt request line will be equal to V_{dd}

$$INTR = INTR_1 + INTR_2 + \ldots\ldots\ldots + INTRn$$

Common Interrupt Request Line



.5 Enabling and disabling interrupts

- The arrival of an interrupt request from an external device causes the processor to suspend the execution of one program and start the execution of another program
- INTERRUPT signal is active during the execution of **Interrupt Service Routine.**
- The sequence of events in handling interrupt can be summarized as follows
  - *The device raises an interrupt request.*

- The processor interrupts the program currently being executed.
- Interrupts are disabled by changing the control bits is PS (Processor Status register)
- The action requested by the interrupt is performed by the interrupt-service routine
- The device is informed that its request has been recognized and in response, it deactivates the INTERRUPT signal.
- Interrupts are enabled and execution of the interrupted program is resumed.

6 Exceptions

- An interrupt is an event that stops execution of one program and start execute another program
- Also referred as interruption
- Example
  - I/O interrupt

Types of exception

1. Recovery from errors
2. Debugging
3. Privileged Exception

.6.1 Recovery from Errors

- Computers have error-checking code in Main Memory, which allows detection of errors in the stored data.
- If an error occurs, the control hardware detects and informs the processor by raising an interrupt.
- The processor also interrupts the program, if it detects an error or an unusual condition while executing the instance (ie) it suspends the program being executed and starts an execution service routine.
- It takes appropriate action to recover from the error.

.6.2 Debugging

- System software has a program called debugger, which find errors in a program

- The debugger provides two important facilities
    1. Trace
    2. Breakpoint

Trace Mode

- When processor is in trace mode, an exception occurs after execution of every instance using the debugging program as the exception service routine.
- The debugging program examine the contents of registers, memory location etc.
- On return from the debugging program the next instance in the program being debugged is executed
- The trace exception is disabled during the execution of the debugging program.

Break point

- Program being debugged is interrupted only at specific points selected by the user
- An instance called the Trap (or) software interrupt is usually provided for this purpose
- While debugging the user may interrupt the program execution after instance
- When the program is executed and reaches that point it examine the memory and register contents

6.3 Privileged Exception

- To protect the OS of a computer from being corrupted by user program certain instance can be executed only when the processor is in supervisor mode called privileged exceptions.
- When the processor is in user mode, it will not execute instance (ie) when the processor is in supervisor mode, it will execute instance.

Interconnection Standards

- A typical desktop or notebook computer has several ports that can be used to connect I/Odevices, such as a mouse, a memory key, or a disk drive.
- Standard interfaces have been developed to enable I/O devices to use interfaces that are independent of any particular processor.

- A memory key that has a USB connector can be used with any computer that has a USB port.

Universal Serial Bus (USB)

- The Universal Serial Bus (USB) is the most widely used interconnection standard.
- A large variety of devices are available with a USB connector, including mice, memory keys, disk drives, printers, cameras, and many more.
- The success of the USB is due to its simplicity and low cost.
- The original USB specification supports two speeds of operation, called low-speed (1.5 Megabits/s) and full-speed (12 Megabits/s).

- USB 2, called High-Speed USB, was introduced.
- It enables data transfers at speeds up to 480 Megabits/s.

- As I/O devices continued to evolve with even higher speed requirements, USB 3 (called Superspeed) was developed.
- It supports data transfer rates up to 5 Gigabits/s.

- Key objectives:
  1. Provide a simple, low-cost, and easy to use interconnection system
  2. Accommodate a wide range of I/O devices and bit rates, including Internet connections, and audio and video applications
  3. Enhance user convenience through a “plug-and-play” mode of operation

Device Characteristics

- The kinds of devices that may be connected to a computer cover a wide range of functionality.
- The speed, volume, and timing constraints associated with data transfers to and from these devices vary significantly.
- The sampling process yields a continuous stream of digitized samples that arrive at regular intervals, synchronized with the sampling clock. Such a data stream is called isochronous, meaning that successive events are separated by equal periods of time. A signal must be sampled quickly enough to track its highest-frequency components.
- Data transfers for images and video have similar requirements, but require much higher data transfer rates. To maintain the picture quality of commercial television, an image should be represented by about 160 kilobytes and transmitted 30 times per second. Together with control information, this yields a total bit rate of 44 Megabits/s. Higher-quality images, as in HDTV (High Definition TV), require higher rates.

Plug-and-Play

- The USB standard defines both the USB hardware and the software that communicates with it.
- Its plug-and-play feature means that when a new device is connected, the system detects its existence automatically.
- The software determines the kind of device and how to communicate with it, as well as any special requirements it might have.
- As a result, the user simply plugs in a USB device and begins to use it, without having to get involved in any of these details.
- The USB is also hot-pluggable, which means a device can be plugged into or removed from a USB port while power is turned on.

USB Architecture

- The USB uses point-to-point connections and a serial transmission format.
- When multiple devices are connected, they are arranged in a tree structure

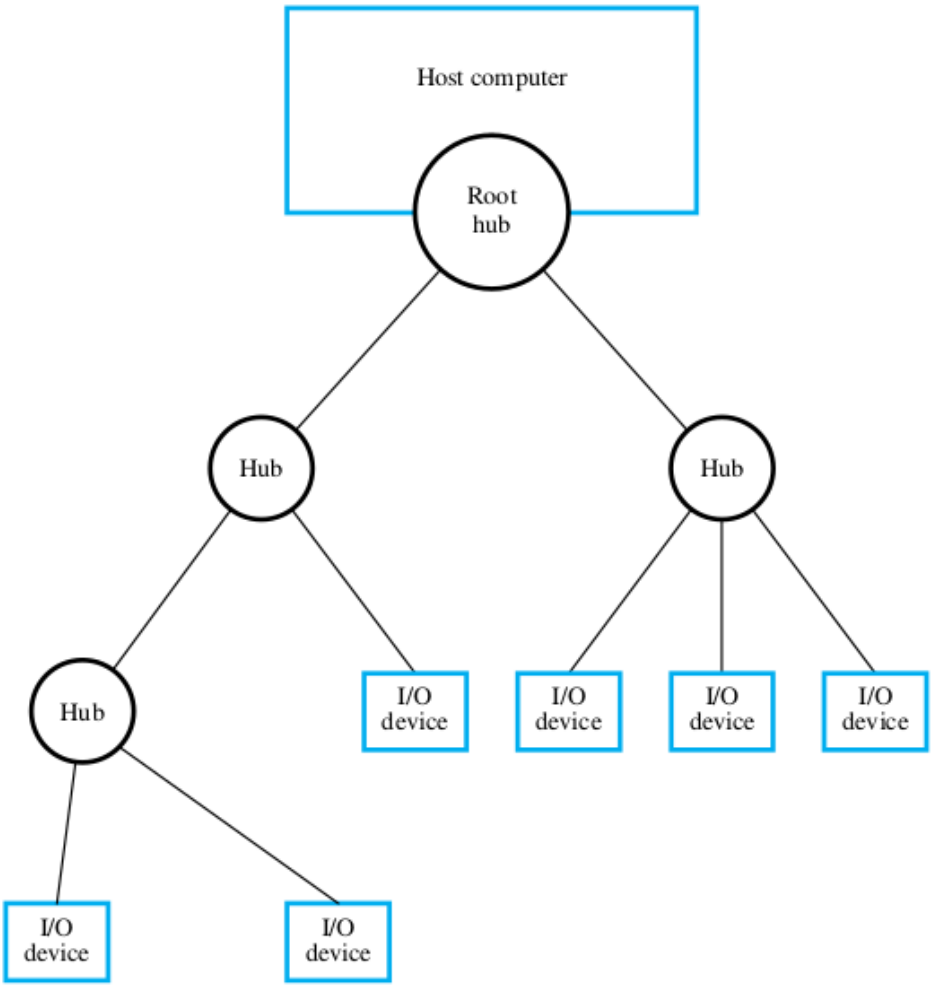


Figure 7.17 Universal Serial Bus tree structure.

- Each node of the tree has a device called a hub, which acts as an intermediate transfer point between the host computer and the I/O devices.
- At the root of the tree, a root hub connects the entire tree to the host computer.
- The leaves of the tree are the I/O devices: a mouse, a keyboard, a printer, an Internet connection, a camera, or a speaker.
- The tree structure makes it possible to connect many devices using simple point-to-point serial links.

- Polling:
- If I/O devices are allowed to send messages at any time, two messages may reach the hub at the same time and interfere with each other.
- The USB operates strictly on the basis of polling.
- A device may send a message only in response to a poll message from the host processor.
- Hence, no two devices can send messages at the same time.
- This restriction allows hubs to be simple, low-cost devices.

- Address:
- Each device on the USB, whether it is a hub or an I/O device, is assigned a 7-bit address. This address is local to the USB tree and is not related in any way to the processor's address space.

- The root hub of the USB, which is attached to the processor, appears as a single device.
- The host software communicates with individual devices by sending information to the root hub, which it forwards to the appropriate device in the USB tree.

- Connection:
- When a device is first connected to a hub, or when it is powered on, it has the address 0.
- Periodically, the host polls each hub to collect status information and learn about new devices that may have been added or disconnected.
- When the host is informed that a new device has been connected, it reads the information in a special memory in the device's USB interface to learn about the device's capabilities.
- It then assigns the device a unique USB address and writes that address in one of the device's interface registers.
- It is this initial connection procedure that gives the USB its plug-and-play capability.

Isochronous Traffic on USB

- An important feature of the USB is its ability to support the transfer of isochronous data in a simple manner.
- isochronous data need to be transferred at precisely timed regular intervals.
- To accommodate this type of traffic, the root hub transmits a uniquely recognizable sequence of bits over the USB tree every millisecond. This sequence of bits, called a Start of Frame character, acts as a marker indicating the beginning of isochronous data, which are transmitted after this character.
- Thus, digitized audio and video signals can be transferred in a regular and precisely timed manner.

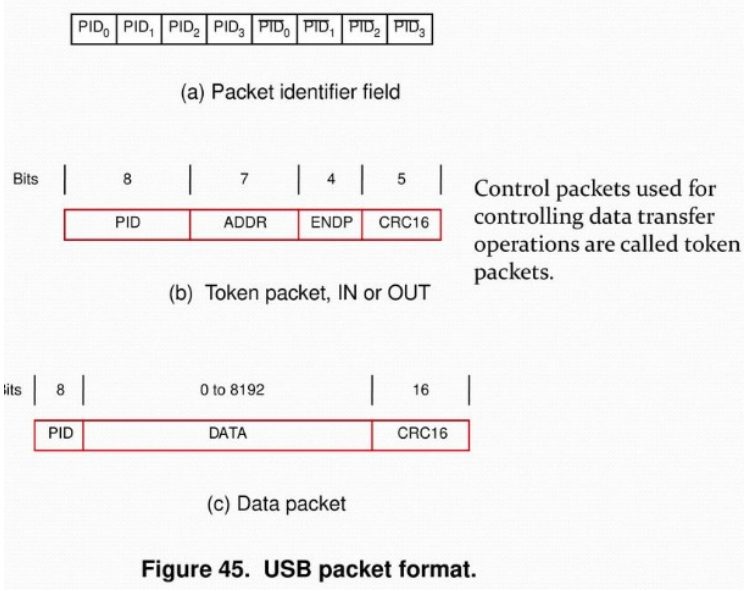


Figure 45. USB packet format.

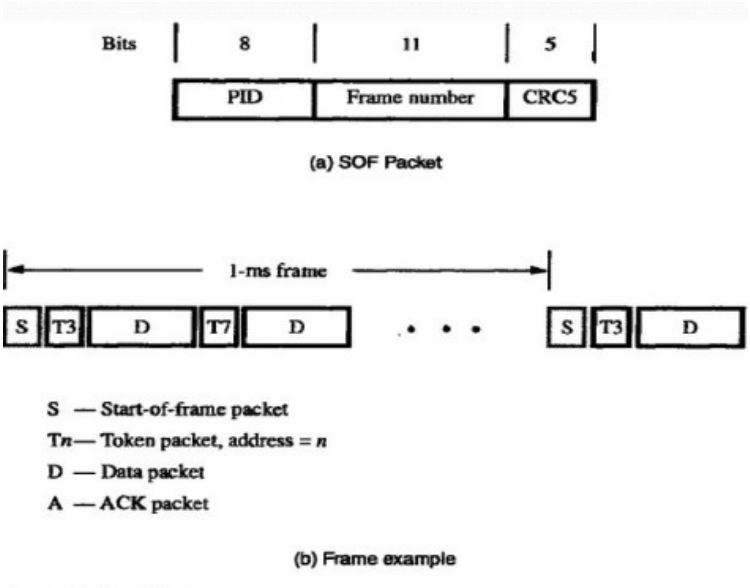


Figure 4.47 USB frames.

Electrical Characteristics

- USB connections consist of four wires, of which two carry power, +5 V and Ground, and two carry data.
- Thus, I/O devices that do not have large power requirements can be powered directly from the USB.

- Two methods are used to send data over a USB cable.
- When sending data at low speed, a high voltage relative to Ground is transmitted on one of the two data wires to represent a 0 and on the other to represent a 1.
- The Ground wire carries the return current in both cases.
- Such a scheme in which a signal is injected on a wire relative to ground is referred to as single-ended transmission.

- The speed at which data can be sent on any cable is limited by the amount of electrical noise present.
- The term noise refers to any signal that interferes with the desired data signal and hence could cause errors.

- Single-ended transmission is highly susceptible to noise.
- The voltage on the ground wire is common to all the devices connected to the computer.
- Signals sent by one device can cause small variations in the voltage on the ground wire, and hence can interfere with signals sent by another device. Interference can also be caused by one wire picking up noise from nearby wires.

- The High-Speed USB uses an alternative arrangement known as differential signaling.
- The data signal is injected between two data wires twisted together.
- The ground wire is not involved.
- The receiver senses the voltage difference between the two signal wires directly, without reference to ground.
- This arrangement is very effective in reducing the noise seen by the receiver, because any noise injected on one of the two wires of the twisted pair is also injected on the other.
- Since the receiver is sensitive only to the voltage difference between the two wires, the noise component is cancelled out.
- The ground wire acts as a shield for the data on the twisted pair against interference from nearby wires. Differential signaling allows much lower voltages and much higher speeds to be used compared to single-ended signaling.

SATA

- In the early days of the personal computer, the bus of a popular IBM computer called AT, which was based on Intel's 8080 microprocessor bus, became an industry standard.
- It was named ISA, for Industry Standard Architecture.
- An enhanced version, including a definition of the basic software needed to support disk drives, was later named ATA, for AT Attachment bus.
- A serial version of the same architecture became known as SATA, which is now widely used as an interface for disks.
- Like all standards, several versions of SATA have been developed with added features and higher speeds.
- The original parallel version has been renamed PATA, but it is no longer used in new equipment.

- The basic SATA connector has 7 pins, connecting two twisted pairs and three ground wires.
- Differential transmission is used, with clock frequencies ranging from 1.5 to 6.0 Gigabits/s.
- Some of the recent versions provide an isochronous transmission feature to support audio and video devices.

Features:

1. **Low Voltage Requirement:** SATA operates on 500mV (0.5V) peak-to-peak signaling. This help in promoting a much low interference and crosstalk between conductors.
2. **Hot Plugging:** This feature helps users to change or remove storage devices even when the computer is running.
3. **Staggered Spin-Up:** Allows sequential hard disk drive startup, which helps even out power load distribution during system booting.
4. **Native Command Queuing (NCQ):** Usually, the commands reach a disk for or writing from different locations on the disk. When the commands are carried out based on the order in which they appear, a substantial amount of mechanical overhead is generated because of the constant repositioning of the read/write head. SATA II drives use an algorithm to identify the most effective order to carry out commands. This helps to reduce mechanical overhead and improve performance.
5. **Port Multipliers:** Allows the connection of up to 15 drives to a SATA controller. This facilitates the building of disk enclosures.
6. **Port Selectors:** Facilitates redundancy for two hosts connected to a single drive, allowing the second host to take over in the event of a primary host failure.
7. **Simplified construction:** PATA cables had 40-pin/80-wire ribbon cable. This was complex in structure. In comparison, SATA had a single 7 pin data cable and a 15 pin power cable. This cable resulted in a higher signaling rate, which translates
8. **Differential Signaling:** SATA uses differential signaling. Differential signaling is a technology which uses two adjacent wires to simultaneously the in-phase and out-of-phase signals. Thus, it is possible to transfer high-speed data with low operating voltage and low power consumption by detecting the phase difference between the two signals at the receiver's end.

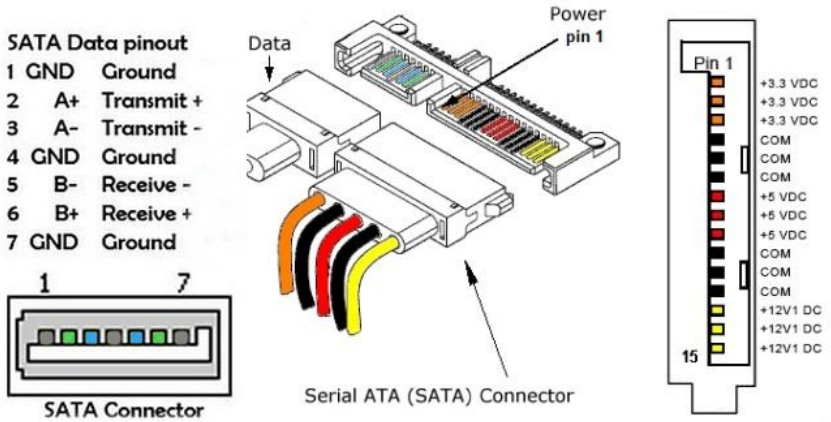
9. **High data transfer rate:** SATA has a high data transfer rate of 150/300/600 MBS/second. This capability of SATA allows for faster program loading, better picture loading and fast document loading.
10. **Large Cable Length :** SATA cable can be of length up to 1 meter, whereas PATA cable can only have a length of maximum 18 inches.

Operating Modes:
SATA operates on two modes:

1)IDE mode: IDE stands for Integrated Drive Electronics. This mode is used to provide backward compatibility with older hardware, which runs on PATA, at low performance.

2)AHCI mode: AHCI is an abbreviation for Advanced Host Controller Interface. AHCI is a high-performance mode that also provides support for hot-swapping.

The Serial ATA [SATA] bus is defined over two separate connectors, one connector for the data lines and one for the power lines.



Pin	Definition
1	Ground
2	A+(Transmit)
3	A-(Transmit)
4	Ground
5	B-(Receive)
6	B+(Receive)
7	Ground