SNS COLLEGE OF TECHNOLOGY

Coimbatore-35

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by
NAAC – UGC with 'A++' Grade

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

19ECT312 – EMBEDDED SYSTEM DESIGN

III YEAR/ VI SEMESTER

UNIT 4 : EMBEDDED OPERATING SYSTEM AND MODELING

TOPIC 4.5 : Input Output Subsystems

I/O System

- The I/O subsystem is treated as an independent unit in the computer
  - The CPU initiates I/O commands generically
    - Read, write, scan, etc
    - This simplifies the CPU
  - I/O modules are components that connect an I/O device to the I/O bus
    - The I/O module is an intermediary between CPU and the I/O device, and possibly between memory and the I/O device
  - This allows us to tailor I/O devices to specific uses without having to worry about how the CPU might be able to handle that new type of device

I/O Performance

Another reason to offload I/O to separate components is that I/O takes place at a different speed than processing and memory

- Often, I/O is not tuned to the system clock
- Often, I/O is slowed down by mechanical parts (e.g., disk drive read/write head, printer) and/or human interaction

We can look at enhancing any part of the computer and computing the speedup from the enhancement using Amdahl's Law

- $S = \frac{1}{1 - f + f/k}$
  - S = speedup
  - f = fraction of time enhancement can be used
  - k = speedup of the enhancement itself
- It should be noted that Amdahl's Law is used to determine any speedup, but here we will concentrate solely on I/O speedup

Using Amdahl's Law

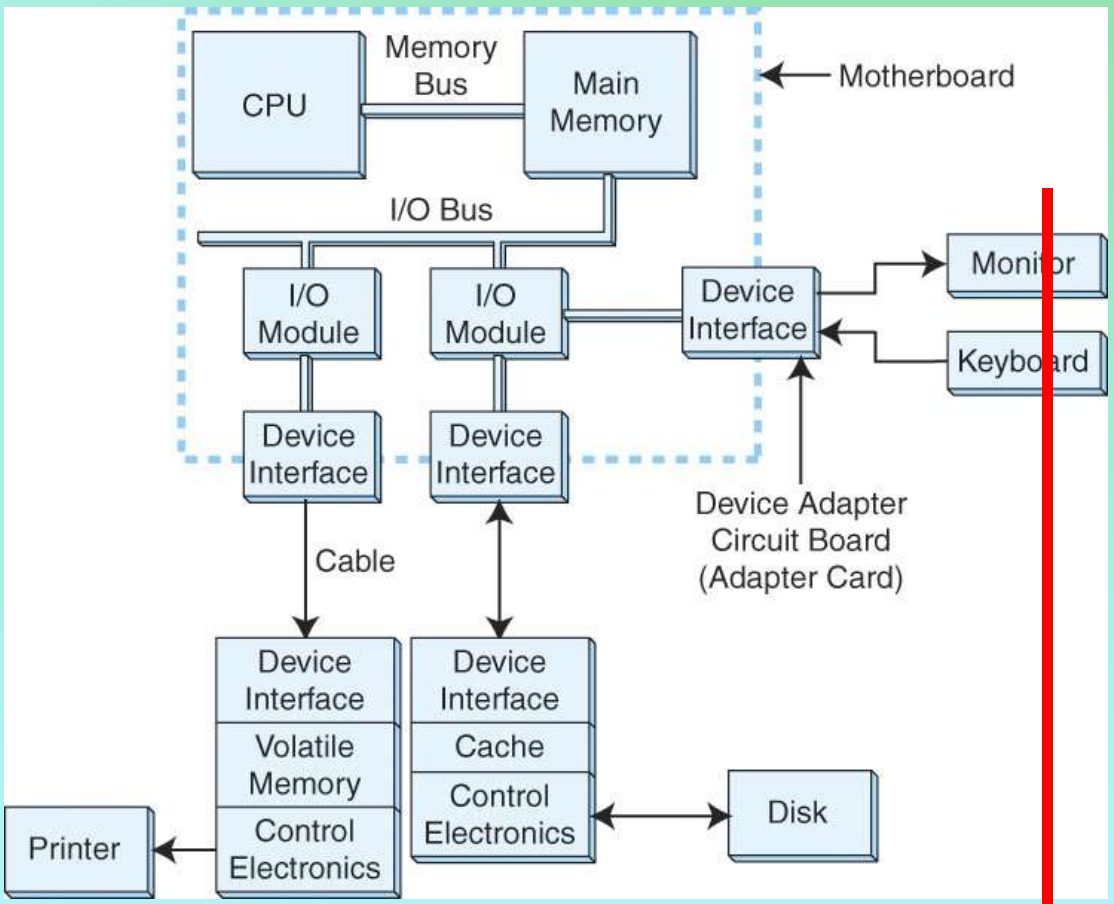
We have two possible enhancements, which should we choose?

- Processor upgrade for a process that is 50% faster
- New disk drives that offer a 150% speedup when in use
  - assume the processor is operating 70% of the time and that 30% of the time, the processor is waiting for disk access
- For enhancement 1, S = 70%, k = 1.5 (the book has a typo and lists this as 5, not 1.5)
- For enhancement 2, S = 30%, k = 2.5
  - $S1 = \frac{1}{1 - .7 + \frac{.7}{1.5}} = 1.30$
  - $S2 = \frac{1}{1 - .3 + \frac{.3}{2.5}} = 1.22$
  - Amdahl's Law is sometimes referred to as "make the common case fast" – even the disk drives offer a much greater speedup, they are used far less often, the more common case (70%) is the better one to pursue

I/O Architectures

I/O subsystem will typically include

- Blocks of memory dedicated to I/O buffering
- I/O bus(es)
- I/O devices
- Specialized interfaces (for instance for keyboard and monitor) and interface cards
- Possibly other connections (network, cable, etc)



Programmed I/O

Programmed I/O

- The CPU monitors the I/O module so that the CPU is not free to work on other things
  - This is an inefficient form of I/O, and the oldest form, it is typically not used today unless the program directly calls for input from keyboard or other device
- The process:
  - I/O instruction in the program causes CPU to examine I/O module's control register's ready bit – if it is ready, then the CPU issues the I/O command by setting the proper bits in this register
  - the I/O module issues the command to the I/O device and watches over device
  - when the I/O device has completed the task the I/O module sets the proper bits in its control register
  - the CPU, which has been watching this register, then cleans up the operation (on input, moves the input datum from the I/O module's data register to the CPU register or memory location)

Interrupt-Driven I/O

Here, the Interrupt system is used so that the CPU does not have to watch and wait for the I/O module

- The process:
  - The CPU issues the command to the I/O module
  - The CPU then continues with what it was doing
  - The I/O module, like before, issues the command to the I/O device and waits for the I/O to complete
  - Upon completion of one byte input or output, the I/O module sends an interrupt signal to the CPU
  - The CPU finishes what it was doing, then handles the interrupt
    - This will involve moving the resulting datum to its proper location on input
  - Once done with the interrupt, the CPU resumes execution of the program
- This is much more efficient than Programmed I/O as the CPU is not waiting during the (time-consuming) I/O process
  - However, if the I/O itself takes more than 1 input or output (that is, if the amount being transferred is greater than 1 byte), then this interrupts the CPU often, so instead…

Handling Interrupts

Device raises the interrupt (sends interrupt signal)

At the end of the current fetch execute cycle, CPU checks for interrupt, if there is one, it saves its status (register values)

CPU identifies interrupting device (we talk about how this is done later)

Given the interrupting device number, use this to map into the Interrupt vector table

- For instance, interrupt 15 would be at location 15 in memory, this entry stores the location of interrupt 15's handler in operating system memory

Move this address into the PC

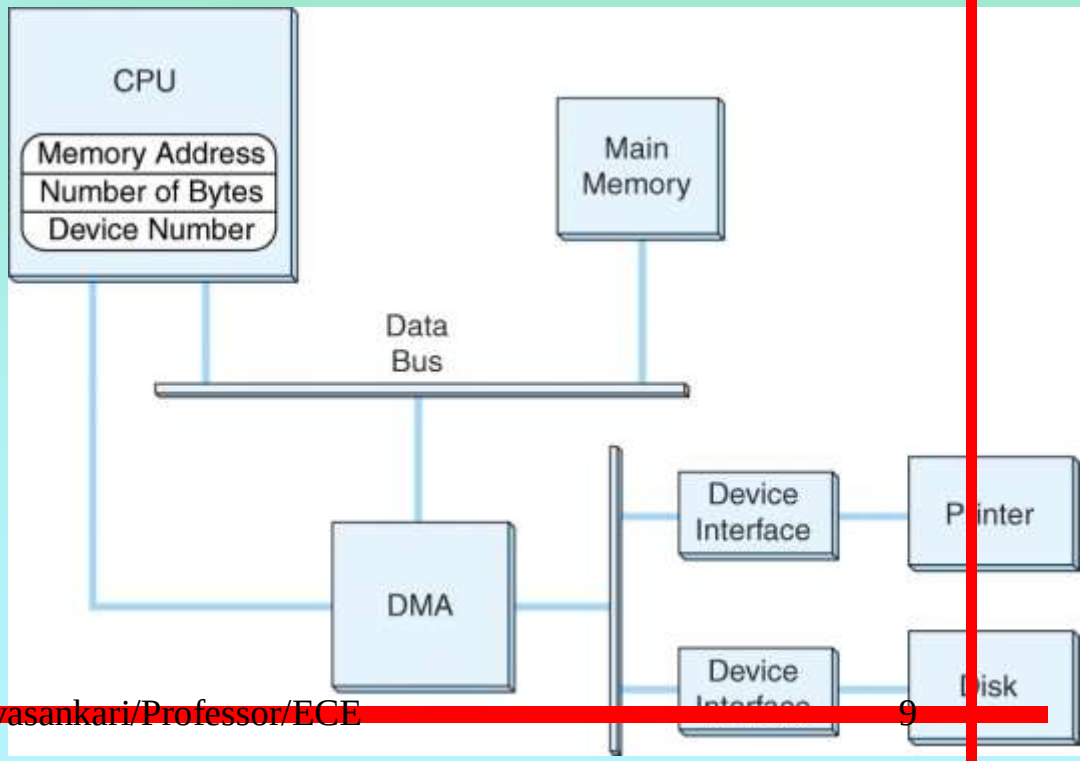
Execute interrupt handler

Upon completion, restore saved register values

DMA I/O

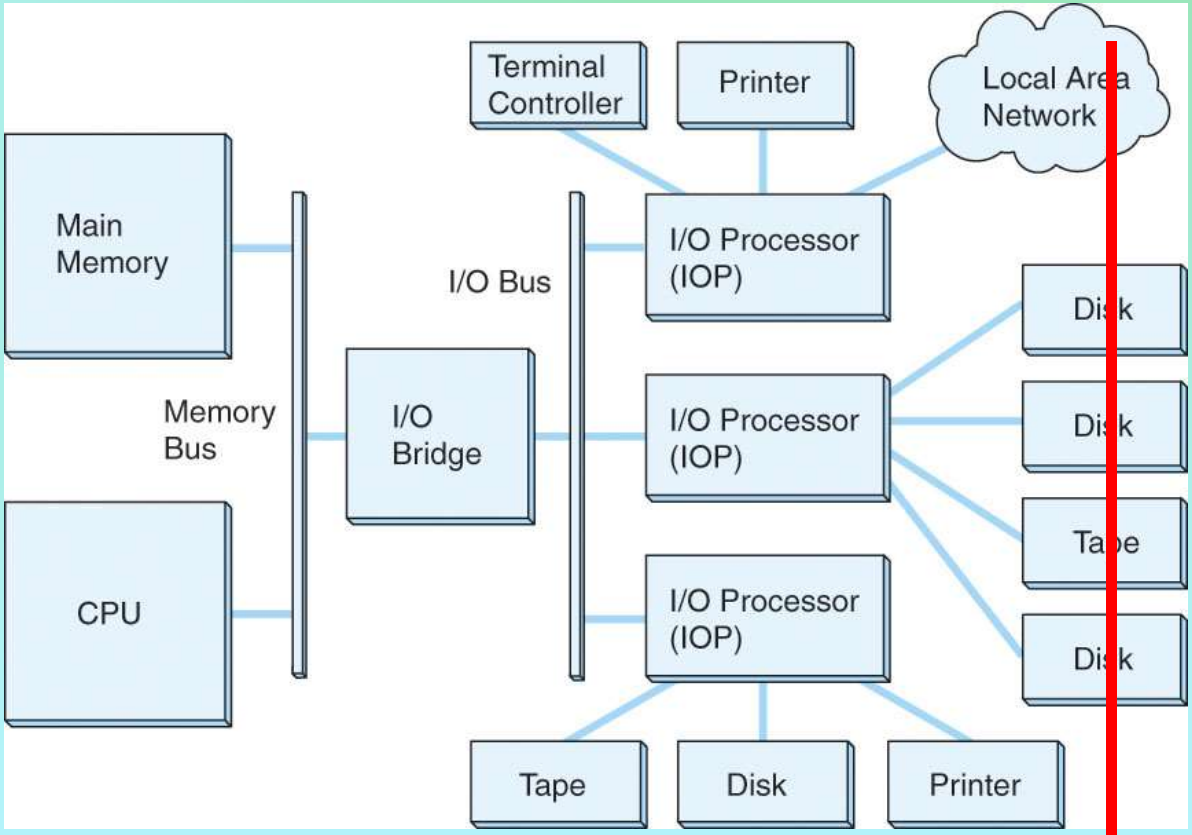
Direct memory access allows an I/O module to communicate directly with memory so that the CPU does not have to be interrupted for each data movement

- In any large transfer, DMA I/O is preferable to Interrupt-driven I/O
- The DMA controller only interrupts the CPU once, once the entire data transfer is complete
- Note: memory is a slave device so it cannot initiate bus transfers, the DMA controller then acts as a memory master
- The DMA controller includes a count register set by the CPU to count the number of bytes still to be transferred
- The DMA controller acts as an interface between I/O device and memory



Channel I/O

- In DMA I/O, if there is a failure of the I/O device, the I/O module interrupts the CPU to handle the problem
- A more sophisticated I/O module is known as an I/O channel
  - this is an I/O module with its own mini-processor (known as an I/O processor)
- Channel I/O can handle any problem without having to interrupt the CPU



I/O architecture with I/O Channels

I/O Accessing and Addressing

I/O accesses use the Expansion bus and System bus

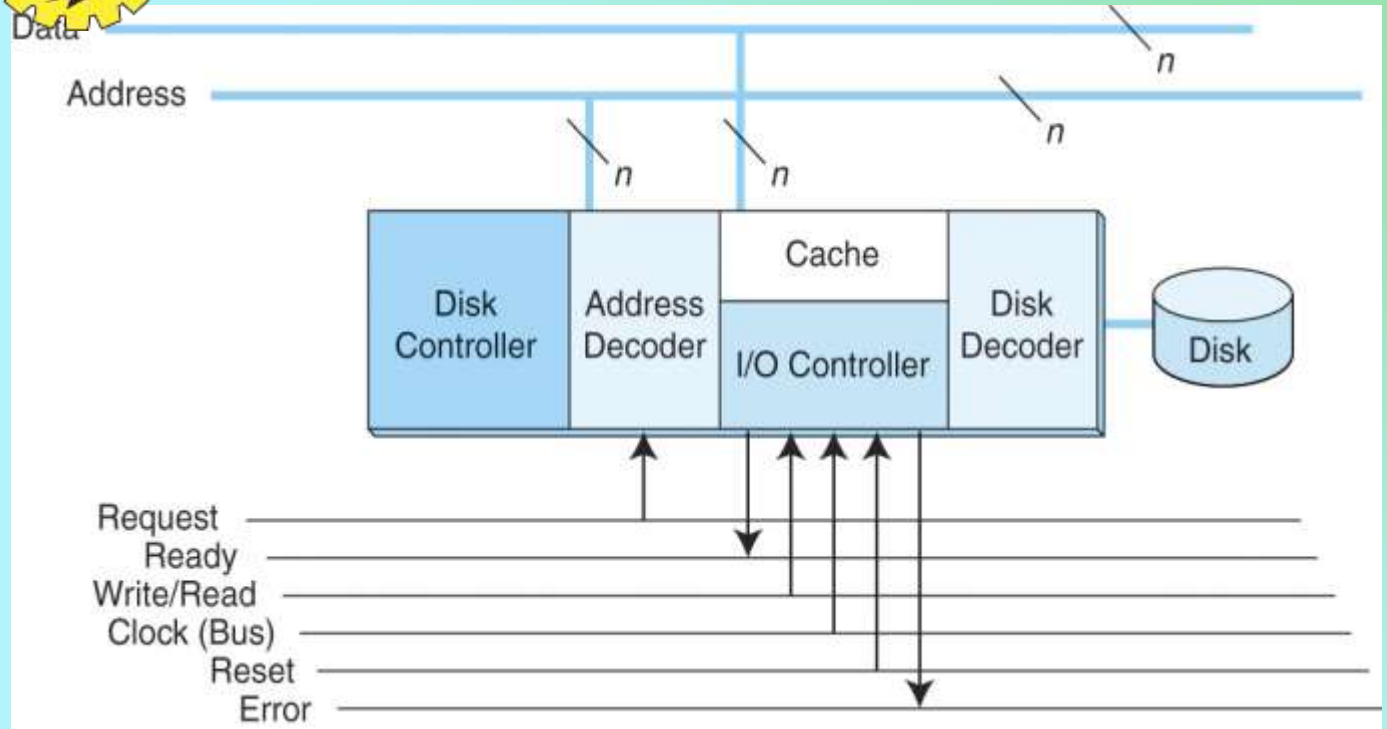
- as multiple devices might need access at the same time, bus access must be governed
- three approaches are:
  - bus arbiter, a controller to select the device that next gets to use the bus based on a priority scheme
  - an I/O monitor checks for bus traffic and if it is busy, waits
  - use the CPU as an arbiter

- To determine which device is to handle a CPU command, the CPU sends the device's address on the address bus (along with the control command of read or write)
  - There are two forms of addressing used in computers:
    - Isolated I/O – every device has a unique address and so device j watches for address over the address bus, and the CPU uses two sets of control lines, one to memory, one to the I/O subsystem
    - Memory-mapped I/O – some memory addresses are reserved for I/O module registers, and communication from the CPU to memory & I/O is generic
    - Memory-mapped while being cheaper because of the need for fewer control lines takes away some memory addresses so those memory locations are never used

Forms of I/O

- Character versus block
  - We differentiate between whether we expect a single datum (byte or word size) or a block of data
    - Character: keyboard, mouse, monitor, printer, modem
    - Block: storage device (disk drive, tape drive)
  - Notice that block devices are usually much higher speed than character
  - Also, block devices tend to use DMA while character uses interrupt-driven
- Synchronous vs asynchronous
  - Synchronous devices use the system clock to regular usage (note that a transfer may not occur at every clock cycle but it will start with a clock pulse)
  - Asynchronous devices are not regulated by the clock and the device may start or stop communicating at any time (think of a network message coming in, or a response from the printer)

I/O Control Lines



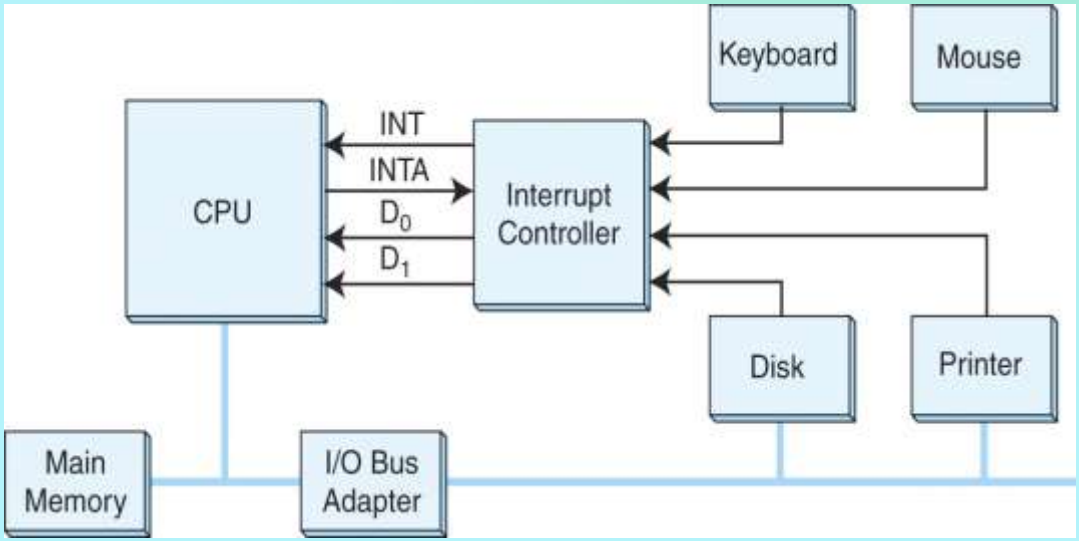
Here we see more detail on the I/O bus including control lines

Request from CPU goes to I/O module – basically asking “are you ready?”
Ready goes from I/O module to CPU
Read/write indicates the type of operation
Clock is used to synchronize transfer if available
Error denotes that the I/O module wants the CPU’s attention

Interrupt Controller

One final controller is the interrupt controller

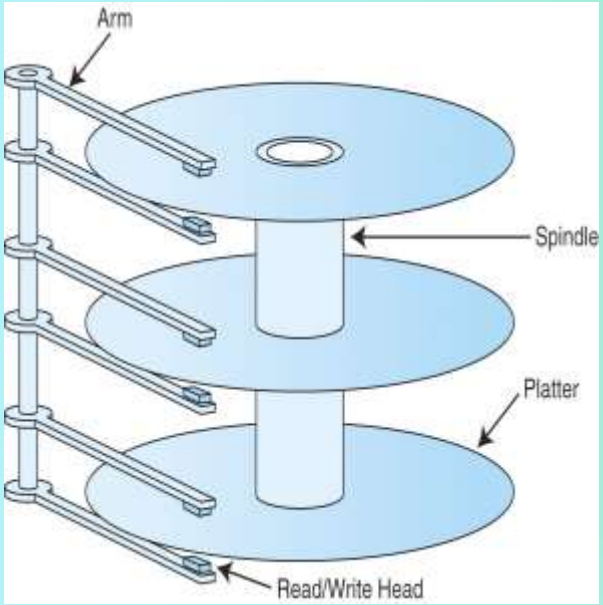
- All interrupts go directly to a controller rather than to the CPU
- The controller then interrupts the CPU and waits for the interrupt to be acknowledged
  - The interrupt controller prioritizes interrupts from the devices based on the importance of the device
    - Prioritization is established by the architects when the controller is designed

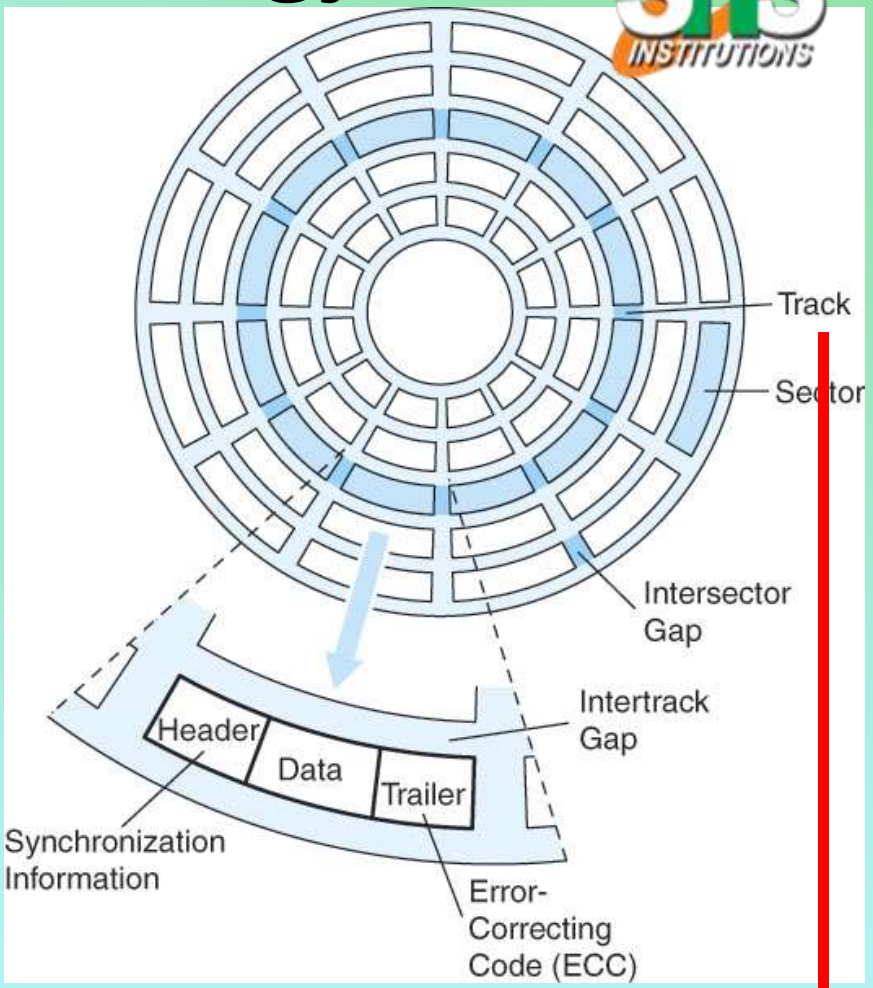


The data lines D0, D1 are used to specify the interrupting device

Some architectures do not have data lines, instead the interrupting device must be determined some other way

Disk Technology





We now turn to some specific I/O devices

Disk includes both magnetic disk (floppy, hard)
and optical disk

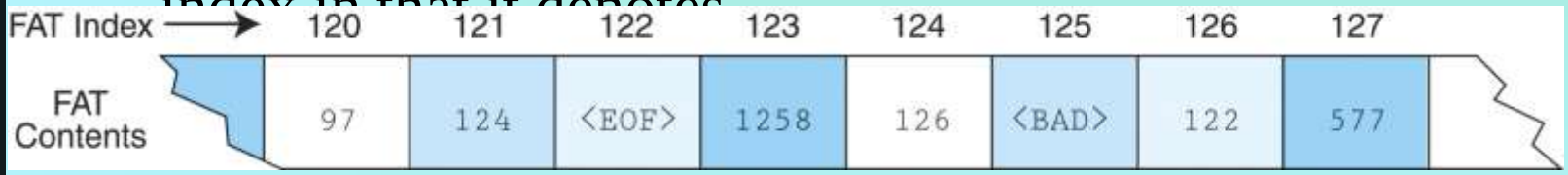
Data is stored on disk by orienting magnetized spots on the surface (in one direction, we will call them positive or 1, other direction is negative or 0)

Disk Access

- Disk access is direct
  - the read/write head can move directly to the proper disk location
    - this is sometimes erroneously called random access
- A special directory, often called a File Allocation Table (FAT) is like an index in that it denotes

- Usually a disk file is broken into blocks and distributed around the disk (not in sequential order), often interleaved to skip sectors on the same track

Disk access time = seek time
+ rotational delay
+ transfer time



The FAT index number is a combination of the track and sector number, so a file which starts at 124 continues at 126 and then 122, the FAT also denotes bad sectors and free locations

Floppy vs. Hard Disks

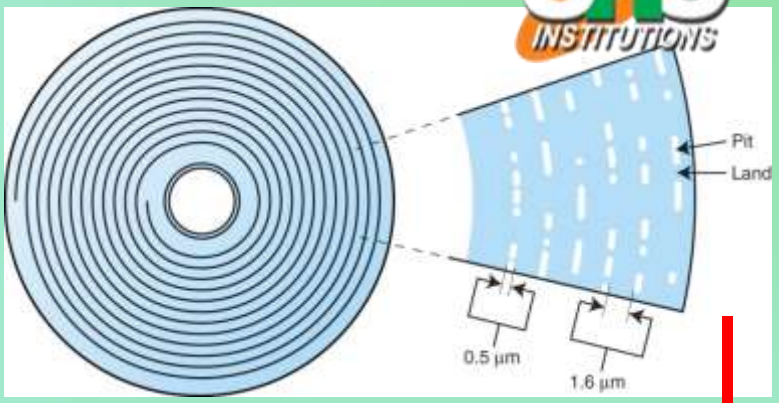
Hard disks are permanently sealed inside of their disk drives

- This permits greater density by packing the bits (magnetic charges) on the disk closer together
- Usually hard disks have 3 or more platters provided multiple surfaces
- These disks can spin as much as 15,000 RPMs although typically spin between 5400 and 7200
- There may be hundreds of sectors and thousands of tracks
- Hard disks are susceptible to failure because of their moving components and head crashes

- Floppy disks are removable from their drives giving the user portability
- Floppy disks used to be much more common back when hard disk technology was very expensive
  - Floppy disks have a capacity of 1.44 MByte, hard disk capacities are now in the 10s of GBytes
  - A 20 MByte harddisk used to cost $500+, now a 40 GByte drive costs $100 so floppy disks are seldomly used except for portability

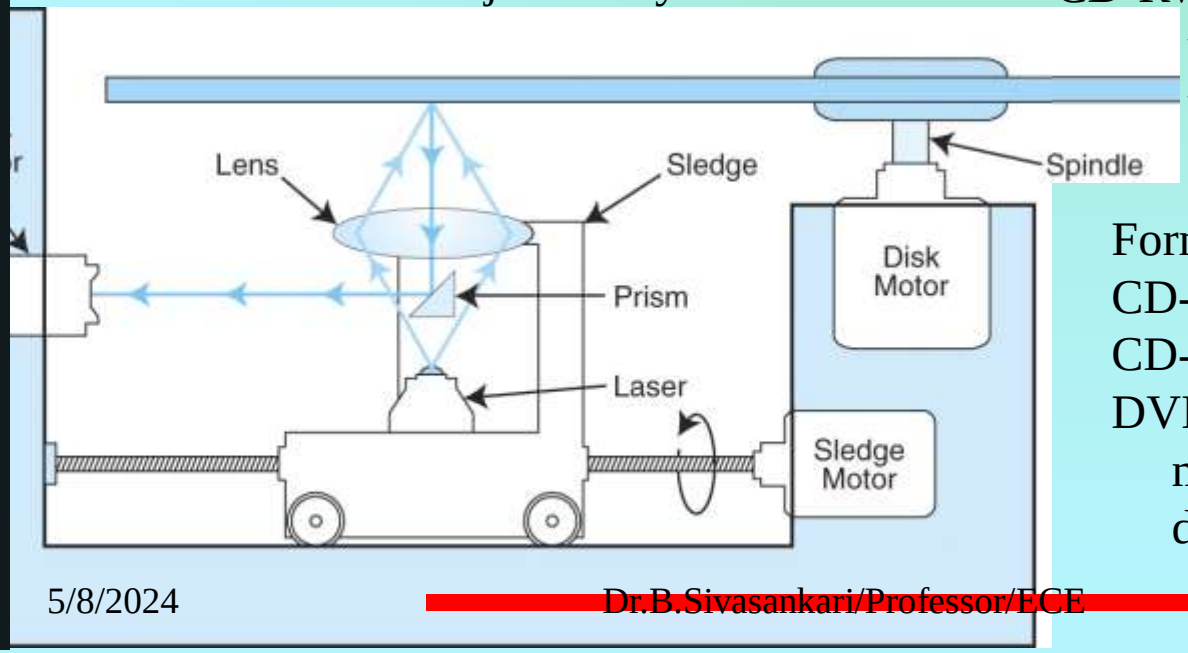
Optical Disk Drives

Magnetic media can be interferred with by a strong
magnetic field (magnet, speaker, tv, telephone)



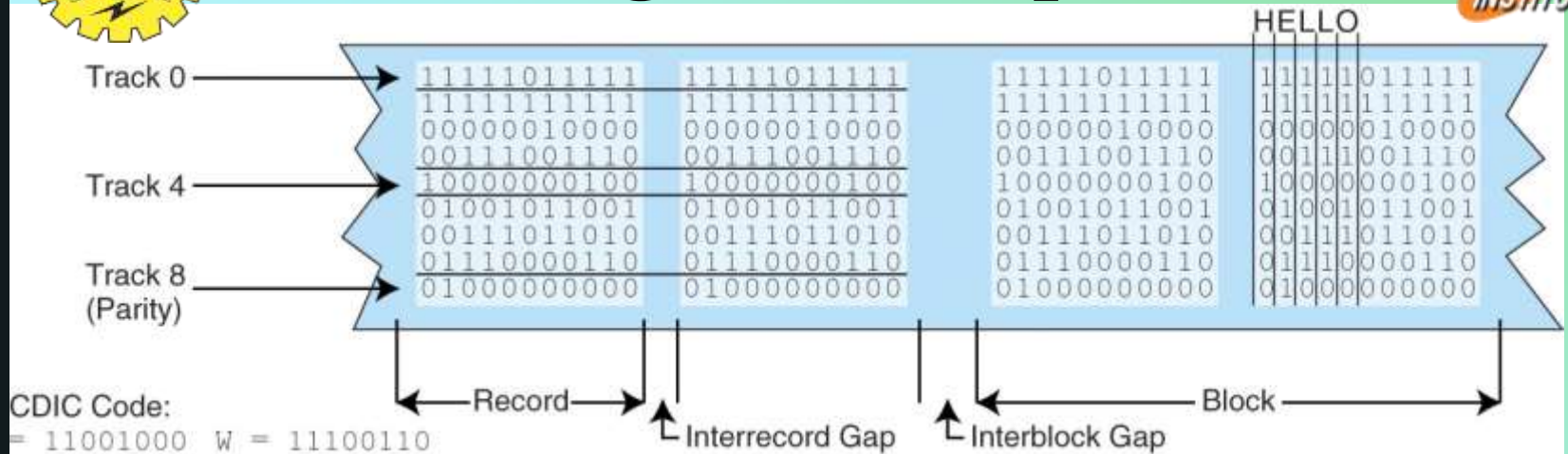
A more reliable medium is the optical disk information is either burned onto the surface by
laser or the laser adjusts a crystal to reflect

Burning onto the disc creates pits that cannot be erased, the newer CD-RW technology adjusts crystals, this allows for erasures but the technology is more expensive



Forms of optical disk:
CD-ROM – factory manufactured
CD-R, CD-RW – blank, erasable
DVD – greater capacity thanks to more density, multi-layered and data compression

Magnetic Tape



The oldest form of storage, and virtually obsolete

Uses sequential access, so is very slow (if you want file 31, you have to fast forward

past files 1-30 first)

It also does not permit deletion and rewriting in the same space because you might

erase part of another file

Primarily used for large scale backup and archive because the sequential access does

Solid State Drives

Disk access is far slower than memory access but is non-volatile (contents are retained without power)

Main memory is much faster, much more rugged (not susceptible to head crashes, has no moving parts, etc) but is more expensive and volatile

Compromise: solid state disk drives

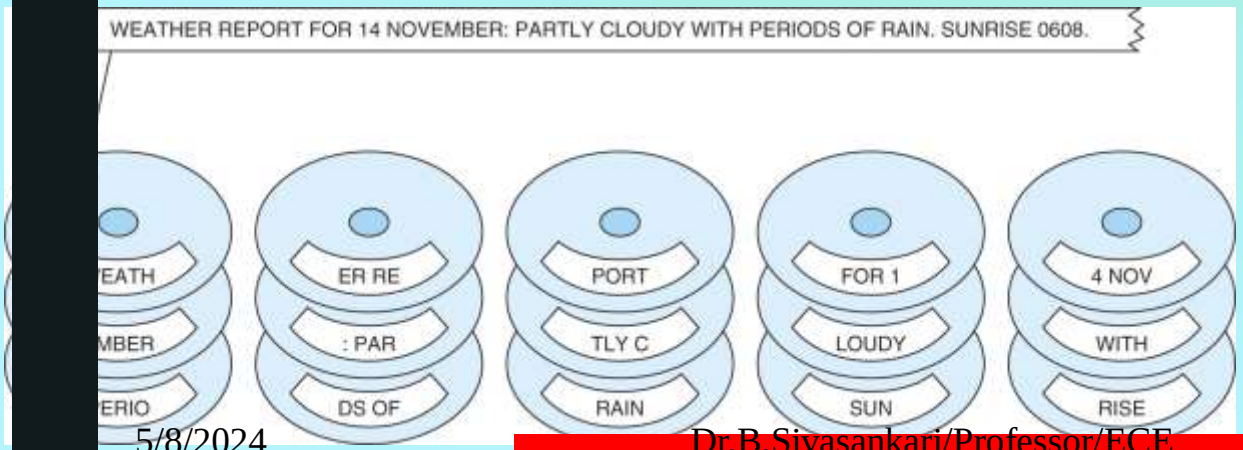
- NAND or NOR gates are used to build flash memory
  - NOR-based is byte erasable, NAND is block erasable
- Far more expensive than hard disk (for the quantity) but much faster
  - 2-3 times more expensive per byte of storage
  - 100 times faster than disk (maybe 100,000 times slower than DRAM)

Flash memory storage locations can wear out (estimated between 30,000 and 1 million updates, this is a vast improvement over the original 1000 update limits from EEPROM as little as a decade ago)

RAID

Hard disk storage is critically important because of our reliance on the hard disk to store virtually everything

- However, the hard disk is the most likely component to fail on us because of its usage and its high speed
- If we want reliable access, we might want to use a degree of redundancy by adding additional disks to our drive unit
  - This is the idea behind RAID
    - Redundant array of independent disks (or)
    - Redundant array of inexpensive disks
- There are 7 forms (or levels) of RAID, each one different, and each with its own strengths and weaknesses

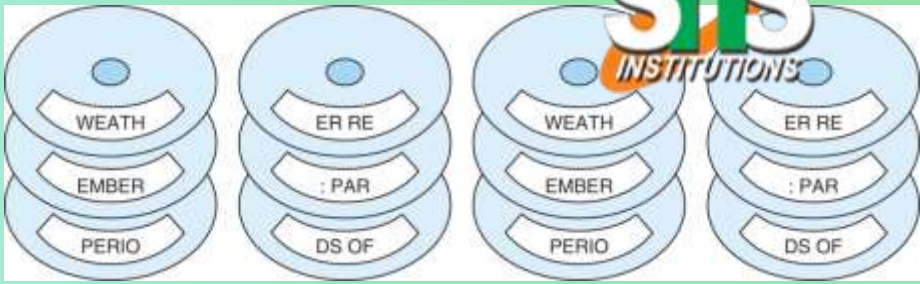


RAID 0 offers no redundancy, but improves disk access

Here, files are broken into strips and distributed across disk surfaces (known as disk spanning) so that access to a single file can be done in parallel disk accesses

RAIDS 1, 2 and 3

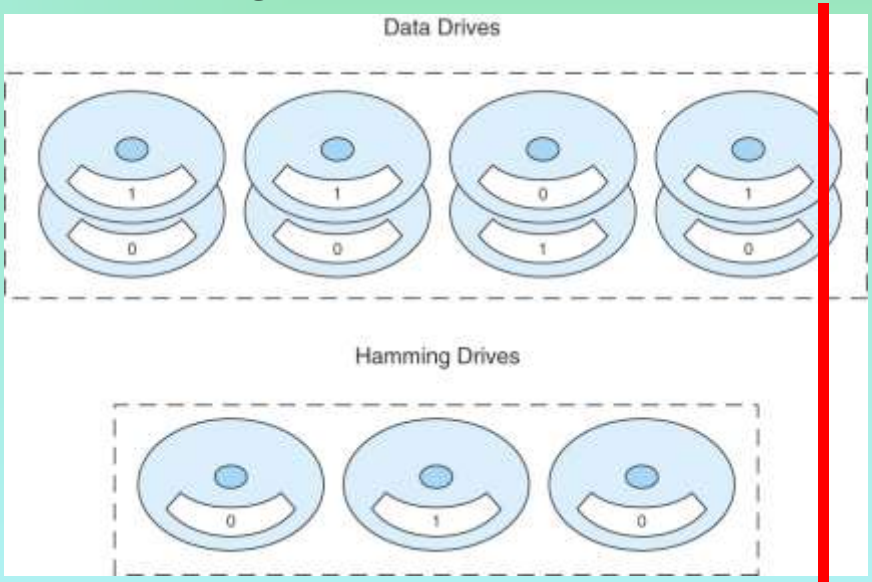
RAID 1 creates a mirror for every disk giving
100% redundancy at twice the cost, 2 reads
can be done in parallel, one from each mirror,
but writes require saving to both sets of disk



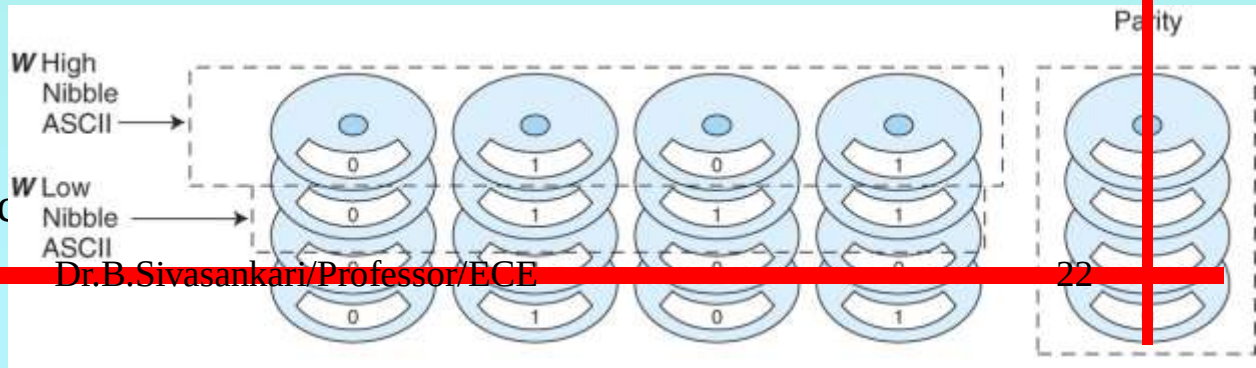
Original Mirror

RAID 1 may be too expensive, so an alternative is to just store parity information

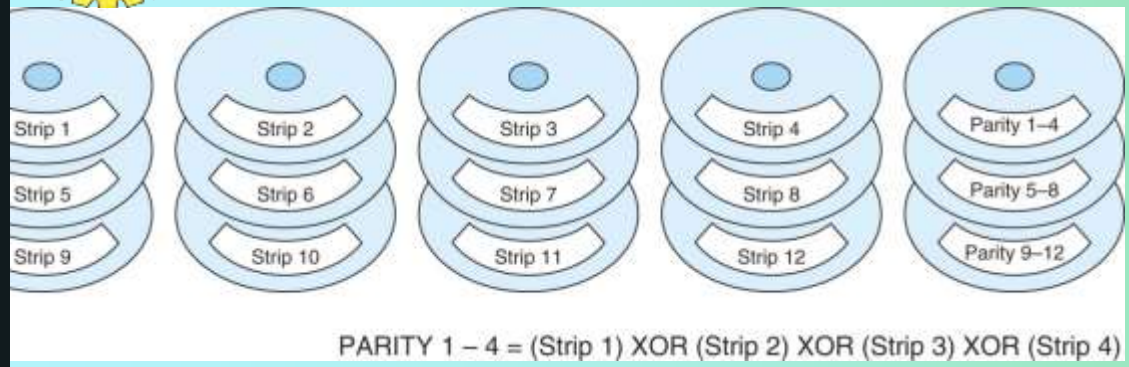
RAID 2 strips each byte into 1 bit per disk and
uses additional disks to store Hamming codes
for redundancy



information for redundanc

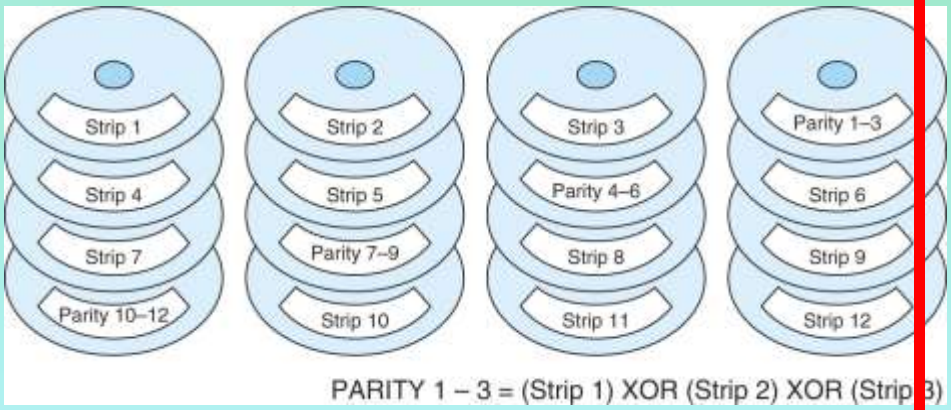


RAID 4, 5 and 6

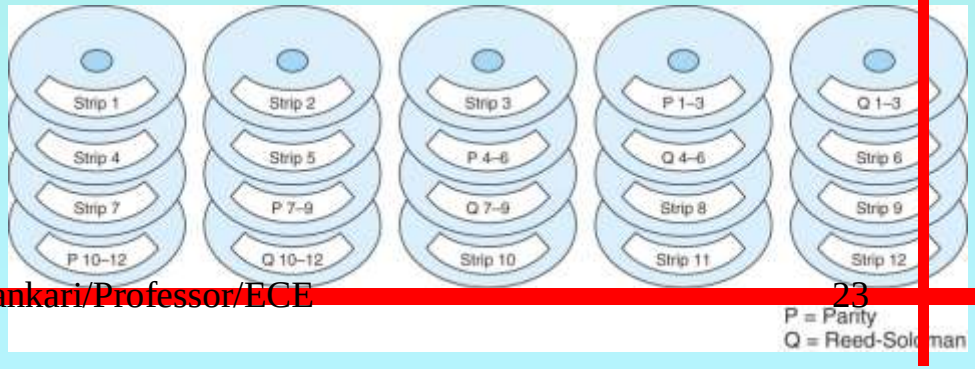


In RAIDs 4, 5 and 6, disk strips are purposefully large to allow multiple I/O accesses to occur in parallel since any two requests will be for two different strips, hopefully on two different drives

In RAID 4, all parity information is placed on a single disk which creates a bottleneck and so defeats the advantage of parallel accesses (if I/O requests require accessing a strip plus the parity disk, then only 1 access can occur at a time)



In RAID 5, parity information is interleaved across disks so that hopefully two accesses can take place in parallel



In RAID 6, parity information is duplicated for greater redundancy