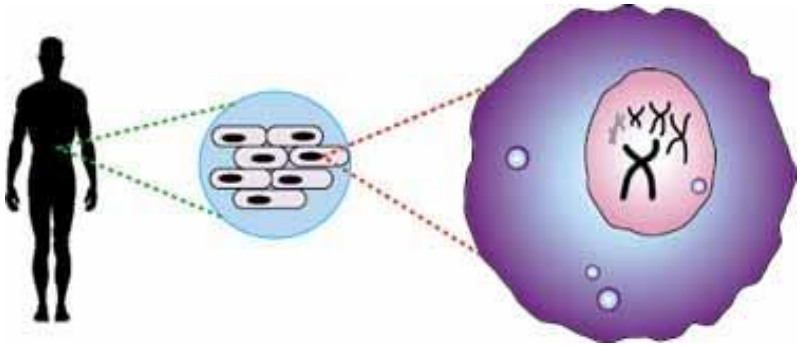
Introduction to I/O

- Where does the data for our CPU and memory come from or go to?
- Computers communicate with the outside world via I/O devices.
  - Input devices supply computers with data to operate on.
  - Results of computations can be sent to output devices.
- Today we'll talk a bit about I/O system issues.
  - I/O performance affects the overall system speed.
  - We'll look at some common devices and estimate their performance.
  - We'll look at how I/O devices are connected (by buses).

I/O is important!



- Many tasks involve reading and processing enormous quantities of data.
  - CCSO (counterpart of ITaP at UIUC) has two machines and 144GB of storage for a local web cache.
  - Institutions like banks and airlines have huge databases that must be constantly accessed and updated.
  - Celera Genomics is a company that sequences genomes, with the help of computers and 100 trillion bytes of storage!
- I/O is important for us small people too!
  - People use home computers to edit movies and music.
  - Large software packages may come on multiple compact discs.
  - Everybody and their grandparents surf the web!

I/O is slow!

- How fast can a typical I/O device supply data to a computer?
  - A fast typist can enter 9-10 characters a second on a keyboard.
  - Common local-area network (LAN) speeds reach 1Gbit/s, which is about 125MB/s.
  - Today's hard disks provide a lot of storage and transfer speeds around 40-60MB per second.
- Unfortunately, this is excruciatingly slow compared to modern processors and memory systems:
  - Modern CPUs can execute more than a billion instructions per second.
  - Modern memory systems can provide 2-4 GB/s bandwidth.

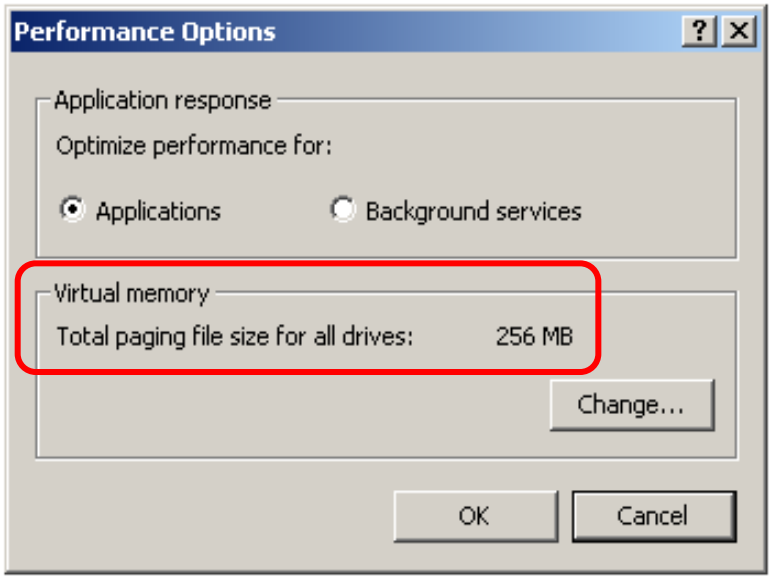
- I/O performance has not increased as quickly as CPU performance, partially due to neglect and partially to physical limitations.
  - This is changing, with faster networks, better I/O buses (e.g., optical bus by IBM, 2010), RAID drive arrays, and other new technologies.

I/O speeds often limit system performance

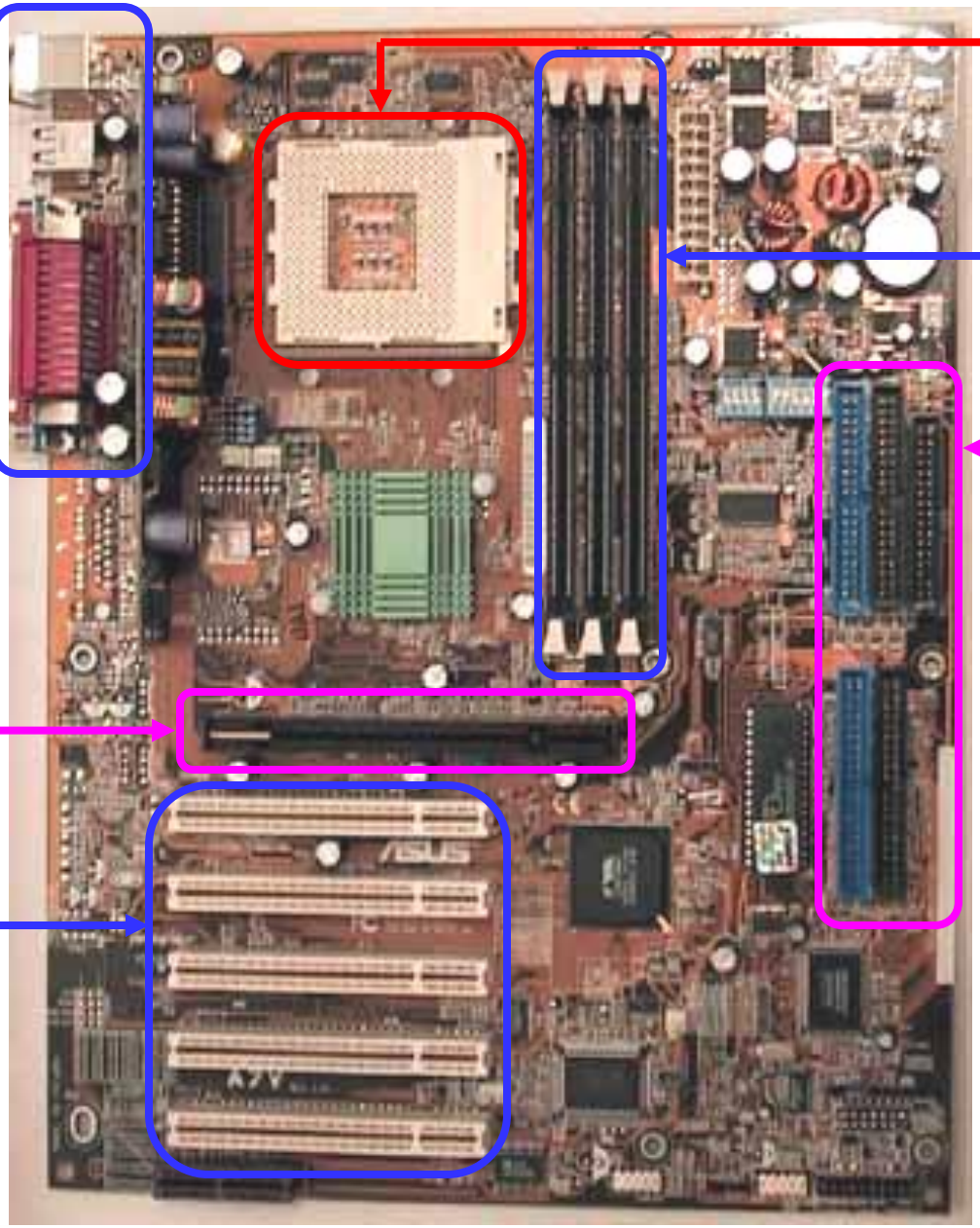
- Many computing tasks are I/O-bound, and the speed of the input and output devices limits the overall system performance.
- This is another instance of Amdahl's Law. Improved CPU performance alone has a limited effect on overall system speed.

Common I/O devices

- Hard drives are almost a necessity these days, so their speed has a big impact on system performance.
  - They store all the programs, movies and assignments you crave.
  - Virtual memory systems let a hard disk act as a large (but slow) part of main memory.
- Networks are also ubiquitous nowadays.
  - They give you access to data from around the world.
  - Hard disks can act as a cache for network data. For example, web browsers often store local copies of recently viewed web pages.



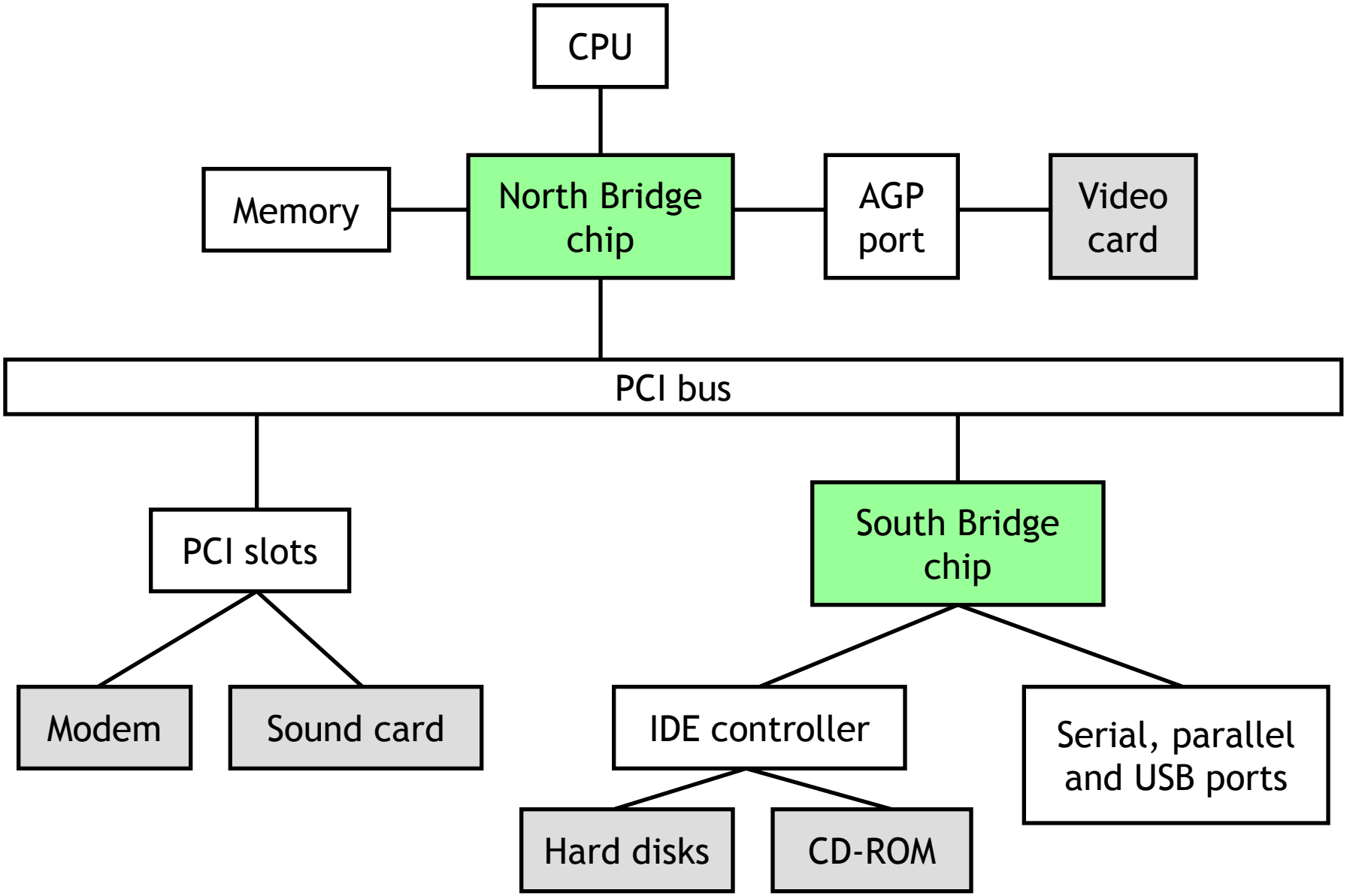
The Hardware (the motherboard)



What is all that stuff?

- Different motherboards support different CPUs, types of memories, and expansion options.
- The picture is an Asus A7V.
  - The CPU socket supports AMD Duron and Athlon processors.
  - There are three DIMM slots for standard PC100 memory. Using 512MB DIMMs, you can get up to 1.5GB of main memory.
  - The AGP slot is for video cards, which generate and send images from the PC to a monitor.
  - IDE ports connect internal storage devices like hard drives, CD-ROMs, and Zip drives.
  - PCI slots hold other internal devices such as network and sound cards and modems.
  - Serial, parallel and USB ports are used to attach external devices such as scanners and printers.

How is it all connected?

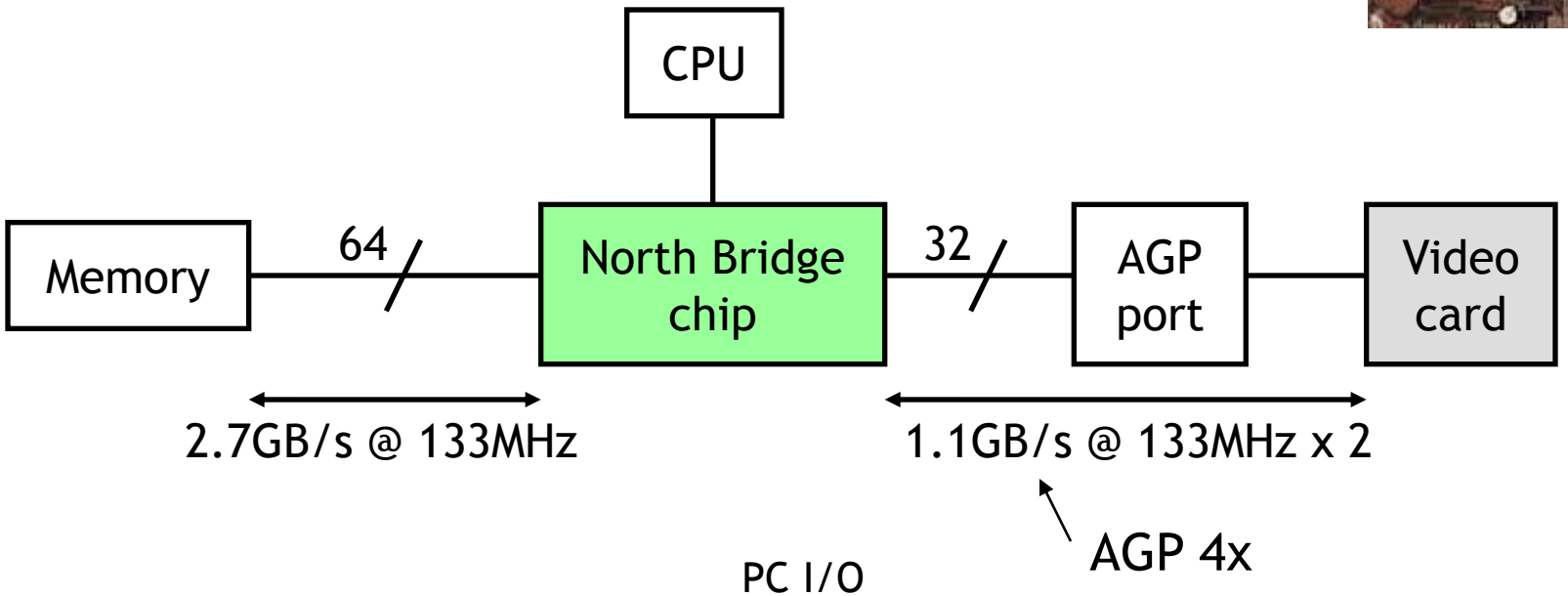


Frequencies

- CPUs actually operate at two frequencies.
  - The internal frequency is the clock rate inside the CPU, which is what we've been talking about so far.
  - The external frequency is the speed of the processor bus, which limits how fast the CPU can transfer data.
- The internal frequency is usually a multiple of the external bus speed.
  - A 2.167 GHz Athlon XP sits on a 166 MHz bus (166 x 13).
  - A 2.66 GHz Pentium 4 might use a 133 MHz bus (133 x 20).
    - *You may have seen the Pentium 4's bus speed quoted at 533MHz. This is because the Pentium 4's bus is "quad-pumped", so that it transfers 4 data items every clock cycle.*

The North Bridge

- To achieve the necessary bandwidths, a "frontside bus" is often dedicated to the CPU and main memory.
  - "bus" is actually a bit of a misnomer as, in most systems, the interconnect consists of point-to-point links.
  - The video card, which also need significant bandwidth, is also given a direct link to memory via the Accelerated Graphics Port (AGP).
- All this CPU-memory traffic goes through the "north bridge" controller, which can get very hot (hence the little green heatsink).

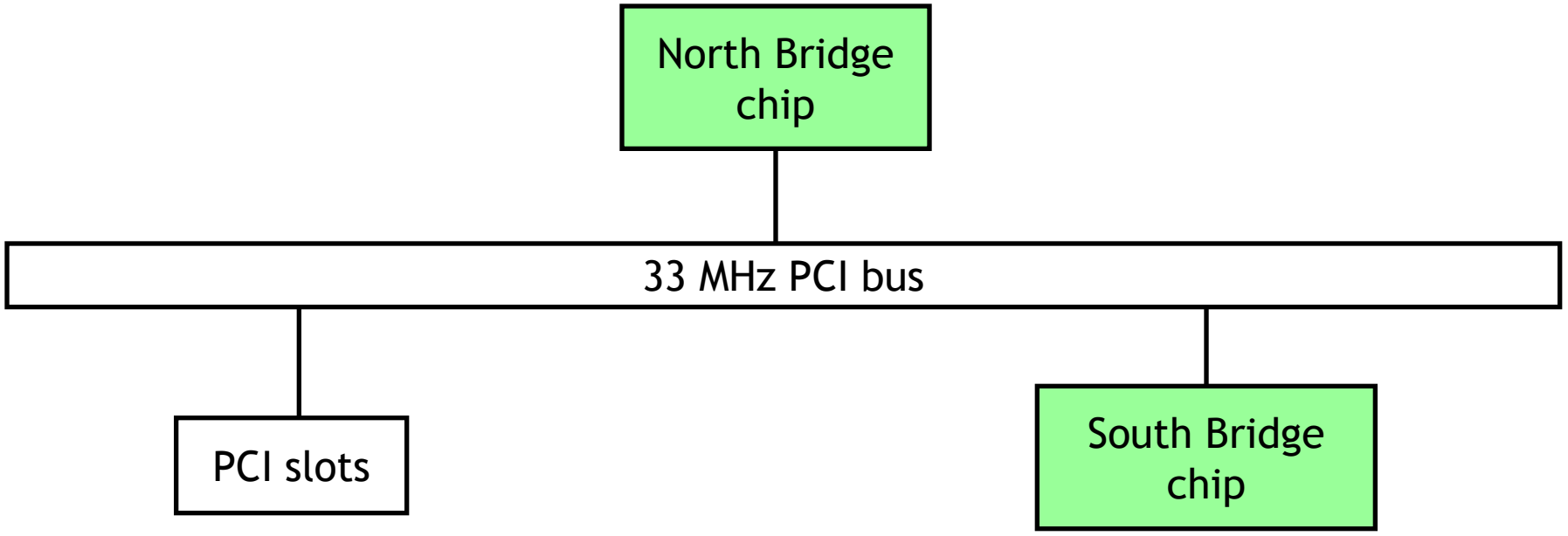


PCI

- Peripheral Component Interconnect is a synchronous 32-bit bus running at 33MHz, although it can be extended to 64 bits and 66MHz.
- The maximum bandwidth is about 132 MB/s.

33 million transfers/second x 4 bytes/transfer = 132MB/s

- Cards in the motherboard PCI slots plug directly into the PCI bus.
- Devices made for the older and slower ISA bus standard are connected via a "south bridge" controller chip, in a hierarchical manner.



External buses

- External buses are provided to support the frequent plugging and un-plugging of devices
  - As a result their designs significantly differ from internal buses

- Two modern external buses, Universal Serial Bus (USB) and FireWire, have the following (desirable) characteristics:
  - Plug-and-play standards allow devices to be configured with software, instead of flipping switches or setting jumpers.
  - Hot plugging means that you don't have to turn off a machine to add or remove a peripheral.
  - The cable transmits power! No more power cables or extension cords.
  - Serial links are used, so the cable and connectors are small.

FireWire

The Serial/Parallel conundrum

- Why are modern external buses serial rather than parallel?

- Generally, one would think that having more wires would increase bandwidth and reduce latency, right?
  - Yes, but only if they can be clocked at comparable frequencies.

- Two physical issues allow serial links to be clocked significantly faster:
  - On parallel interconnects, interference between the signal wires becomes a serious issue.
  - Skew is also a problem; all of the bits in a parallel transfer could arrive at slightly different times.

- Serial links are being increasingly considered for internal buses:
  - Serial ATA is a new standard for hard drive interconnects
  - PCI-Express (aka 3GI/O) is a PCI bus replacement that uses serial links

I/O Mechanisms

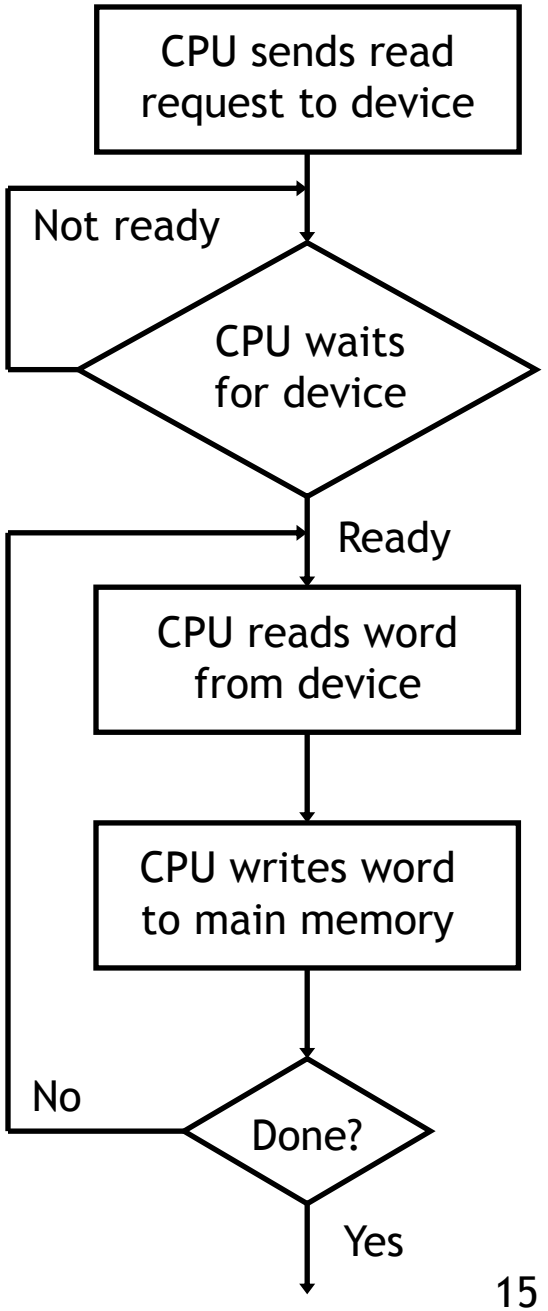
- Most I/O requests are made by applications or the operating system, and involve moving data between an I/O device and main memory.

- There are several ways of performing data transfers between devices and main memory.
  - Programmed I/O
  - Interrupt-driven I/O
  - Direct memory access (DMA)

Transferring data with *programmed I/O*

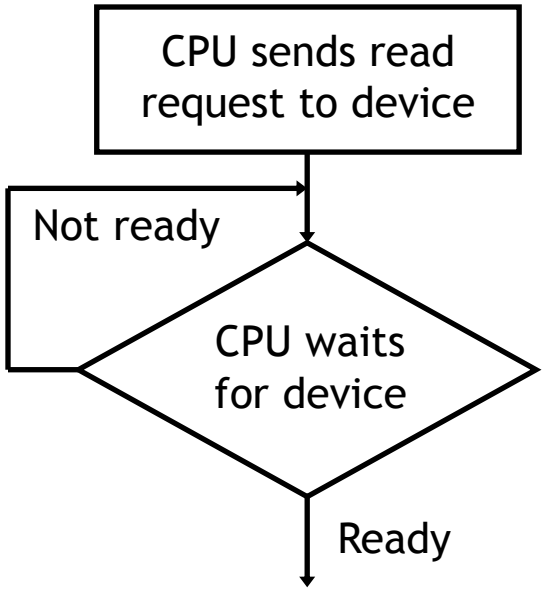
- Under programmed I/O, it's all up to a user program or the operating system.
  - The CPU makes a request and then waits for the device to become ready (*e.g.*, to move the disk head).
  - Buses are only 32-64 bits wide, so the last few steps are repeated for large transfers.
- A lot of CPU time is needed for this!
  - If the device is slow the CPU might have to wait a long time—as we will see, most devices *are* slow compared to modern CPUs.
  - The CPU is also involved as a middleman for the actual data transfer.

(This CPU flowchart is based on one from *Computer Organization and Architecture* by William Stallings.)



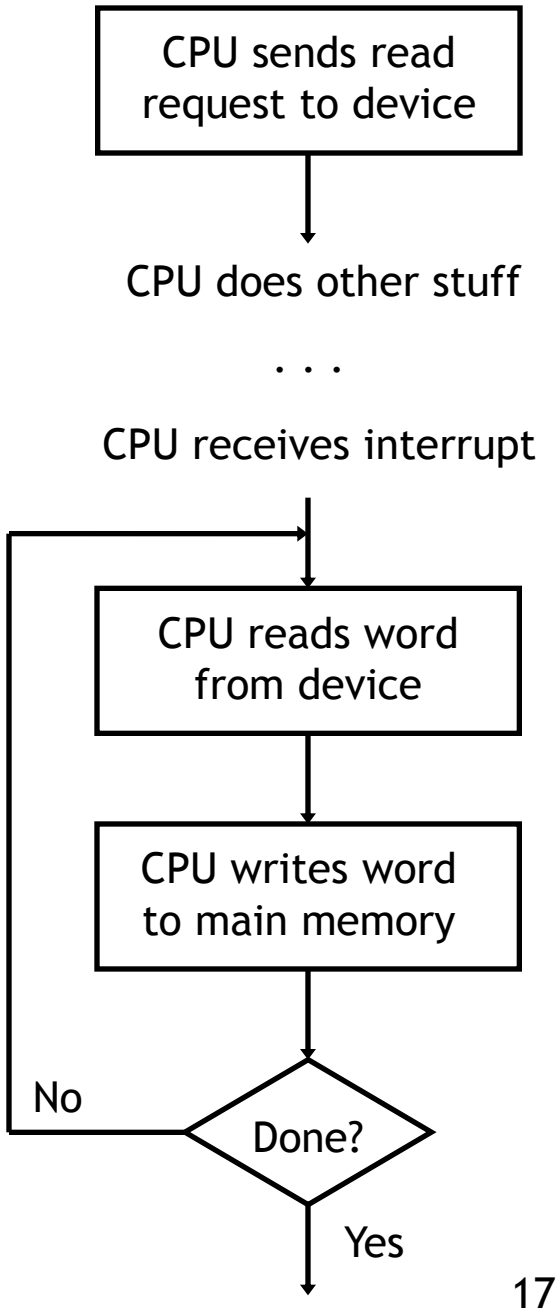
Can you hear me now? Can you hear me now?

- Continually checking to see if a device is ready is called polling.
- It's not a particularly efficient use of the CPU.
  - The CPU repeatedly asks the device if it's ready or not.
  - The processor has to ask often enough to ensure that it doesn't miss anything, which means it can't do much else while waiting.
- An analogy is waiting for your car to be fixed.
  - You could call the mechanic every minute, but that takes up all your time.
  - A better idea is to wait for the mechanic to call *you*.



Interrupt-driven I/O

- Interrupt-driven I/O attacks the problem of the processor having to wait for a slow device.
- Instead of waiting, the CPU continues with other calculations. The device interrupts the processor when the data is ready.
- The data transfer steps are still the same as with programmed I/O, and still occupy the CPU.



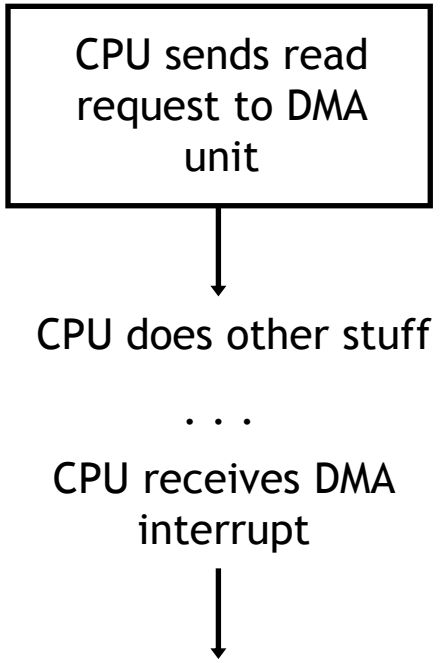
(Flowchart based on Stallings again.)

Interrupts

- Interrupts are external events that require the processor's attention.
  - I/O devices may need attention.
  - Timer interrupts to mark the passage of time.
- These situations are not errors.
  - They happen normally.
  - All interrupts are recoverable:
    - The interrupted program will need to be resumed after the interrupt is handled.
- It is the operating system's responsibility to do the right thing, such as:
  - Save the current state and shut down the hardware devices.
  - Find and load the correct data from the hard disk
  - Transfer data to/from the I/O device, or install drivers.

Direct memory access

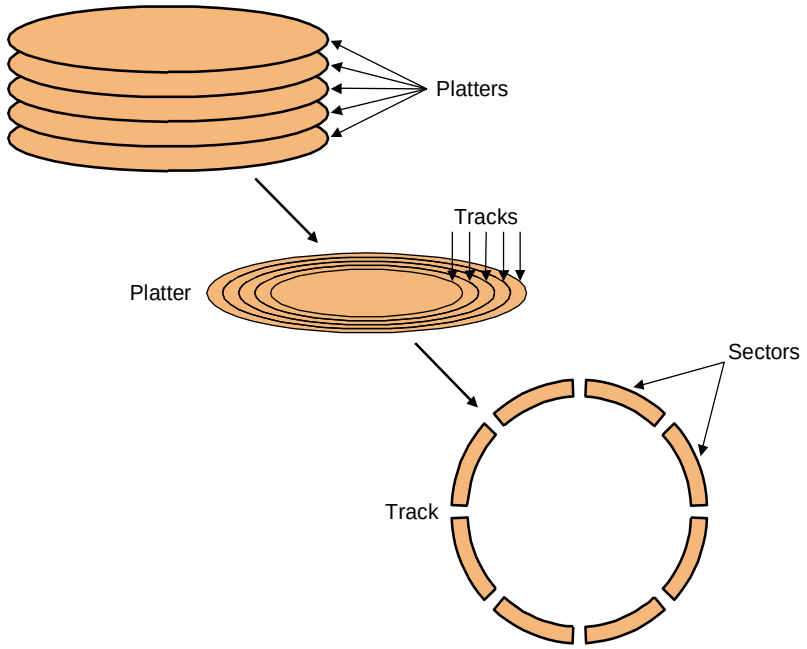
- One final method of data transfer is to introduce a direct memory access, or DMA, controller.
- The DMA controller is a simple processor which does most of the functions that the CPU would otherwise have to handle.
  - The CPU asks the DMA controller to transfer data between a device and main memory. After that, the CPU can continue with other tasks.
  - The DMA controller issues requests to the right I/O device, waits, and manages the transfers between the device and main memory.
  - Once finished, the DMA controller interrupts the CPU.
- This is yet another form of parallel processing.



(Flowchart again.)

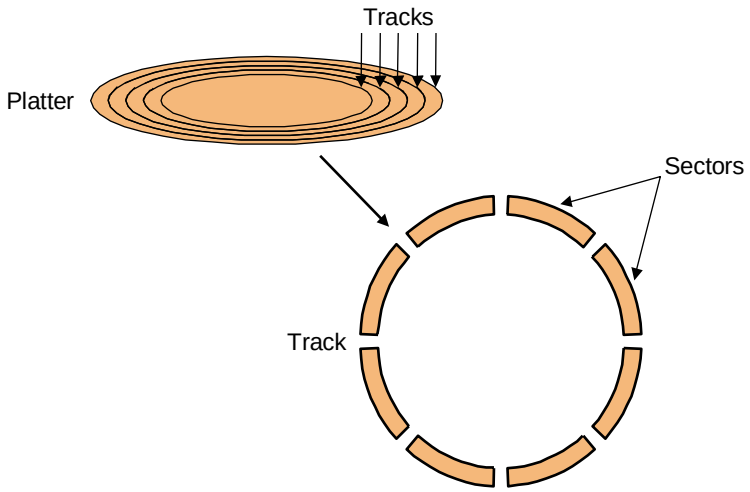
Hard drives

- The figure below shows the ugly guts of a hard disk.
  - Data is stored on double-sided magnetic disks called platters.
  - Each platter is arranged like a record, with many concentric tracks.
  - Tracks are further divided into individual sectors, which are the basic unit of data transfer.
  - Each surface has a read/write head like the arm on a record player, but all the heads are connected and move together.
- A 75GB IBM Deskstar has roughly:
  - 5 platters (10 surfaces),
  - 27,000 tracks per surface,
  - 512 sectors per track, and
  - 512 bytes per sector.



Accessing data on a hard disk

- Accessing a sector on a track on a hard disk takes a lot of time!
  - Seek time measures the delay for the disk head to reach the track.
  - A rotational delay accounts for the time to get to the right sector.
  - The transfer time is how long the actual data read or write takes.
  - There may be additional overhead for the operating system or the controller hardware on the hard disk drive.
- Rotational speed, measured in revolutions per minute or RPM, partially determines the rotational delay and transfer time.



Estimating disk latencies (seek time)

- Manufacturers often report *average* seek times of 8-10ms.
  - These times average the time to seek from any track to any other track.
- In practice, seek times are often much better.
  - For example, if the head is already on or near the desired track, then seek time is much smaller. In other words, locality is important!
  - Actual average seek times are often just 2-3ms.

Estimating Disk Latencies (rotational latency)

- Once the head is in place, we need to wait until the right sector is underneath the head.
  - This may require as little as no time (reading consecutive sectors) or as much as a full rotation (dang, just missed it).
  - On average, for random reads/writes, we can assume that the disk spins halfway on average.

- Rotational delay depends partly on how fast the disk platters spin.

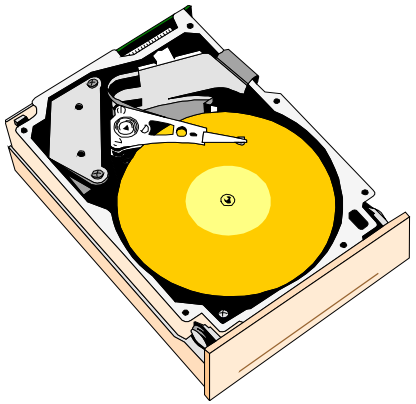
  Average rotational delay = 0.5 rotations / rotational speed

  - For example, a 5400 RPM disk has an average rotational delay of:

    0.5 rotations / (5400 rotations/minute) = 5.55ms

Estimating disk times

- The overall response time is the sum of the seek time, rotational delay, transfer time, and overhead.
- Assume a disk has the following specifications.
  - An average seek time of 9ms
  - A 5400 RPM rotational speed
  - A 10MB/s average transfer rate
  - 2ms of overheads



- How long does it take to read a random 1,024 byte sector?
  - The average rotational delay is 5.55ms.
  - The transfer time will be about (1024 bytes / 10 MB/s) = 0.1ms.
  - The response time is then 9ms + 5.55ms + 0.1ms + 2ms = 16.7ms. That's 16,700,000 cycles for a 1GHz processor!
- One possible measure of throughput would be the number of random sectors that can be read in one second.

(1 sector / 16.7ms) x (1000ms / 1s) = 60 sectors/second.

Estimating disk times

- The overall response time is the sum of the seek time, rotational delay, transfer time, and overhead.
- Assume a disk has the following specifications.
  - An average seek time of 3ms
  - A 6000 RPM rotational speed
  - A 10MB/s average transfer rate
  - 2ms of overheads
- How long does it take to read a random 1,024 byte sector?
  - The average rotational delay is:
  - The transfer time will be about:
  - The response time is then:

- How long would it take to read a whole track (512 sectors) selected at random, if the sectors could be read in any order?

Parallel I/O

- Many hardware systems use parallelism for increased speed.
  - Pipelined processors include extra hardware so they can execute multiple instructions simultaneously.
  - Dividing memory into banks lets us access several words at once.
- A redundant array of inexpensive disks or RAID system allows access to several hard drives at once, for increased bandwidth.
  - The picture below shows a single data file with fifteen sectors denoted A-O, which are "striped" across four disks.