Operating System

The OS is JUST A PROGRAM

- but it runs in SUPERVISOR state
  - access to PHYSICAL addresses
  - access to special registers (like page table register)
  - all IO devices, etc.
- whereas ordinary programs run in USER state
  - only access to VIRTUAL addresses through page tables
  - normally no access to IO devices

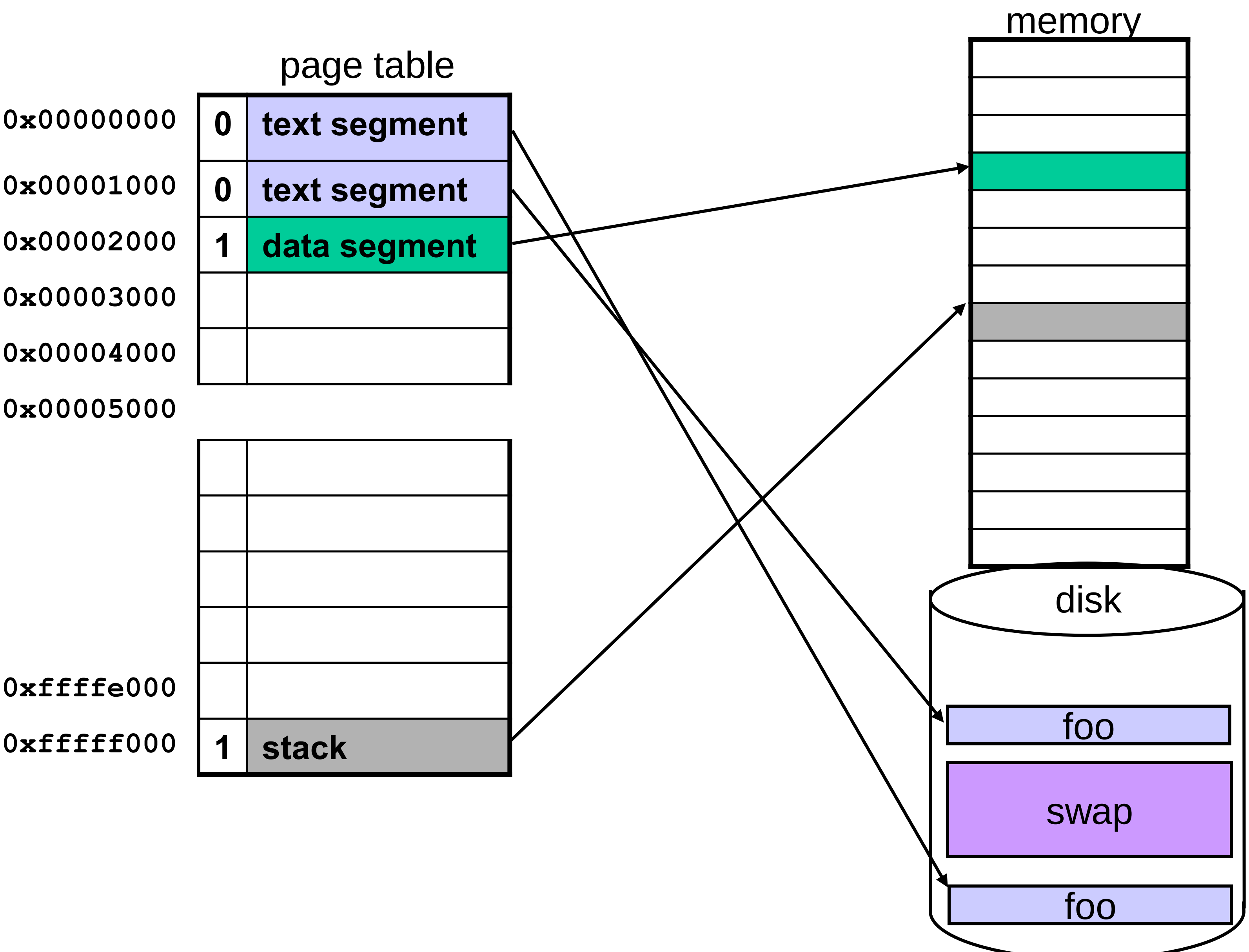
Programs ask the OS for services (syscall)

- give me more memory
- read/write data from/to disk
- put pixel on screen
- give me the next character from the keyboard

OS Execution

- The OS keeps a PROCESS TABLE of all running programs
  - disk location of executable
  - memory location of page tables
  - priority
  - current status (running, waiting ready, waiting on an event, etc.)
  - PID (process ID) a number assigned to the process
- A PROCESS is an independent program running in its own memory space
- The OS allocates a new entry in the PROCESS TABLE
- And sets up the PAGE TABLE for the new process

Initial Page Table



Program Startup

Now everything is ready

- The PROCESS TABLE entry has been set up
- The PAGE TABLE for the process has been initialized
- The TEXT SEGMENT is out on disk
- The DATA SEGMENT is in memory
- The STACK SEGMENT has been allocated 1 PAGE

The OS is ready to take the leap of faith

ONLY ONE program runs at a time

When your program is running the OS is not

To run your program and maintain control the OS must trust that it will eventually regain control

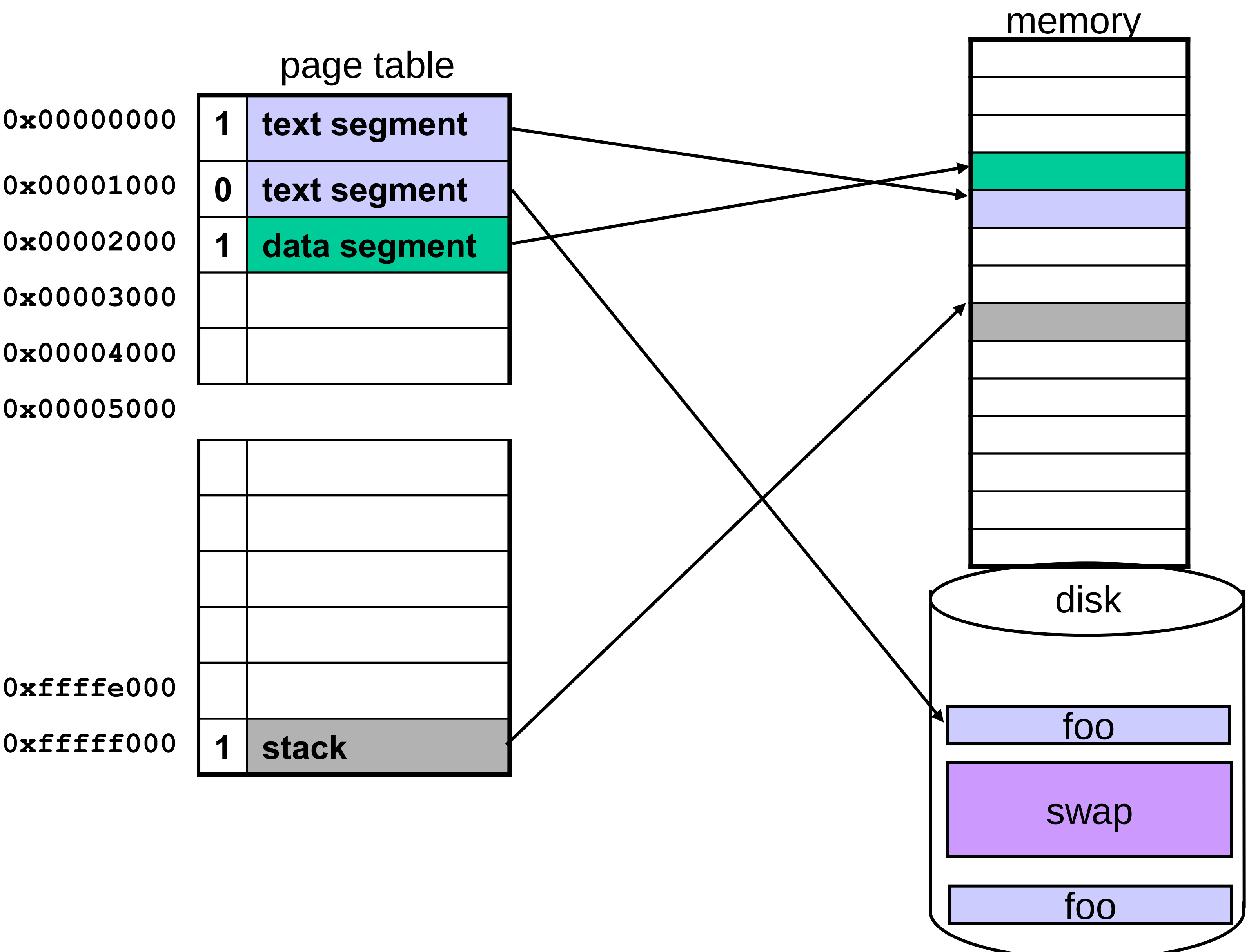
- when the program asks for a service
- when the program does something illegal
- when a timer goes off

Page Fault in the Text

When we branch to the beginning of “main” we get a page fault

So the OS copies the first page of the TEXT of main to a free page in memory

Page Fault in the Text



Allocate a block of memory

Now suppose the first thing our program needs to do is get 6k of memory for an array

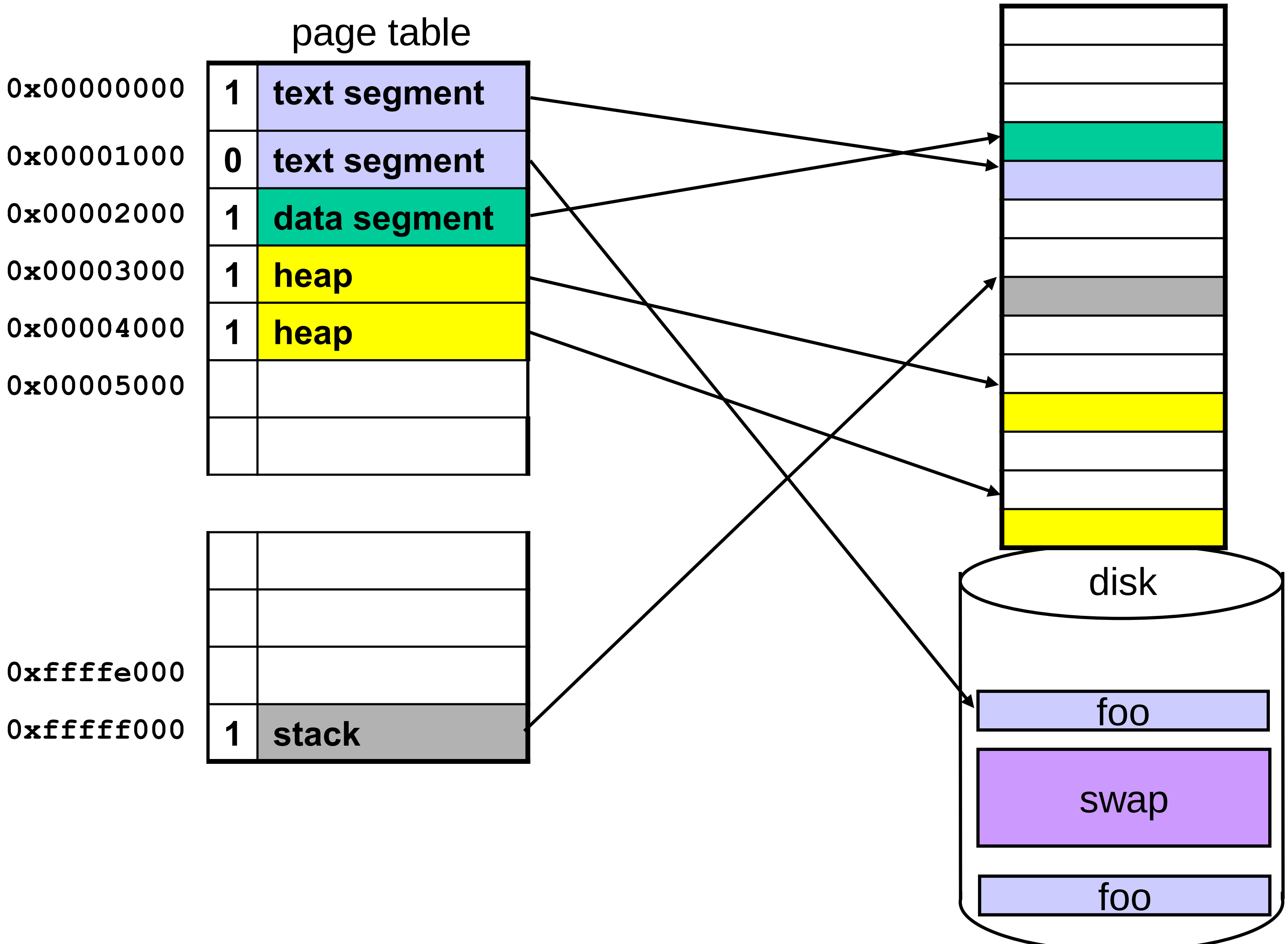
The program uses “new” to make an array

Down inside “new” it calls “malloc”

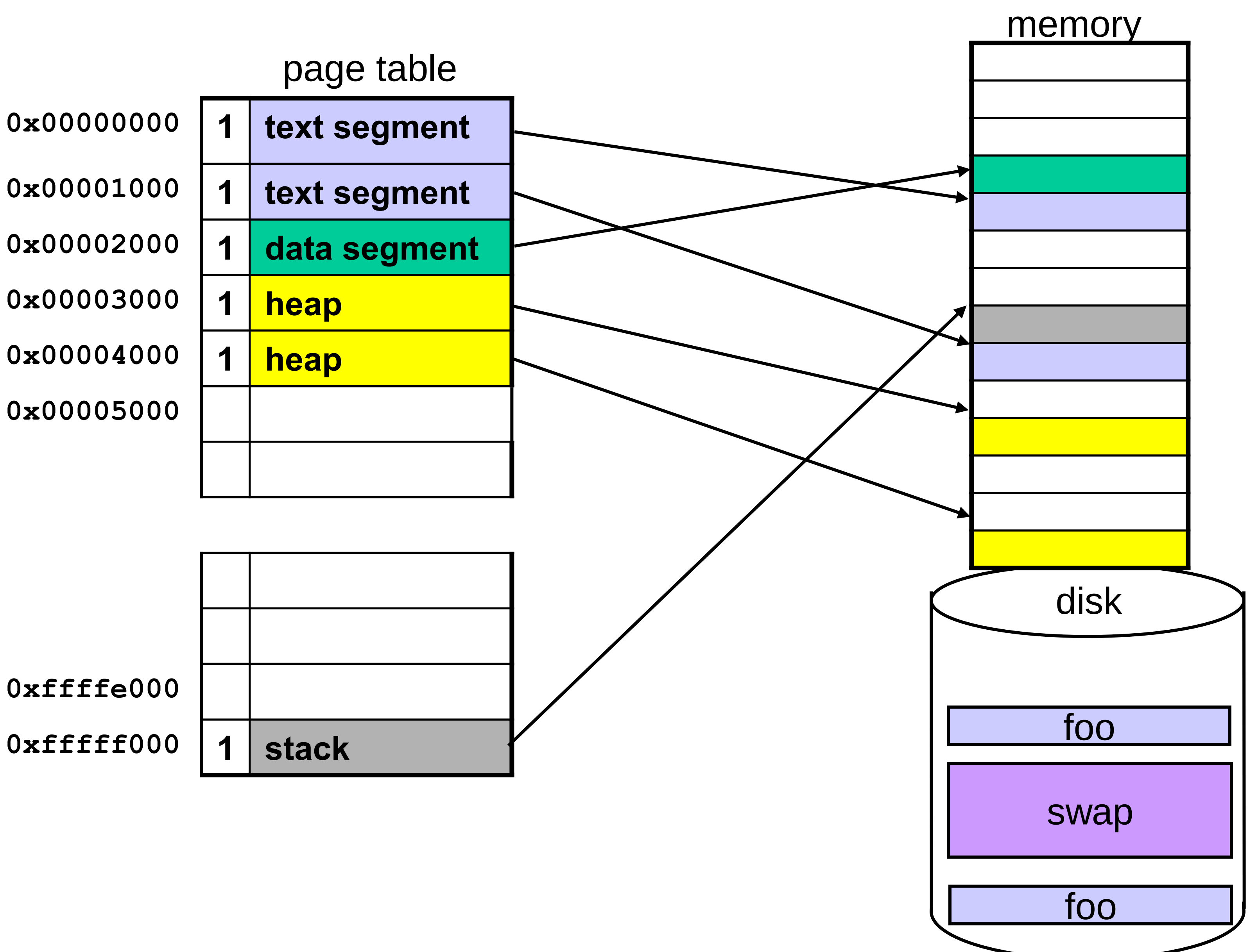
Down inside “malloc” it uses a system call to ask the OS for memory

The OS will have to find 2 pages to hold 6k

Allocate a block of memory



Fault in the other page of TEXT



Grow the stack

Now our program needs more stack space

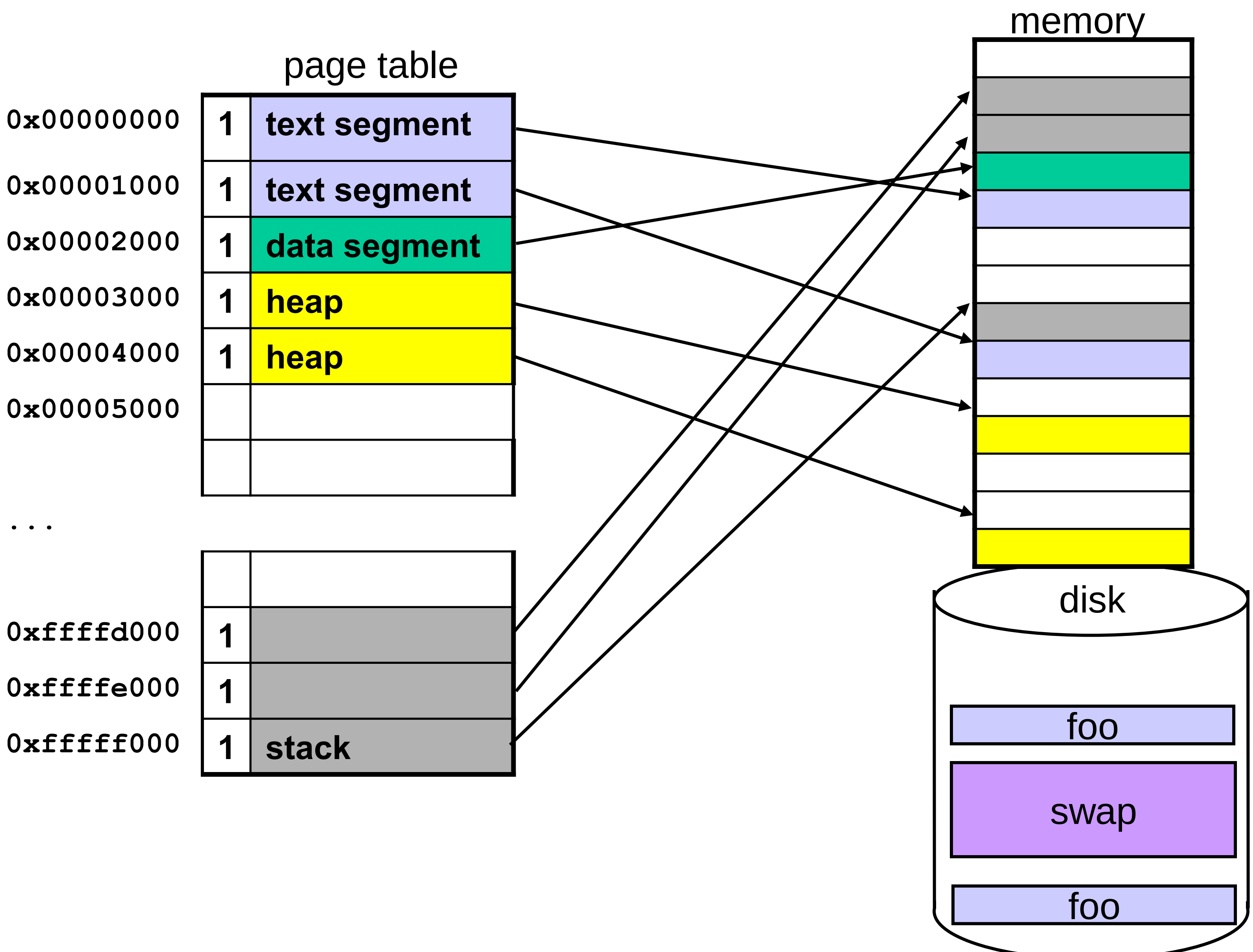
Perhaps it has to call a recursive function to traverse a complex data structure

Or perhaps the user declares an “automatic” array like

double work[1000];

which needs 8000 bytes of memory

Grow the stack



Get partially paged out

Sometime later, some other program running on the system needs more memory

It asks the OS

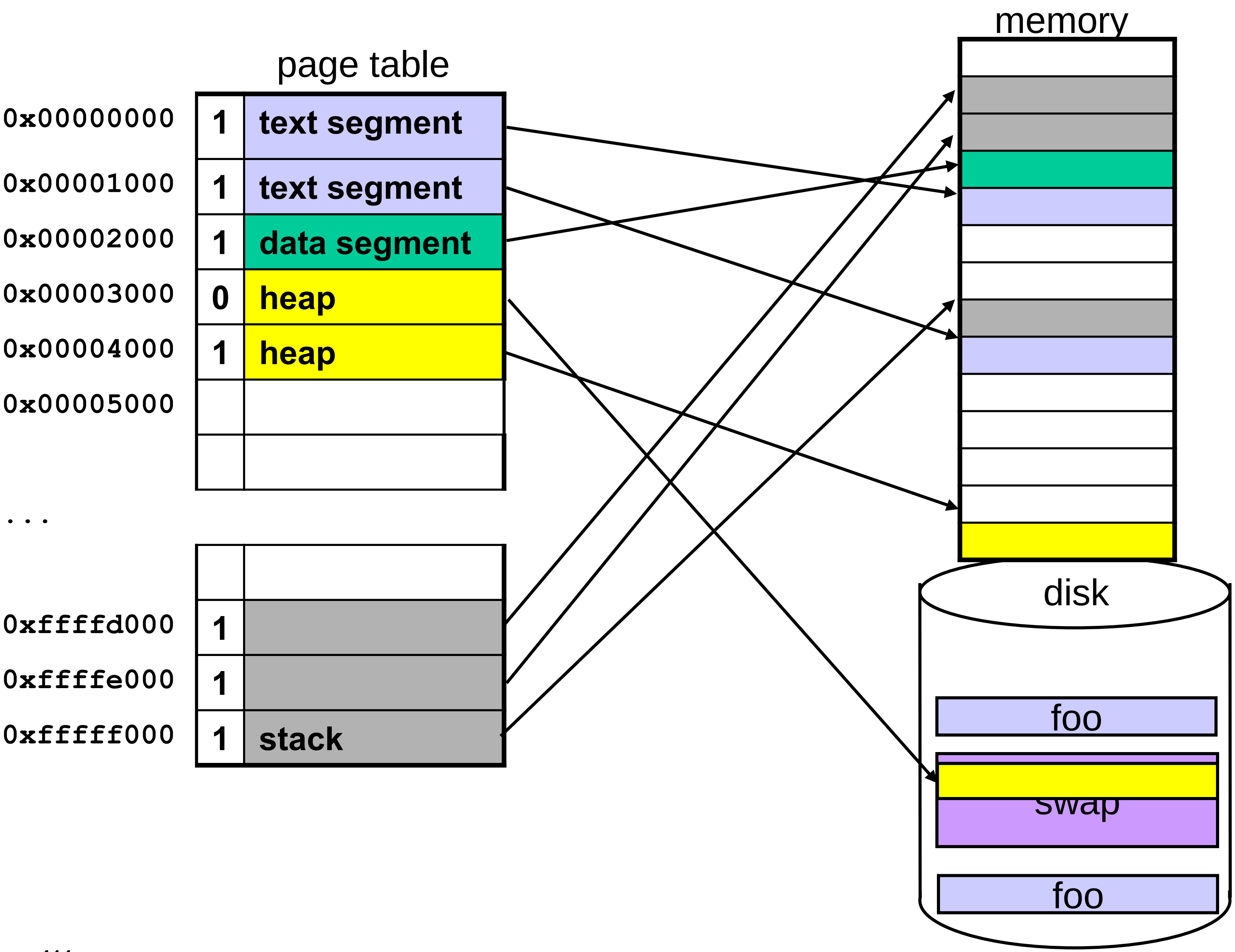
The OS realizes that not enough physical memory remains available

So the OS chooses to PAGE OUT one page from our program

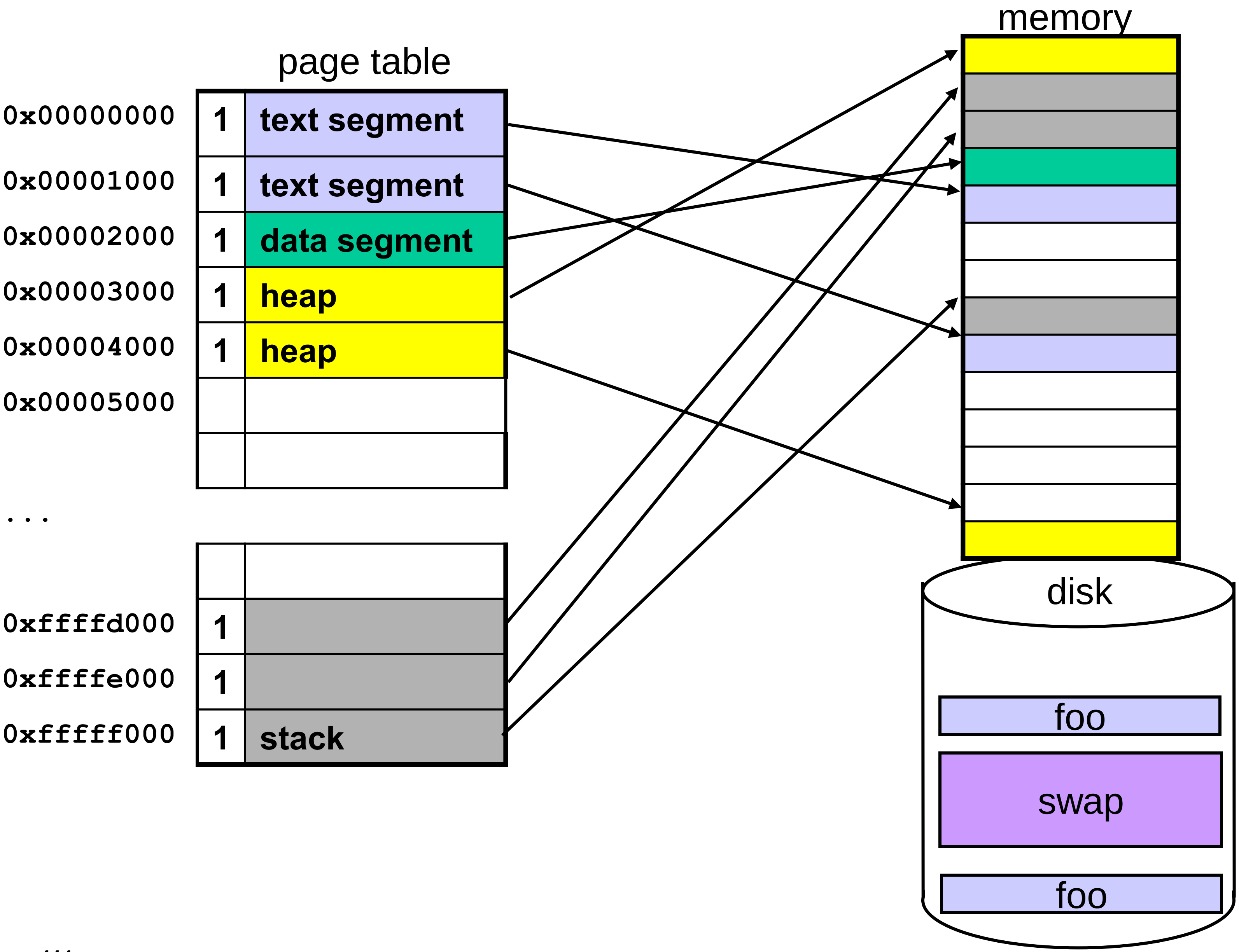
It would choose one that hasn't been used for a while

- like possibly one of the heap segments

Partially Paged Out



Later we need that page



Exit

- Finally our program exits
- It calls the “exit” system call to notify the OS that it is done
- The OS puts the memory back on the free list
- Cleans up the PAGE TABLE and PROCESS TABLE
- And goes on about its business...

Interrupts

How does the CPU manage SLOW **I/O devices?**

Programmed I/O

Interrupt Driven I/O

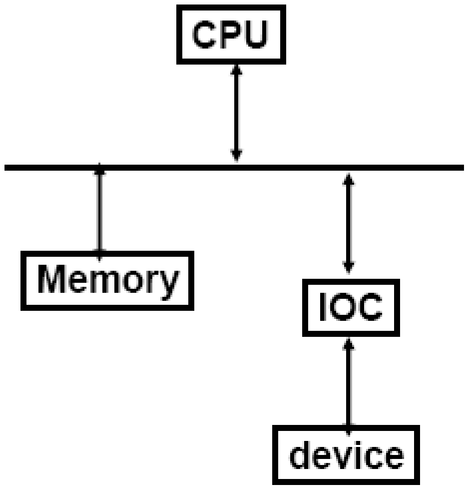
Polling

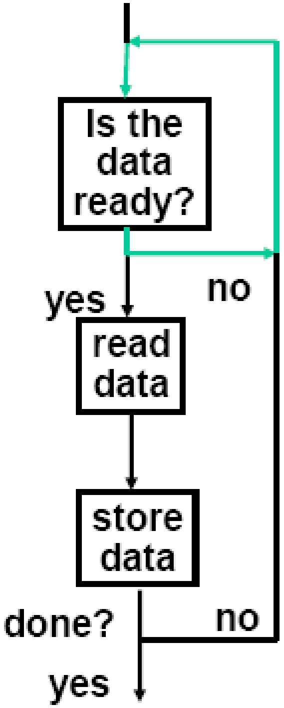
Advantages

- Simple
- No surprises
- Processor in full control

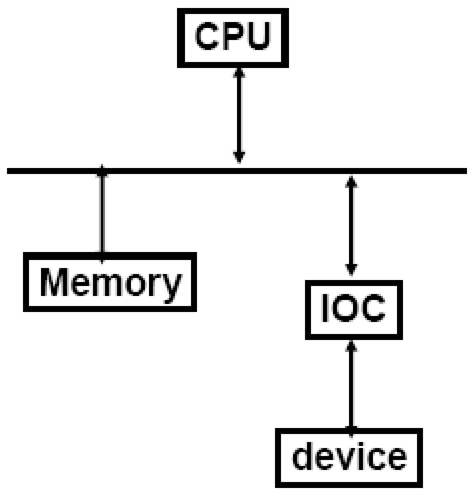
Disadvantages

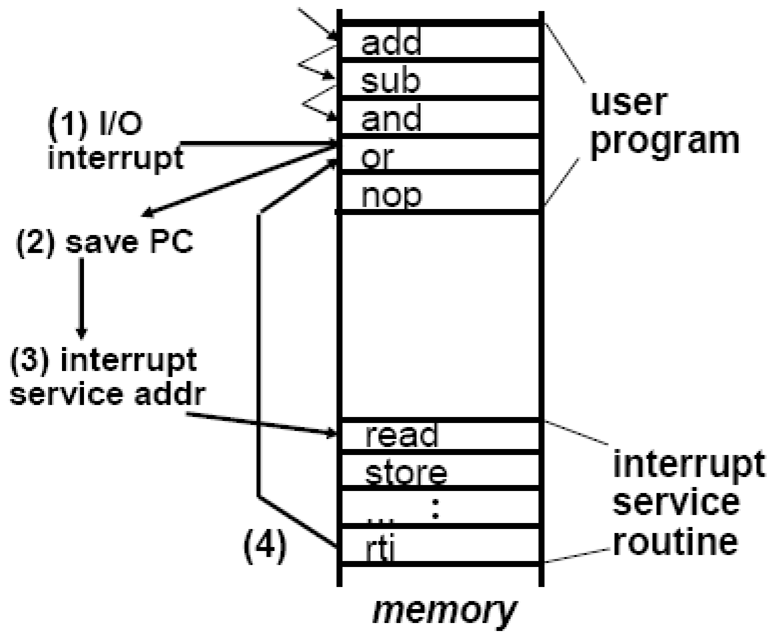
- Polling can waste lots of time





Interrupt Driven I/O





Advantage

- CPU only bothered when actually needed

Disadvantage

- Can occur at surprising or inconvenient times
- Have to save and restore state

MIPS Exceptions

Reset

Hardware Errors (Bus Error, Cache Error)

External Interrupt (6 inputs)

Address Error

Reserved Instruction

TLB Miss

System Call

Breakpoint

Trap

Integer Overflow

Floating Point Error

Timer

And a few more

Exception Processing

EPC gets address of faulty instruction or of next instruction depending on type of exception

Switch to SUPERVISOR mode

Jump to a new location based on type of exception

- PC ← FFFF FFFF BFC0 0000 for Reset
- PC ← FFFF FFFF BFC0 0300 for Hardware error
- PC ← FFFF FFFF BFC0 0380 for external interrupts
- PC ← FFFF FFFF BFC0 0400 for …

Save registers

Examine the "cause" register to find out why you came here

Branch to code to do the right thing

Quick overview of I/O devices

This is the “rest” of the computer

- Used to be called “peripherals”
- …but that term does not do justice to them!

Magnetic Disk

Long term, nonvolatile storage

Large, inexpensive, and slow

Rotating platter(s) coated with magnetic material

Use a movable read/write head to access

When magnetized region zips past coils in head, a tiny signal is produced

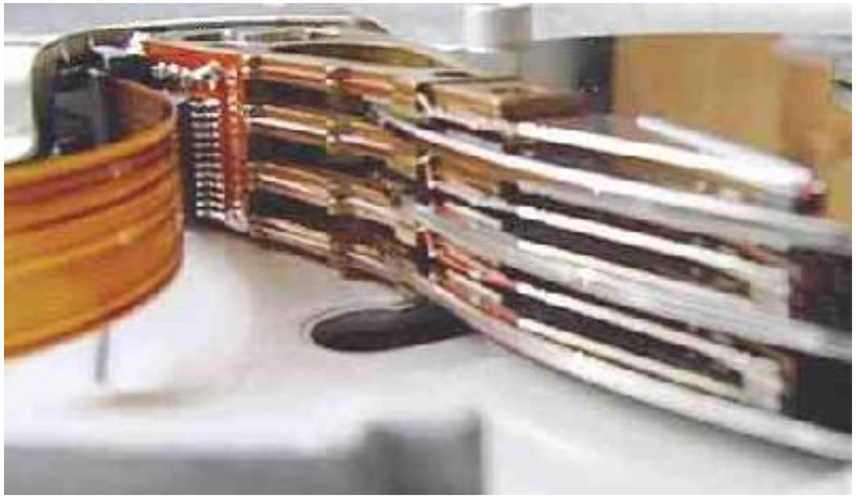
Force current through coils to generate magnetic field to magnetize tiny regions on the disk

Use feedback to keep the head in the right place

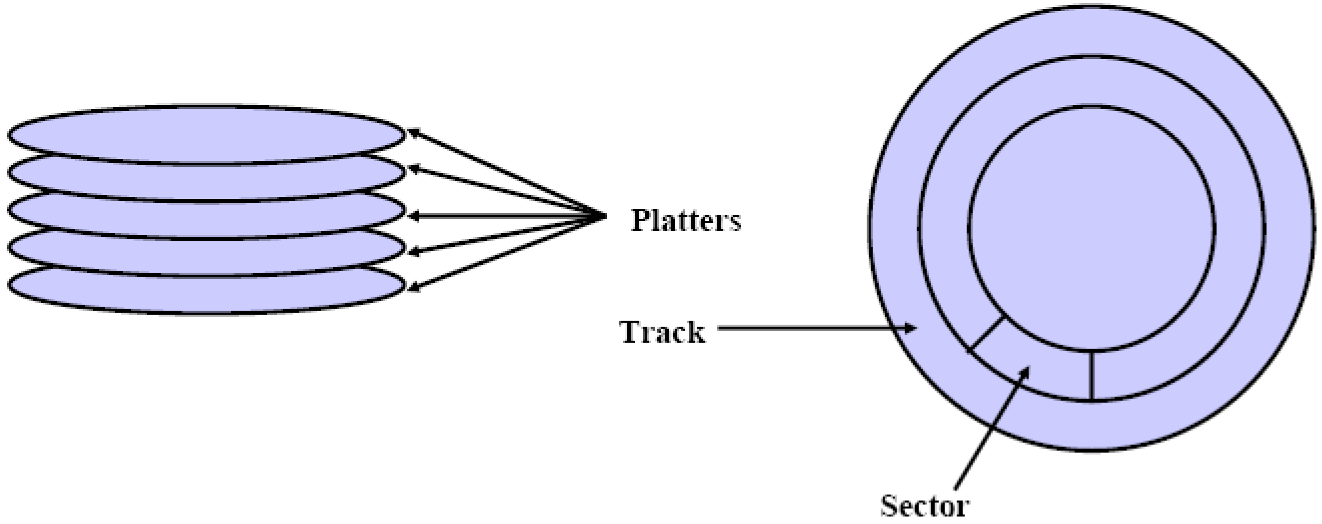
Magnetic Disks: Outside

Inside

Platters and Heads

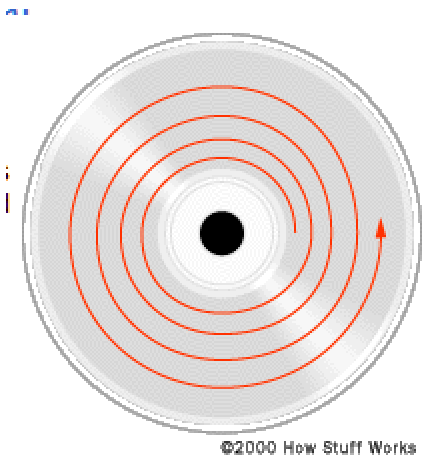


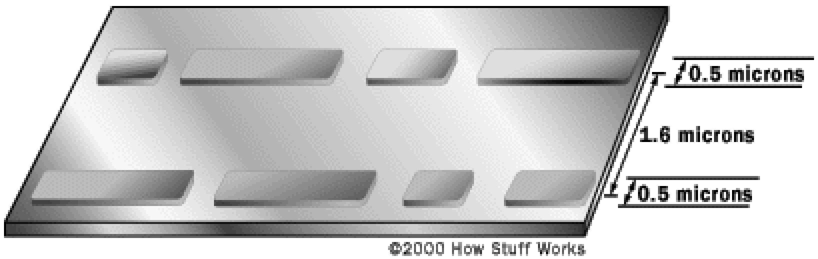
Magnetic Disk Organization

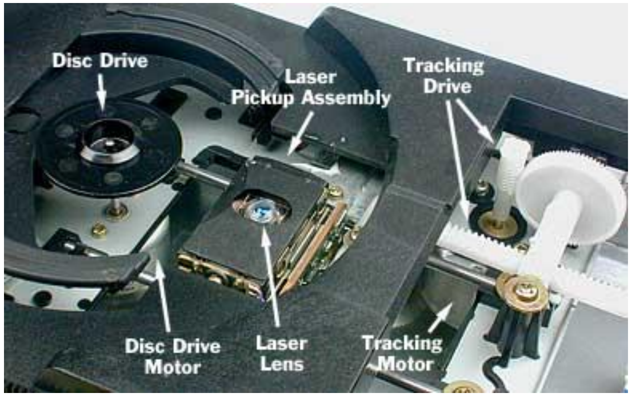


- Cylinder: All tracks under head with arm in a fixed position
- Read/Write time has 3 components
  - Seek time to move the arm
  - Rotational latency: wait for the desired sector to come by
  - Transfer time: transfer bits

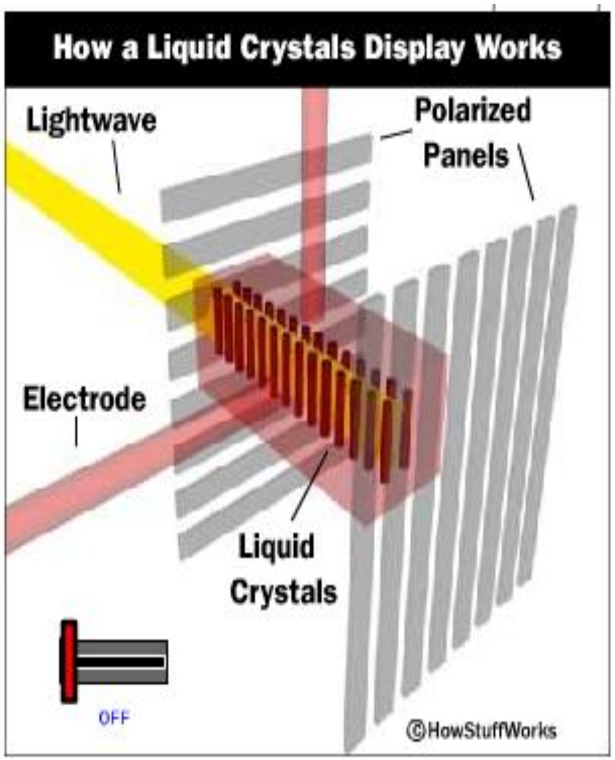
CD

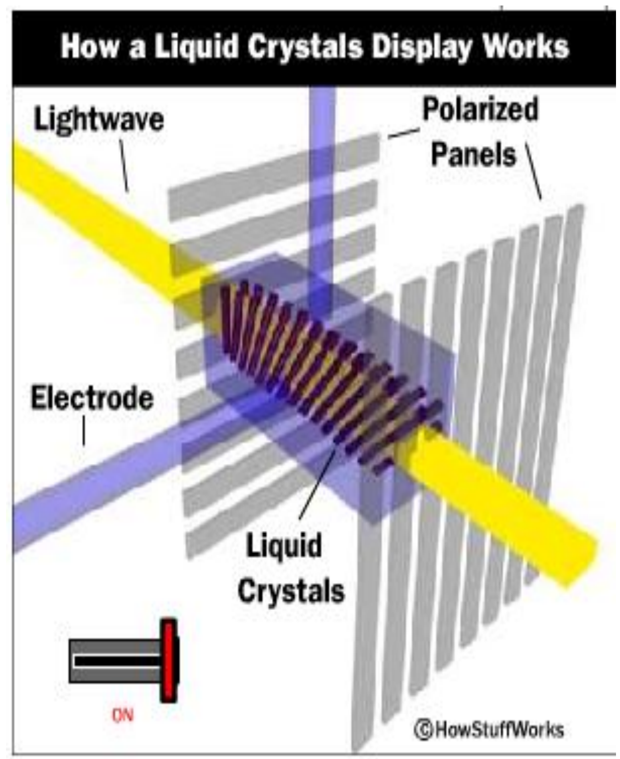

©2000 How Stuff Works


0.5 microns
1.6 microns
0.5 microns
©2000 How Stuff Works

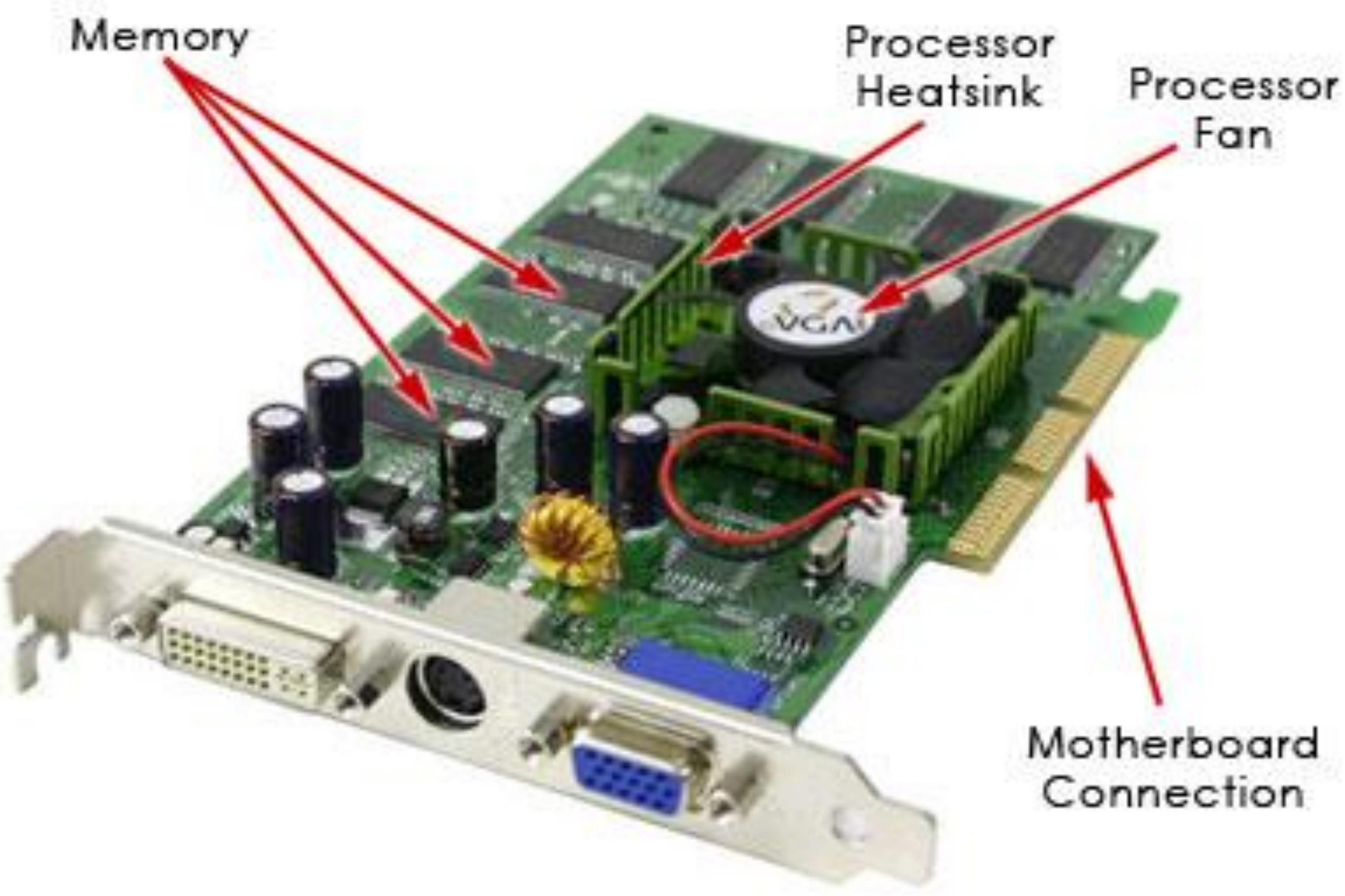

Disc Drive
Laser Pickup Assembly
Tracking Drive
Disc Drive Motor
Laser Lens
Tracking Motor

LCD


How a Liquid Crystals Display Works
Lightwave
Polarized
Panels
Electrode
Liquid
Crystals
OFF
©HowStuffWorks

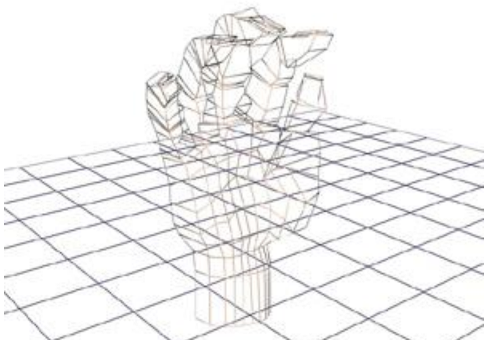

How a Liquid Crystals Display Works
Lightwave
Polarized
Panels
Electrode
Liquid
Crystals
ON
©HowStuffWorks

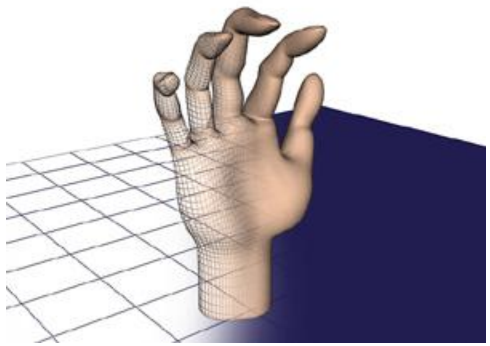
Graphics Cards

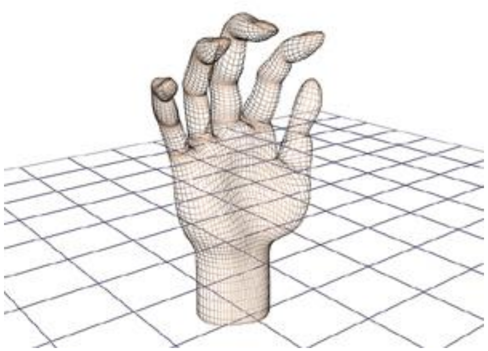


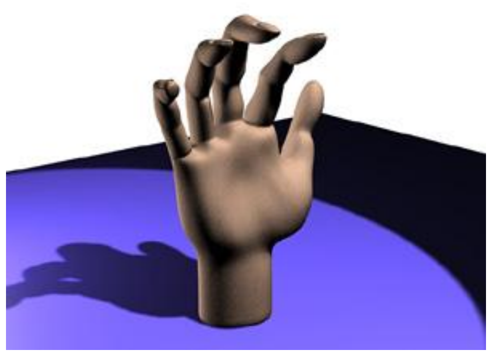
Polygons to Surfaces

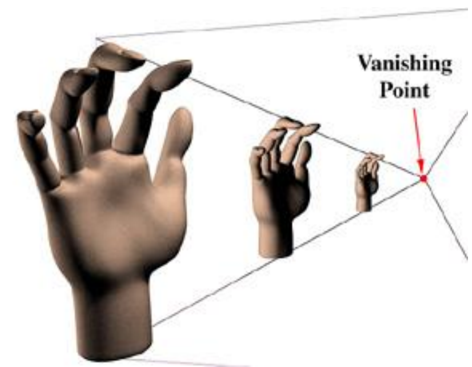
- Numerical coordinates specify vertex positions in 3D
- Matrix multiply transforms 3D coordinates to eye coordinates
- Divide projects 3D to 2D in perspective
- Pixel processors fill polygons with appropriate colors based on lighting model



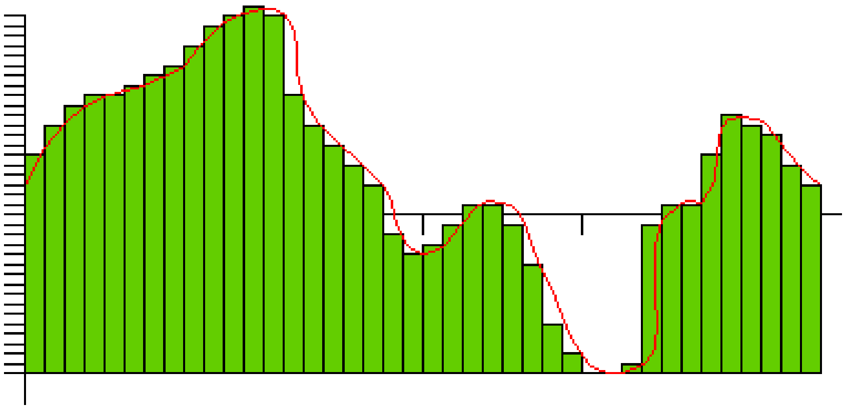








Sound



Sound is variations in air pressure

A microphone converts these into an analog electrical signal

An analog-to-digital converter samples this at frequent intervals

The resulting numbers are stored in a file (.wav)

On playback a digital-to-analog converter changes these numbers into an analog electrical signal

And the moving cone of a speaker converts this into varying air pressure

That's it folks!

You now have a pretty good idea about:

- **How computers are designed and how they work**
  - **How data and instructions are represented**
  - **How arithmetic and logic operations are performed**
  - **How ALU and control circuits are implemented**
  - **How registers and the memory hierarchy are implemented**
  - **How performance is measured**
  - **How performance is increased via pipelining, caching**
  - **How VM works.**
  - **(briefly) What the rest of the computer looks like (disks, sound, etc.)**