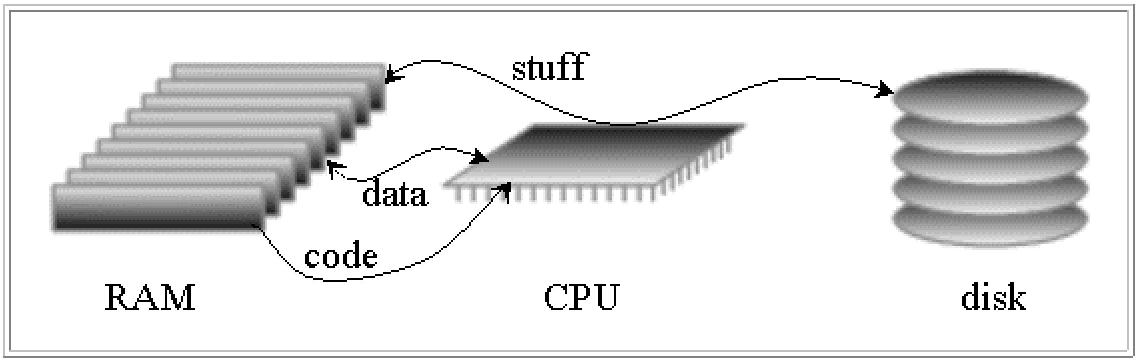
Computer Architecture

The usual state of affairs is a little weird:



- A computer's disk contains some stuff:
  - some of it represents *data*
  - some of it represents *code*
- Before the CPU can use the stuff, it has to stick it in RAM

Q: *Why not just run code and use data straight off the disk?*

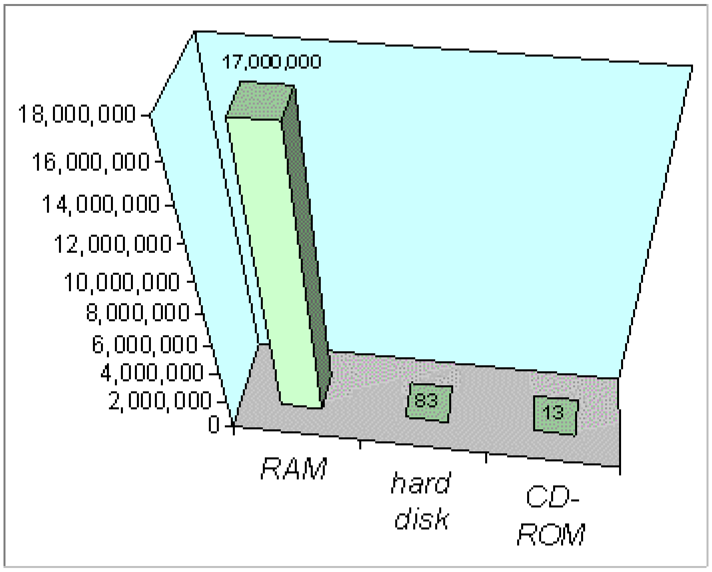
A:

Q: *Okay, then, why not just store everything in RAM all the time?*

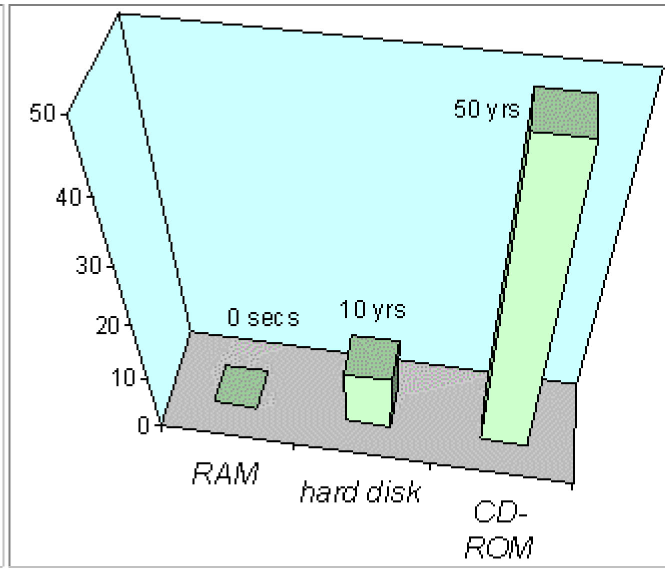
A:

Speed vs. Persistence

Speed (in bytes per second)



Amount of time data will persist after the power is shut off



Specifically...

My PC

Pentium II	*333MHz*
32× CD-ROM	75 ms access
1.44 MB floppy	90 ms access
4.5 GB hard disk	12 ms access
128 MB RAM	60 ns access

Given these specs, how many random bytes can you get from each device in *one second?*

	CD	*floppy*	*hard disk*	*RAM*
bytes/sec				

How long would it take to get 10 megs at random off each device?

	CD	*floppy*	*hard disk*	*RAM*
time				

These tables explain why hard disks and CD-ROM drives are not used for computer storage: *they're too slow!*

But Seriously Folks...

- The program `memio` (written in C) grabs one byte at a time from random locations within a 100MB space, 10,485,760 times (10 megatimes).
  - *Total time for the move: < ½ s*

(that’s about what we’d expect)

- The program `fileio` (written in C) grabs one byte at a time from random locations in a 100MB file.
  - *Total time to move 10,485 random bytes: ~ 90 s*
  - *Total time to move 104,857 random bytes: ~ 480 s*
  - *Total time to move 1,048,576 random bytes: ~ 4000 s*
  - *Total time to move 10,485,760 random bytes: ~ ? s*

(dunno... I got bored at this point)

- The program `seqfile` (written in C) grabs 10,485,760 bytes one at a time *in order* starting at a random location within a 100MB file
  - *Total time for the move: ~ 60 s*

(that's about 1500× faster than random)

What to Do

1. ***maximize*** use of RAM for program operations
   - if you know that you're going to be bouncing around some big block of data, stick it in RAM first

2. ***minimize*** number of disk accesses
   - going to disk once for 1,000 bytes is better than going to disk 1,000 times for 1 byte.

3. ***organize*** information carefully on disk
   - find a way to organize data so you can jump to the right place on disk instead of having to search through large files.

4. ***optimize*** systems to take advantage of 1 - 3
   - write your code and data structures to take advantage of good memory management, disk management and file organization.

This course is where we find out how to do 1 - 4

What We'll Do

1. Housekeeping, introduction, motivation
2. Basic file structures, basic operations on files
3. Secondary storage, physical storage devices
4. Managing secondary storage
5. The primary/secondary storage interface, managing primary storage
6. File compression
7. Storage optimization
8. Sorting, searching, merging
9. Indexing
10. B-trees
11. B+-trees
12. Hashing, extendible hashing