Memory Hierarchy
& Data Locality

CSE 373
Data Structures & Algorithms
Ruth Anderson

Today's Outline

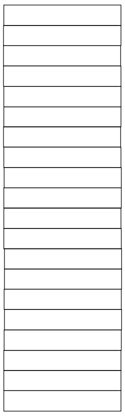
- **Admin**:
  - HW #4 Partner Selection - due TONIGHT, October 31 at 11pm – send email to Tanvir
- **Today**
  - **Hashing**
  - **Memory Hierarchy and Locality**

Why do we need to know about the memory hierarchy/locality?

- One of the assumptions that Big-Oh makes is that *all operations take the same amount of time*.
- Is that really true?

Where are these values in memory?

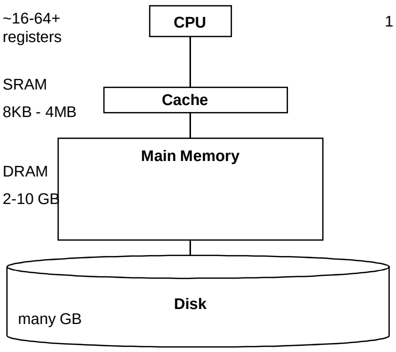
```
int x = 8;
int y = 2 * x;

int[] a = new int[1000];
z = a[0] + a[1] + a[999];

ListNode top = new ListNode(7);
top.next = new ListNode(24);
ListNode temp = top.next;
```



Definitions

Cycle – (for our purposes) the time it takes to execute a single simple instruction. (ex. Add 2 registers together)

Memory Latency – time it takes to access memory



~16-64+ registers

CPU

SRAM

8KB - 4MB

Cache

DRAM

2-10 GB

Main Memory

Disk

many GB

TIme to access:

1 ns per instruction

Cache

2-10 ns

Main Memory

40-100 ns

Disk a few *milli*seconds

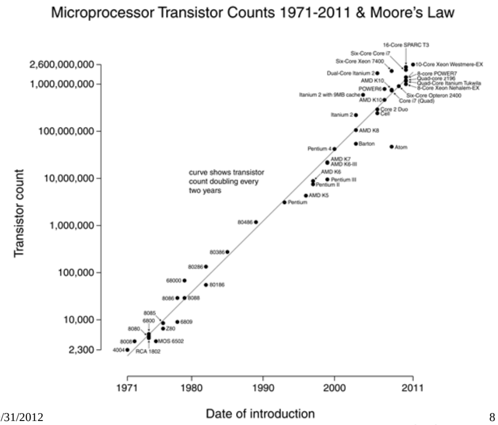
(5-10 *Million* ns)

Morals

It is much faster to do:	Than:
5 million arithmetic ops	1 disk access
2500 L2 cache accesses	1 disk access
400 main memory accesses	1 disk access

Why are computers built this way?

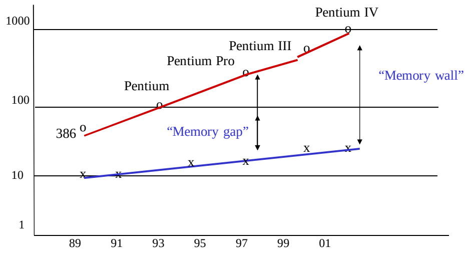
- Physical realities (speed of light, closeness to CPU)
- Cost (price per byte of different technologies)
- Disks get much bigger not much faster
  - Spinning at 7200 RPM accounts for much of the slowness and unlikely to spin faster in the future
- Speedup at higher levels (e.g. a faster processor) makes lower levels *relatively slower.* Argh!



From Wikipedia

Processor-Memory Performance Gap

- x86 CPU speed (100x over 10 years)



What can be done?

- **Goal**: Attempt to reduce the number of accesses to the slower levels.
- How?

So, what can *we* do?

The hardware automatically moves data into the caches from main memory for you

- Replacing items already there
- Algorithms are much faster if "data fits in cache" (often does)

Disk accesses are done by software (e.g., ask operating system to open a file or database to access some data)

So most code "just runs" but sometimes it's worth designing algorithms / data structures with knowledge of memory hierarchy

- And when you do, you often need to know one more thing…

Locality

Temporal Locality (locality in time) – If an item (a location in memory) is referenced, *that same location* will tend to be referenced again soon.

Spatial Locality (locality in space) – If an item is referenced, items *whose addresses are close by* will tend to be referenced soon.

How does data move up the hierarchy?

- Moving data up the memory hierarchy is slow because of *latency* (think distance-to-travel)
  - Since we're making the trip anyway, may as well carpool
    - Get a block of data in the same time it would take to get a byte
  - Sends *nearby memory* because:
    - It's easy
    - Nearby memory is likely to be asked for soon (think fields/arrays)
- Side note: Once a value is in cache, may as well keep it around for awhile; accessed once, a value is more likely to be accessed again in the near future (more likely than some random other value)

Spatial Locality

Temporal locality

Cache Facts

- Each level is a **sub-set** of the level below.

Definitions:

- **Cache Hit** – address requested is in cache
- **Cache Miss** – address requested is NOT in cache
- **Block or Page size** - the number of contiguous bytes moved from disk into memory
- **Cache line size** - the number of contiguous bytes moved from memory into cache

Examples

```
x = a + 6;        x = a[0] + 6;
y = a + 5;        y = a[1] + 5;
z = 8 * a;        z = 8 * a[2];
```

Locality and Data Structures

- Which has (at least the potential for) better spatial locality, arrays or linked lists?

Where is the Locality?

```
for (i = 1; i < 100; i++) {
    a = a * 7;
    b = b + x[i];
    c = y[5] + d;
}
```