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?

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

Morals

It is much faster to do: Than:

- 5 million arithmetic ops
- 2500 L2 cache accesses
- 400 main memory accesses

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!*

Today’s Outline

- **Admin:**
  - HW #4 Partner Selection - due TONIGHT, Nov 2 at 11pm – send email to Svet
- **Today**
  - Hashing
  - Memory Hierarchy and Locality
Moore’s Law

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 is referenced, it 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
    • Gets 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)

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

\[
\begin{align*}
    x &= a + 6; & \quad x &= a[0] + 6; \\
    y &= a + 5; & \quad y &= a[1] + 5; \\
    z &= 8 * a; & \quad z &= 8 * a[2];
\end{align*}
\]

Locality and Data Structures

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