Memory Hierarchy & Data Locality

CSE 373
Data Structures & Algorithms
Ruth Anderson

Where are these values in memory?

```java
int x = 0;
int y = 2 * x;
int[] a = new int[1000];
s = a[0] + a[1] + a[999];
ListNode top = new ListNode(7);
top.next = new ListNode(24);
ListNode temp = top.next;
```

- 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. Then:
- 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 (pace of bytes of different technologies)
- Drives get much bigger but much faster
- Spinning of 7200 RPM accounts for much of the slowdown and inductance to spin faster in the future
- Speedup of higher levels (e.g. a faster processor) makes lower levels relatively slower. Aargh!
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 done)

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 addressors 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
  - **Kilobyte (think: distance-to-travel)**
    - Since we’re making the trip anyway, may as well cache
  - **Load**
    - Get a block of data in one time it would take to get a byte
    - **Block:** Large enough to be cacheable
      - Spatial Locality
        - 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

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

Locality and Data Structures

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

Where is the Locality?

\[
\begin{align*}
    \text{for } (i = 1; i < 100; i++) \{ \\
    \quad a &= a * 7; \\
    \quad b &= b + x[i]; \\
    \quad c &= y[5] + d;
    \}
\end{align*}
\]