Why do we need to know about the memory Hierarchy?

• One of the assumptions that Big-Oh makes is that all operations take the same amount of time
  — This is not quite correct.
Example

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

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

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

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>z[0]</td>
<td>1000</td>
</tr>
<tr>
<td>z[1]</td>
<td>1001</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>z[999]</td>
<td>1999</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>top</td>
<td>3000</td>
</tr>
<tr>
<td></td>
<td>3001</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>val</td>
<td>5000</td>
</tr>
<tr>
<td>next</td>
<td>5001</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>val</td>
<td>7000</td>
</tr>
<tr>
<td>next</td>
<td>7001</td>
</tr>
</tbody>
</table>

address 5000

address 7000

null
Definitions

• **Cycle** (for our purposes): the time it takes to execute a single simple instruction (for example, add 2 registers together)

• **Memory Latency**: The time it takes to access memory
Time to access:

1 ns per instruction

Cache
2-10 ns

Main Memory
40-100 ns

Disk
a few milliseconds
(5-10 Million ns)
Moral of the story

• It is much faster to do:
  – 5 million arithmetic ops than 1 disk access
  – 25000 L2 cache accesses than 1 disk access
  – 400 main memory accesses then 1 disk access

• Why though?
  – Physical realities (speed of light, closeness to CPU)
  – Cost (price per byte of different technologies)
  – Disks get much **bigger** but not much **faster**
    • Spinning at 7200 RPM account for much of the slowness, spinning hard disks are unlikely to get much faster.
    • What about SSDs?
  – Speedup at higher levels (i.e. a faster processor) makes lower level accesses relatively slower. Yikes.
Microprocessor Transistor Counts 1971-2011 & Moore’s Law

The curve shows transistor count doubling every two years.

Date of introduction

Transistor count

11/22/2013
Processor-Memory Performance Gap

![Graph showing the relative performance of CPU frequency and DRAM speeds. The CPU frequency line shows a 2x every 2 years increase, while the DRAM speeds show a 2x every 6 years increase. There is a gap between the two lines, indicating a performance gap.](http://www.mentor.com)
What can we do to optimize?

• Hardware automatically moves data into caches from main memory
  – Replacing items already there
  – Algorithms are much faster if data fits in the cache

• Disk accesses are abstracted away by the operating system

• Code “just runs” but sometimes it’s worth designed algorithms/data structures with knowledge of the memory hierarchy
Località

- **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 as well
How does data move up the hierarchy?

• Moving data up the hierarchy is slow because of latency (distance to travel)
  – Since we are making a trip anyway, might as well carpool!
    • Get a block of data in the same time it takes to get a byte
  – Send nearby memory
    • Because its cheap and easy
    • And spatial locality says it will be asked for soon!

• Once we move something to the cache, keep it around for a while, no rush to get rid of it! (Temporal Locality)
Cache Facts

• Each level is a sub-set of the level below

• Definitions
  – **Cache hit**: address requested already in the cache
  – **Cache miss**: address request NOT in the cache
  – **Block of page size**: the number of contiguous bytes moved from disk to memory
  – **Cache line size**: the number of contiguous bytes moved from memory to the cache
Examples

\[
x = a + 6; \\
y = a + 5; \\
z = 8 * a;
\]
- Miss
- Hit 

Temporal Locality

\[
x = a[0]; \\
y = a[1] + 5; \\
z = 8 * a[2];
\]
- Miss
- Hit 

Spatial Locality
Locality in Data Structure

• Which has the least potential for better spatial locality, arrays or linked lists?
Where is the locality?

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

- = Spatial Locality on locations in array x
- = Temporal Locality