CSE373: Data Structure & Algorithms

Lecture 24: Memory Hierarchy and Data Locality

Aaron Bauer

Winter 2014

Why memory hierarchy/locality?

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

Where are these values in memory?

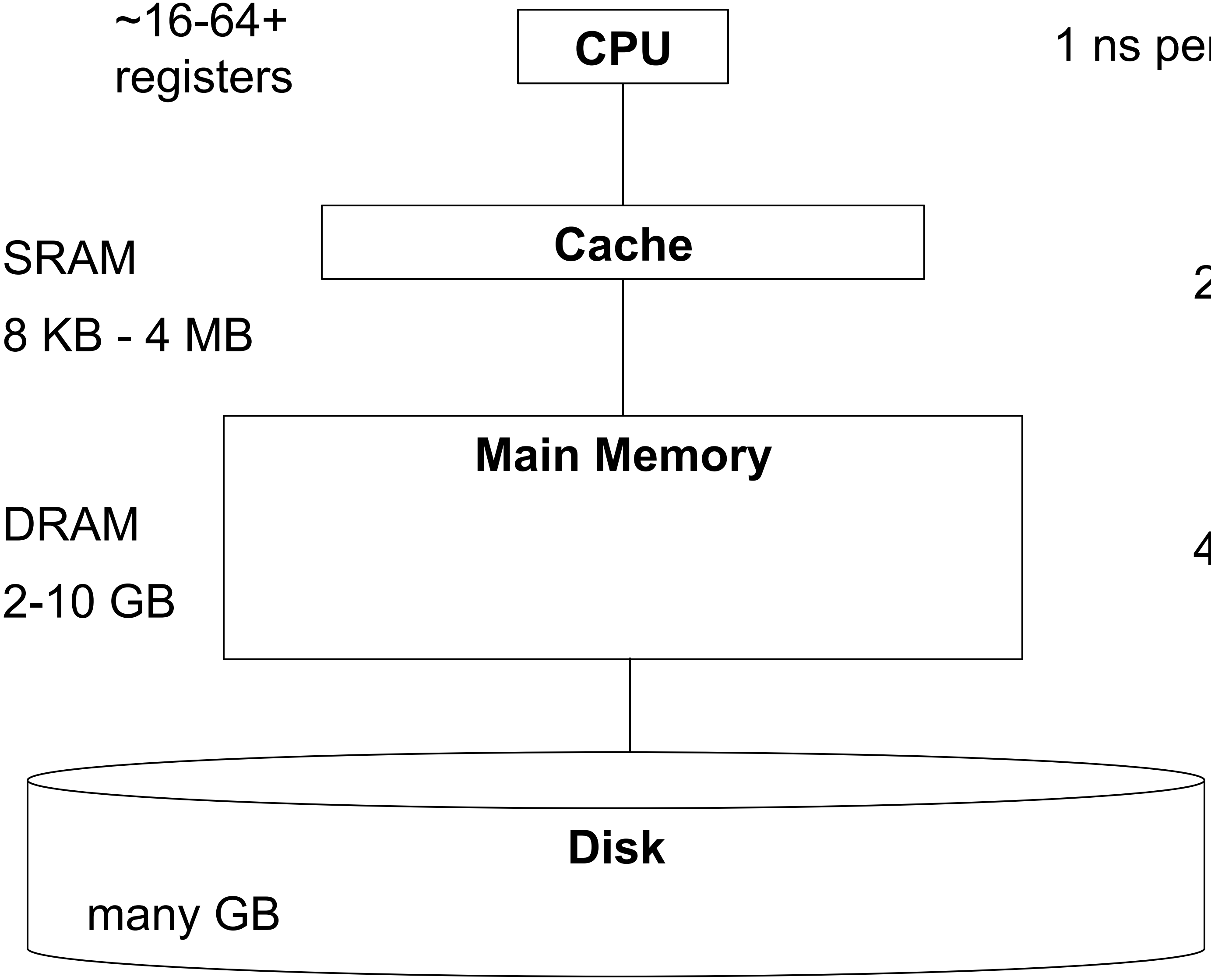
```
int x = 8;
int y = 2 * x;

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;
```

Ref	Loc	Value
x	0	8
y	1	16
	2	…
	…	
a[0]	1000	
a[1]	1001	
…	…	…
a[999]	1999	
	…	
top	3000	5000
	…	
val	5000	7
next	5001	7000
	…	
val	7000	24
next	7001	

Definitions

- A cycle (for our purposes) is the time it takes to execute a single simple instruction (e.g. adding two registers together)
- Memory latency is the time it takes to access memory

Time to access:



~16-64+ registers

CPU

1 ns per instruction

SRAM

8 KB - 4 MB

Cache

2-10 ns

DRAM

2-10 GB

Main Memory

40-100 ns

Disk

many GB

a few *milli*seconds

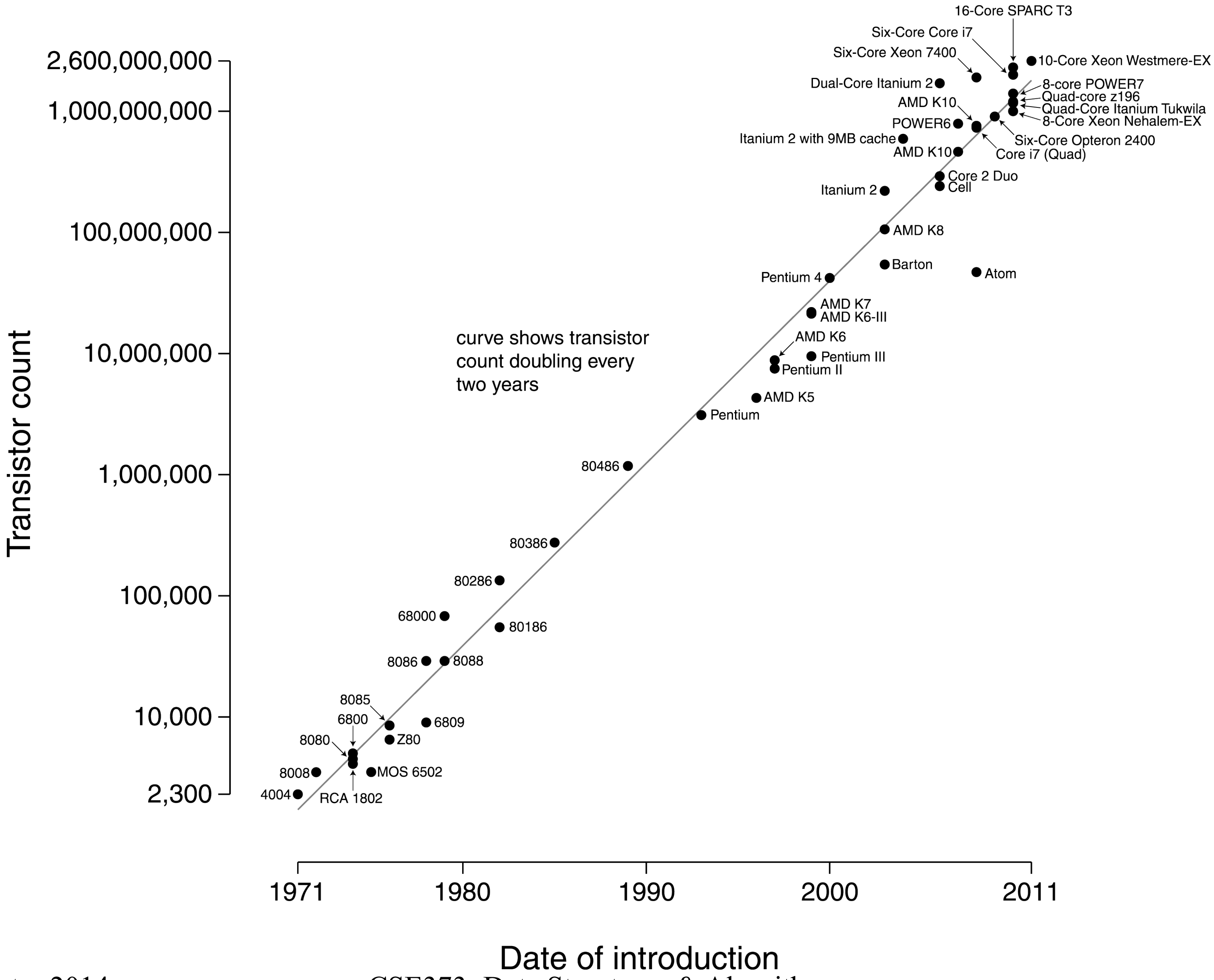
(5-10 million ns)

What does this mean?

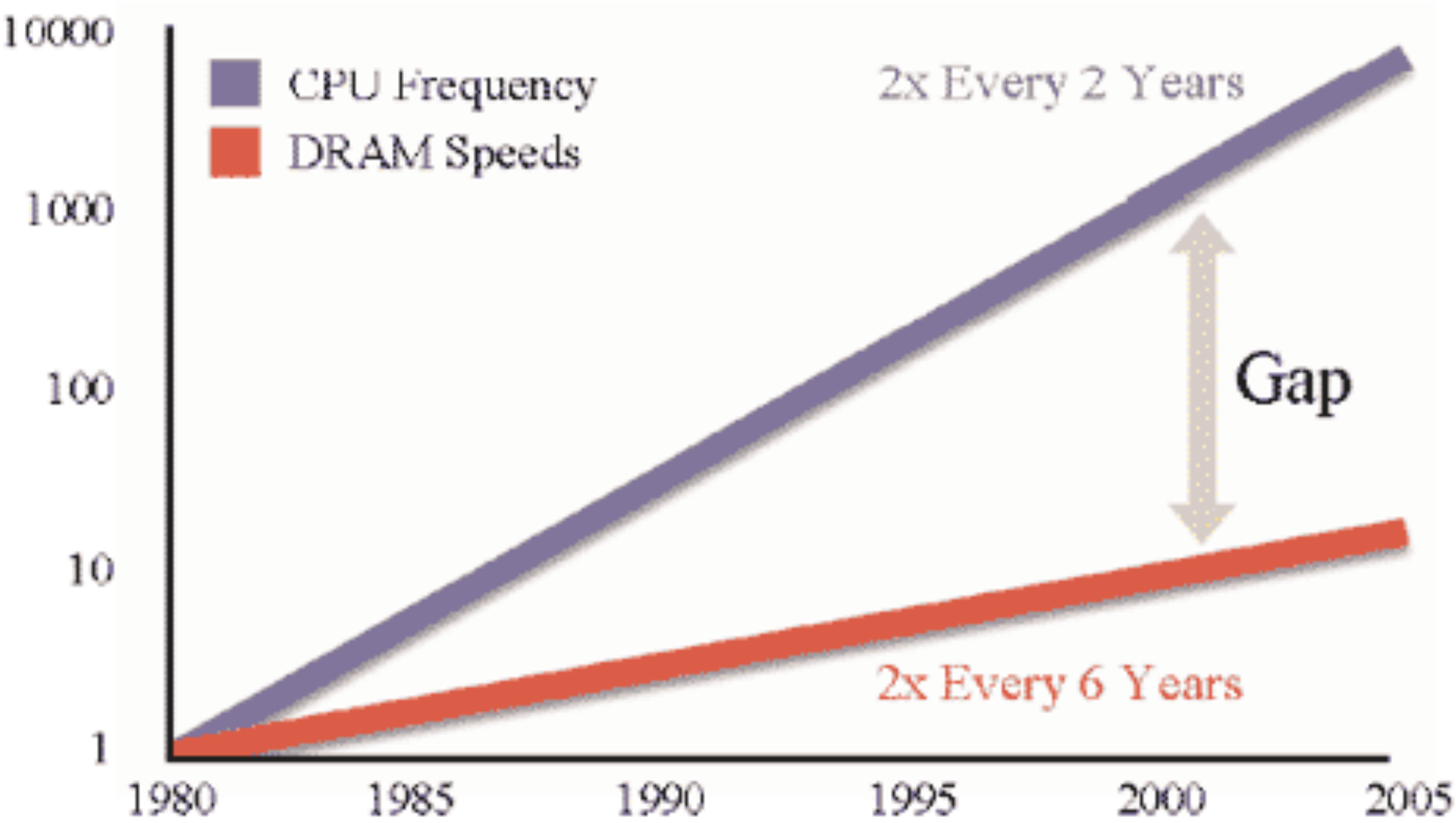
• It is much faster to do:	Than:
5 million arithmetic ops	1 disk access
2500 L2 cache accesses	1 disk access
400 main memory accesses	1 disk access

- Why are computers build this way?
  - Physical realities (speed of light, closeness to CPU)
  - Cost (price per byte of different storage technologies)
  - Under the right circumstances, this kind of hierarchy can simulate storage with access time of highest (fastest) level and size of lowest (largest) level

Microprocessor Transistor Counts 1971-2011 & Moore's Law



Processor-Memory Performance Gap



What can be done?

- **Goal**: attempt to reduce the accesses to slower levels
- How?

So, what can we do?

- The hardware automatically moves data from main memory into the caches 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 records)

- So most code "just runs," but sometimes it's worth designing algorithms / data structures with knowledge of memory hierarchy
  - To do this, we need to understand locality

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** tend to be referenced soon.

How does data move up the hierarchy?

- Moving data up the hierarchy is slow because of *latency* (think distance to travel)
  - Since we’re making the trip anyway, might as well carpool
    - Get a **block** of data in the same time we could get a byte
  - Sends *nearby memory* because
    - It’s easy
    - Likely to be asked for soon (think fields/arrays)

Spatial Locality

- Once a value is in cache, may as well keep it around for a while; accessed once, a value is more likely to be accesses again in the near future (as opposed to some random other value)

Temporal Locality

Cache Facts

- Every level is a **sub-set** of the level below

- Definitions:
  - Cache hit – address requested is in the cache
  - Cache miss – address requested is NOT in the cache
  - Block or page size – the number of contiguous bytes moved from disk to memory
  - Cache line size – the number of contiguous bytes move from memory to cache

Examples

```
x = a + 6          x = a[0] + 6

y = a + 5          y = a[1] + 5

z = 8 * a          z = 8 * a[2]
```

Examples

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

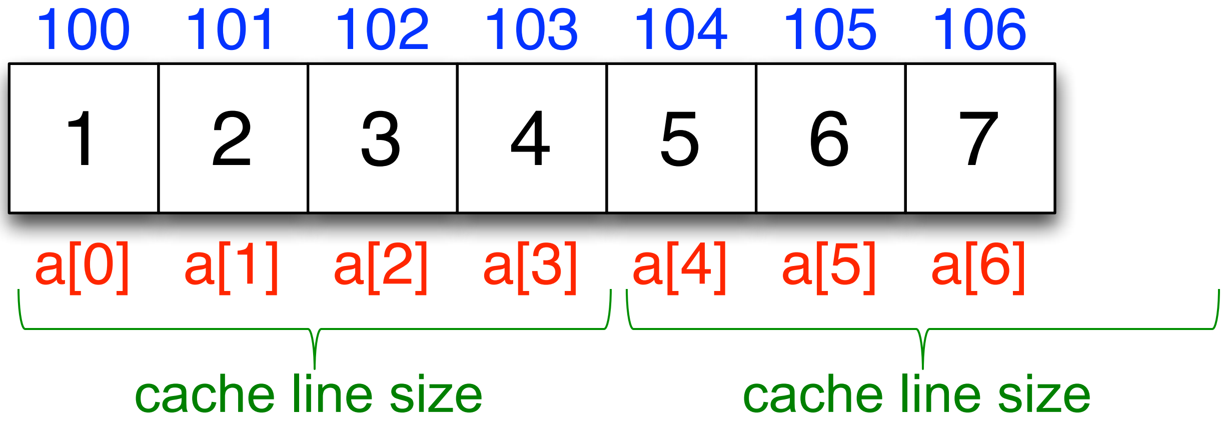
Examples

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

temporal locality	spatial locality

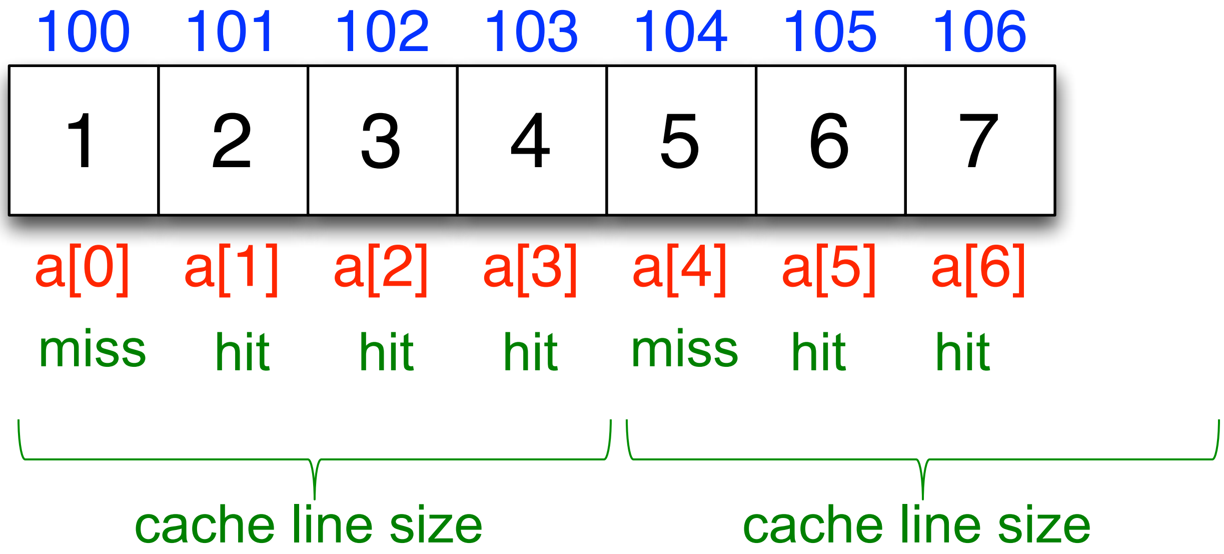
Locality and Data Structures

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



Locality and Data Structures

- Which has (at least the potential) for better spatial locality, arrays or linked lists?
  - e.g. traversing elements

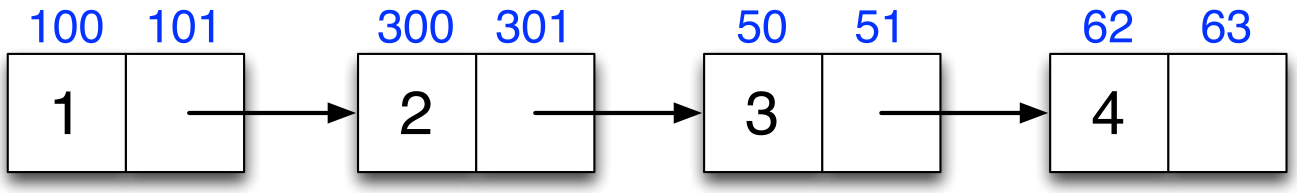


100	101	102	103	104	105	106
1	2	3	4	5	6	7
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
miss	hit	hit	hit	miss	hit	hit

- Only miss on first item in a cache line

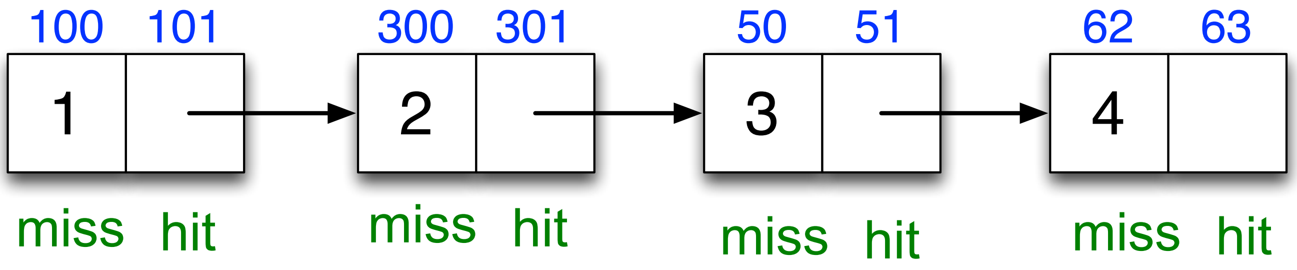
Locality and Data Structures

- Which has (at least the potential) for better spatial locality, arrays or linked lists?
  - e.g. traversing elements



Locality and Data Structures

- Which has (at least the potential) for better spatial locality, arrays or linked lists?
  - e.g. traversing elements



- Miss on **every** item (unless more than one randomly happen to be in the same cache line)

Where is the locality?

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

Where is the locality?

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

Where is the locality?

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

Temporal Locality

Where is the locality?

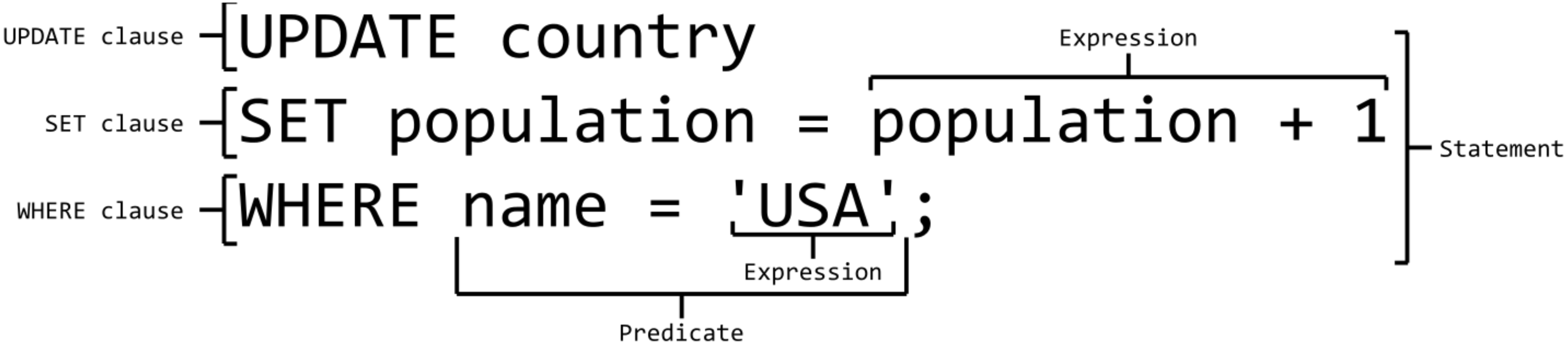
```
for (i = 1; i < 100; i++) {
    a = a * 7;
    b = b + x[i];
    c = y[5] + d;
}
```

Temporal Locality

Spatial Locality

SQL (Structured Query Language)

- Age: 40 years
- Developer: ISO
- Paradigms: declarative
- Type system: static
- Used as a database query language
  - Declarative paradigm perfect for this application



```
UPDATE country
SET population = population + 1
WHERE name = 'USA';
```

- Using SQL is both easy and very powerful
- If you have a lot of data, definitely consider using free database software like MySQL

Python

- Age: 23 years
- Developer: Python Software Foundation
- Paradigm: imperative, object-oriented, functional, procedural
- Type system: dynamic, duck
- Has a Read-Eval-Print-Loop (REPL)
  - Useful for experimenting or one-off tasks
- Scripting language
  - Supports “scripts,” small programs run without compilation
- Often used in web development or scientific/numeric computing
- Variables don’t have types, only values have types
- Whitespace has semantic meaning
- Lack of variable types and compile-time checks mean more may be required of documentation and testing
- Python is my language of choice for accomplishing small tasks

JavaScript

- Age: 19 years
- Developer: Mozilla Foundation
- Paradigm: imperative, object-oriented, functional, procedural
- Type system: dynamic, duck
- Also a scripting language (online/browser REPLs exist)
- Primary client-side language of the web
- Does inheritance through prototypes rather than classes
  - Objects inherit by cloning the behavior of existing objects
- Takes a continue at any cost approach
  - Shared by many web-focused languages (PHP, HTML)
  - Things that would be errors in other languages don't stop execution, and are allowed to fail silently
- JavaScript is nice for simple things, immediately running on the web is great, but doing larger/more complex software is terrible

PHP

- Age: 19 years
- Developer: The PHP Group
- Paradigm: imperative, object-oriented, functional, procedural
- Type system: dynamic
- Works with Apache (>50% all websites), so very common server-side language
- Minimal type system, lots of strange behavior, just awful
- I’ve never used it and I never will (hopefully)

PHP example

```
$a = md5('240610708');
$b = md5('QNKCDZO');

echo "$a\n";
echo "$b\n";
echo "\n";

var_dump($a == $b);
```

LOLCODE

- Age: 7 years
- An example of an esoteric programming language

```
HAI
 CAN HAS STDIO?
 PLZ OPEN FILE "LOLCATS.TXT"?
     AWSUM THX
         VISIBLE FILE
     O NOES
         INVISIBLE "ERROR!"
 KTHXBYE
```

```
HAI
CAN HAS STDIO?
IM IN YR LOOP UPPIN YR VAR TIL BOTH SAEM VAR AN 10
    VISIBLE SUM OF VAR AN 1
IM OUTTA YR LOOP
KTHXBYE
```