Announcement

- Lab 9 (graded)
  - To be returned this afternoon
- Assignment 6
  - Due November 11
  - Sorting: Combining C and IA-32 assembly

Lecture 25

Locality

CPSC 275
Introduction to Computer Systems

IA-32 Linux Memory Layout

- Stack
  - Runtime stack (8MB limit)
  - e. g., local variables
- Heap
  - Dynamically allocated storage
  - When call `malloc`, `calloc`, `new`
- Data
  - Statically allocated data
  - e.g., globals & strings declared in code
- Text
  - Executable machine instructions
  - Read-only (*re-entrant*)

Physical location?

Upper 2 hex digits
= 8 bits of address

Random-Access Memory (RAM)

- Key features
  - RAM is traditionally packaged as a chip.
  - Basic storage unit is normally a cell (one bit per cell).
  - Multiple RAM chips form a memory.
- RAM comes in two varieties:
  - DRAM (dynamic RAM)
  - SRAM (static RAM)

SRAM vs DRAM

	Trans. per bit	Access time	Needs refresh?	Needs EDC?	Cost	Applications
SRAM	4 or 6	1X	No	Maybe	100x	Cache memories
DRAM	1	10X	Yes	Yes	1X	Main memories, frame buffers

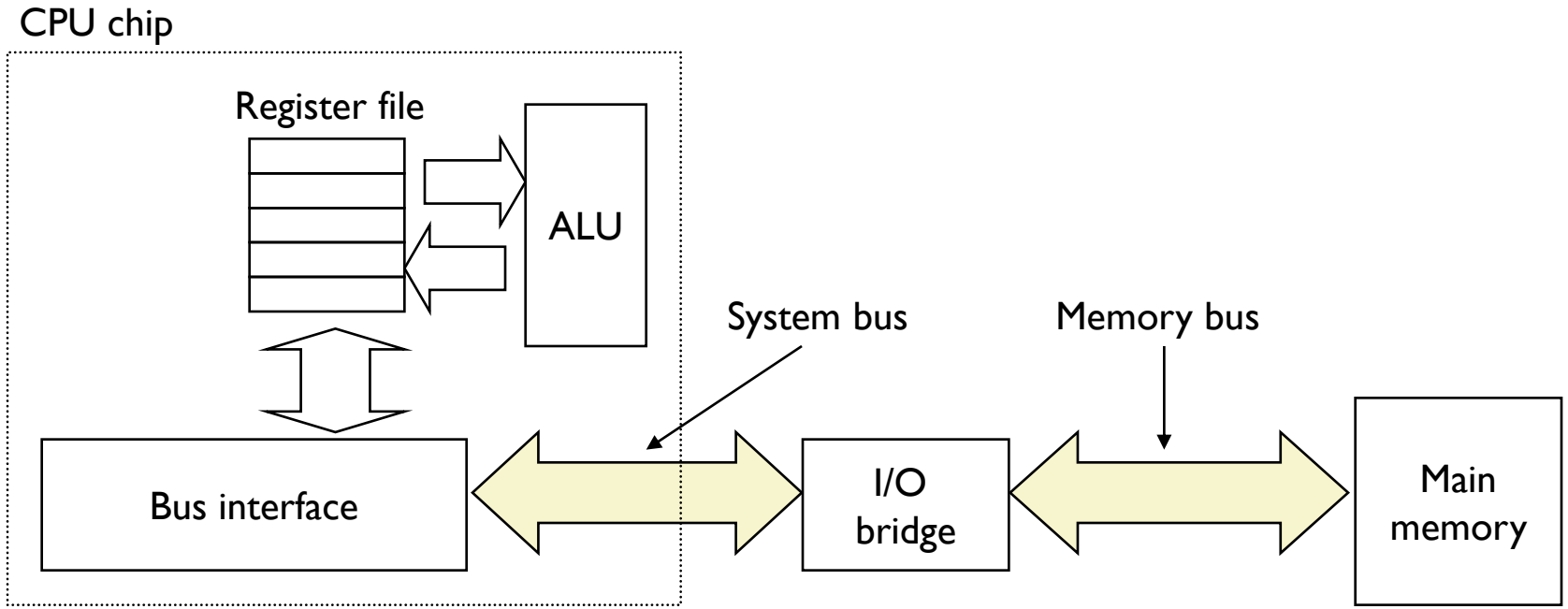
DRAM and SRAM are *volatile* memories
(lose information if powered off)

Nonvolatile Memories

- *Nonvolatile* memories retain value even if powered off
  - Read-only memory (ROM): programmed during production
  - Programmable ROM (PROM): can be programmed once
  - Eraseable PROM (EPROM): can be bulk erased (UV, X-Ray)
  - Electrically eraseable PROM (EEPROM): electronic erase capability
  - Flash memory: EEPROMs. with partial (block-level) erase capability
- Uses for nonvolatile memories
  - *Firmware* programs stored in a ROM (BIOS, controllers for disks, network cards, graphics accelerators, security subsystems,…)
  - Solid state disks (SSD)
  - Disk caches

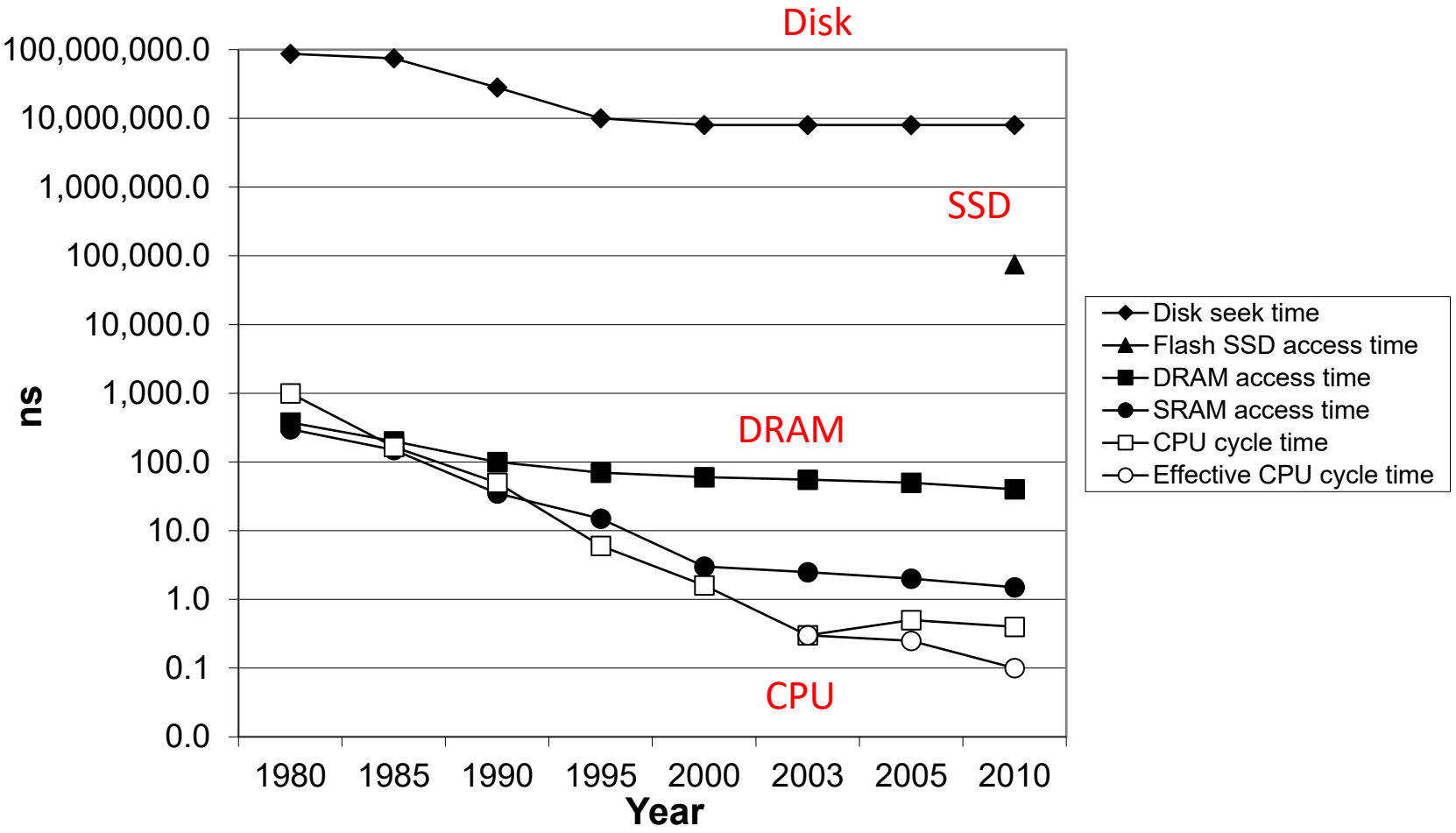
Connecting CPU and Memory

- A bus is a collection of parallel wires that carry address, data, and control signals.
- Buses are typically shared by multiple devices.



The CPU-Memory Gap

The gap widens between DRAM, disk, and CPU speeds.



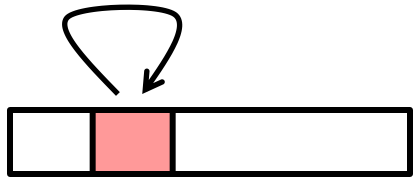
Locality to the rescue!

The key to bridging this CPU-Memory gap is a fundamental property of computer programs known as locality.

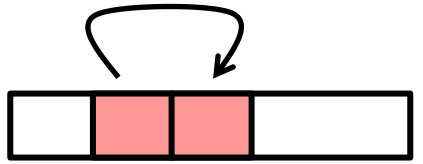
What is Locality?

- Principle of Locality: Programs tend to use data and instructions with addresses near or equal to those they have used recently

- Temporal locality:
  - Recently referenced items are likely to be referenced again in the near future



- Spatial locality:
  - Items with nearby addresses tend to be referenced close together in time



Locality Example

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

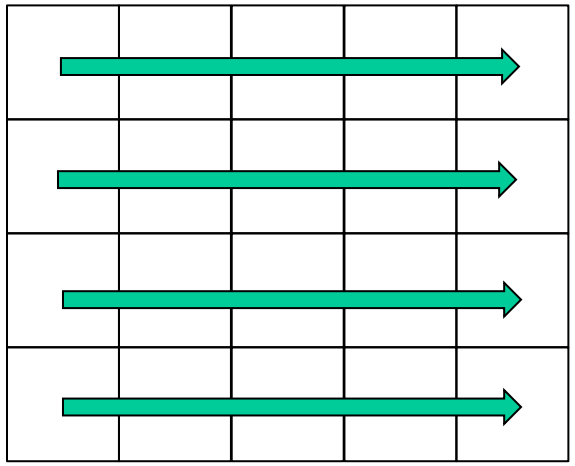
- data references
  - Reference array elements in succession (stride-1 reference pattern). — spatial locality
  - Reference variable `sum` each iteration. — temporal locality
- instruction references
  - Reference instructions in sequence. — spatial locality
  - Cycle through loop repeatedly. — temporal locality

Qualitative Estimates of Locality

- Being able to look at code and get a qualitative sense of its locality is a key skill for a professional programmer.
- *Q*: Does this function exhibit good locality with respect to array `a`?

```
int sum_array_rows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
       for (j = 0; j < N; j++)
          sum += a[i][j];
    return sum;
}
```



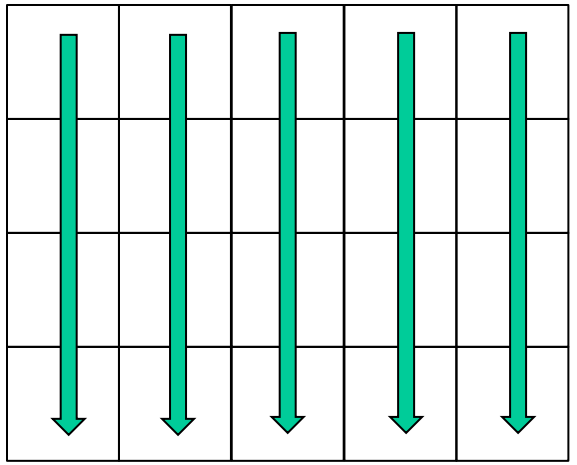
Q: Stride? 1

Locality Example

- *Q*: Does this function exhibit good locality with respect to array `a`?

```
int sum_array_cols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
       for (i = 0; i < M; i++)
          sum += a[i][j];
    return sum;
}
```



Q: Stride? `N`

- *Q*: Which one is better?

Locality Example

- Q: Can you permute the loops so that the function scans the 3-d array `a` with a stride-1 reference pattern and thus has good spatial locality? (Homework)

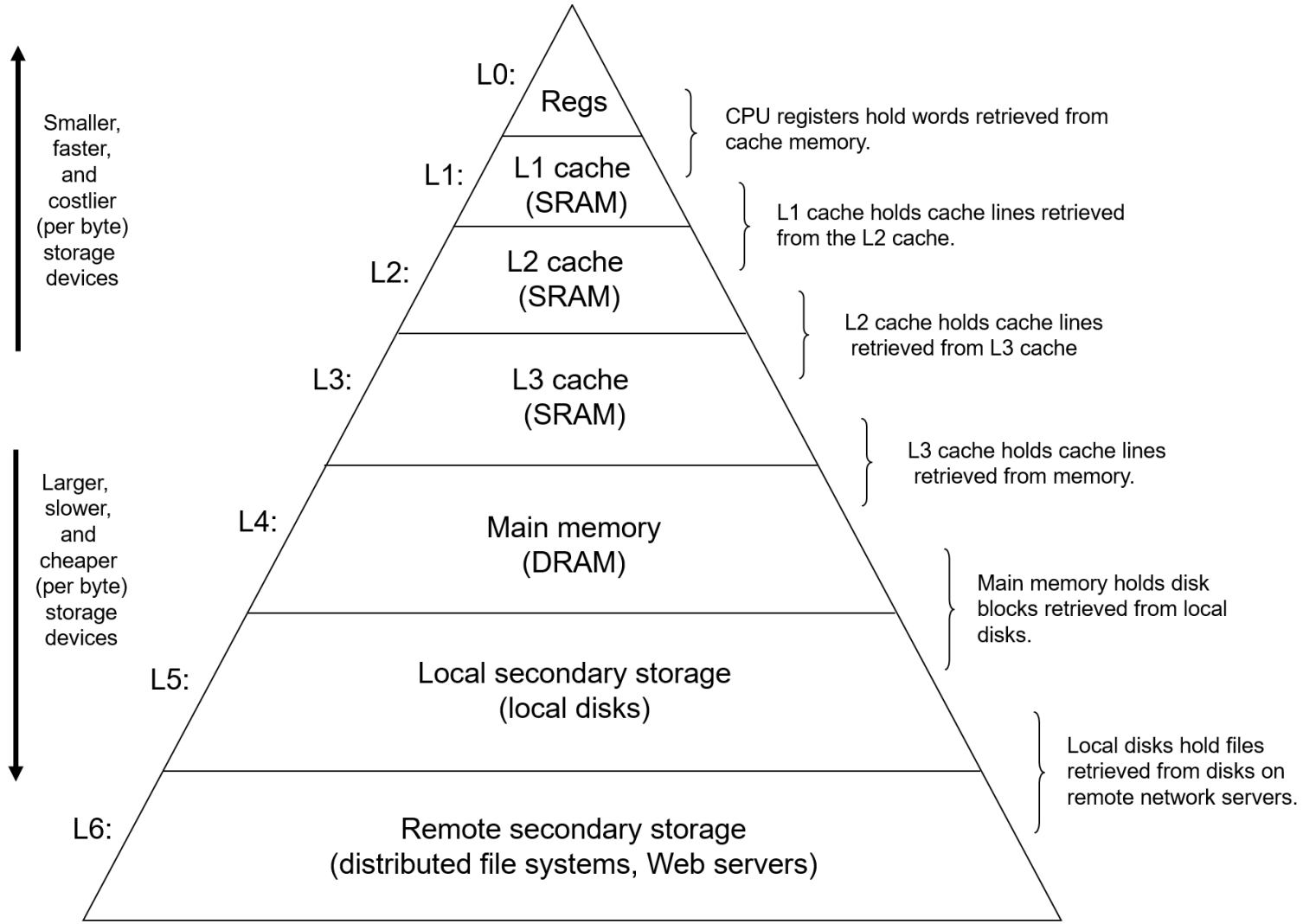
```
int sum_array_3d(int a[N][N][N])
{
    int i, j, k, sum = 0;

    for (i = 0; i < N; i++)
       for (j = 0; j < N; j++)
          for (k = 0; k < N; k++)
             sum += a[k][i][j];
    return sum;
}
```

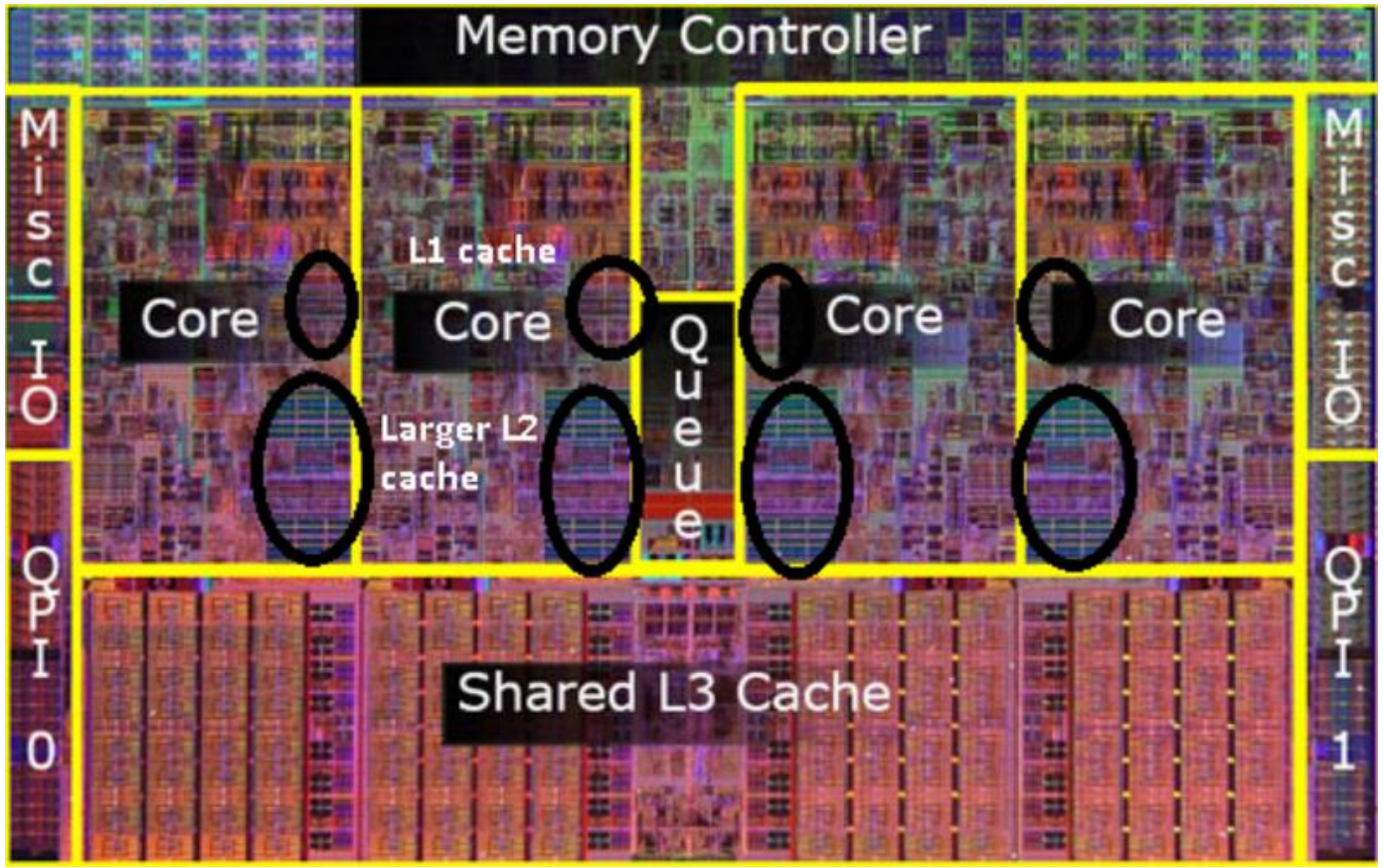
Memory Hierarchy

- Some fundamental and enduring properties of hardware and software:
  - Fast storage technologies cost more per byte, have less capacity, and require more power (heat!).
  - The gap between CPU and main memory speed is widening.
  - **Well-written** programs tend to exhibit **good locality**.
- This led to an approach of organizing memory and storage systems known as a memory hierarchy.

Memory Hierarchy



Where is cache?



A typical quad-core processor (Intel i7)

Cache

- *Cache*: A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device.
- Fundamental idea of a memory hierarchy:
  - For each *k*, the faster, smaller device at level *k* serves as a cache for the larger, slower device at level *k*+1.
- Why do memory hierarchies work?
  - Because of locality, programs tend to access the data at level *k* more often than they access the data at level *k*+1.
  - Thus, the storage at level *k*+1 can be slower, and thus larger and cheaper per bit.