MEMORY SYSTEM

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

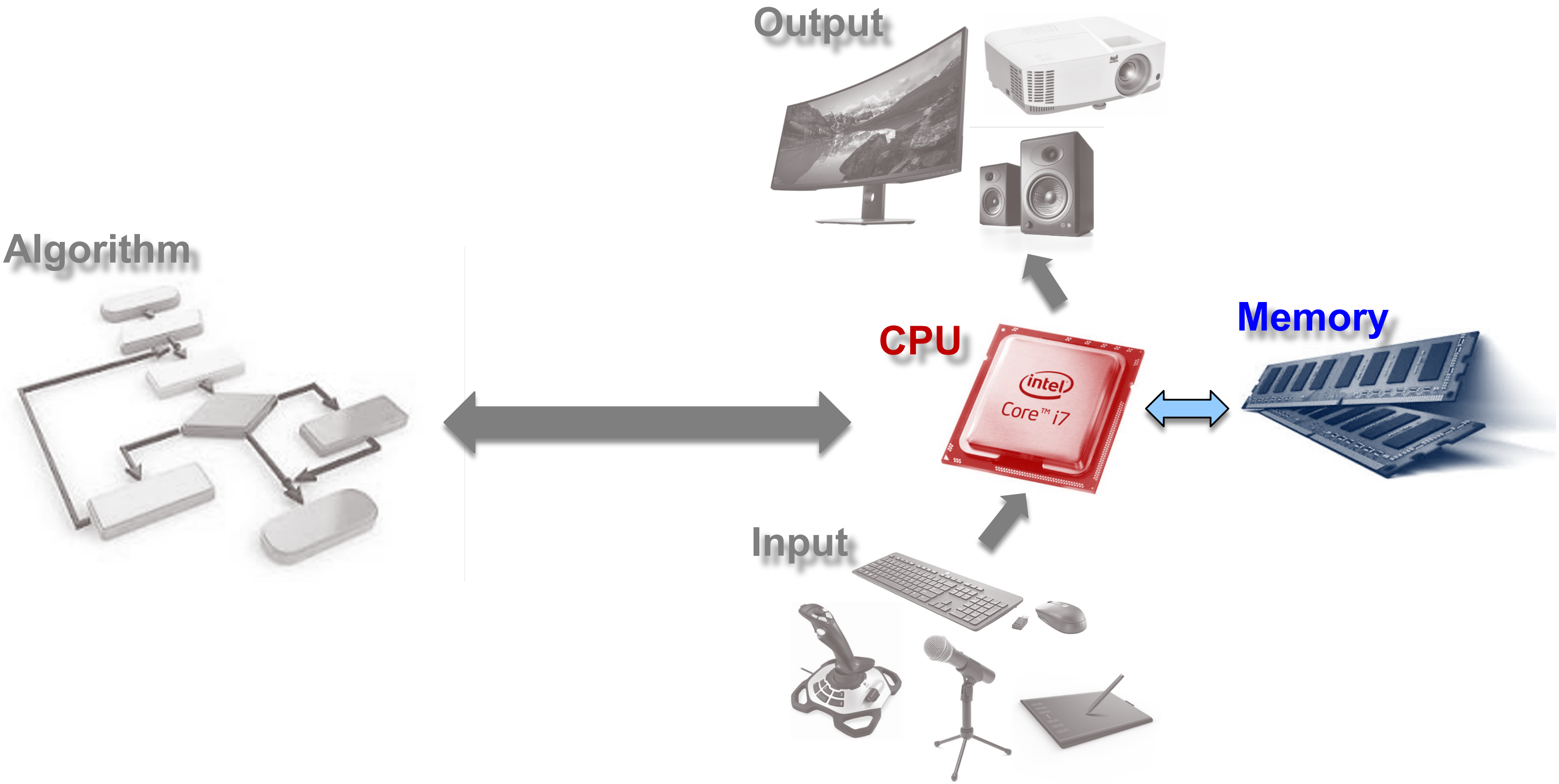
University of Utah

Overview

- This lecture
  - Memory system
    - Cache

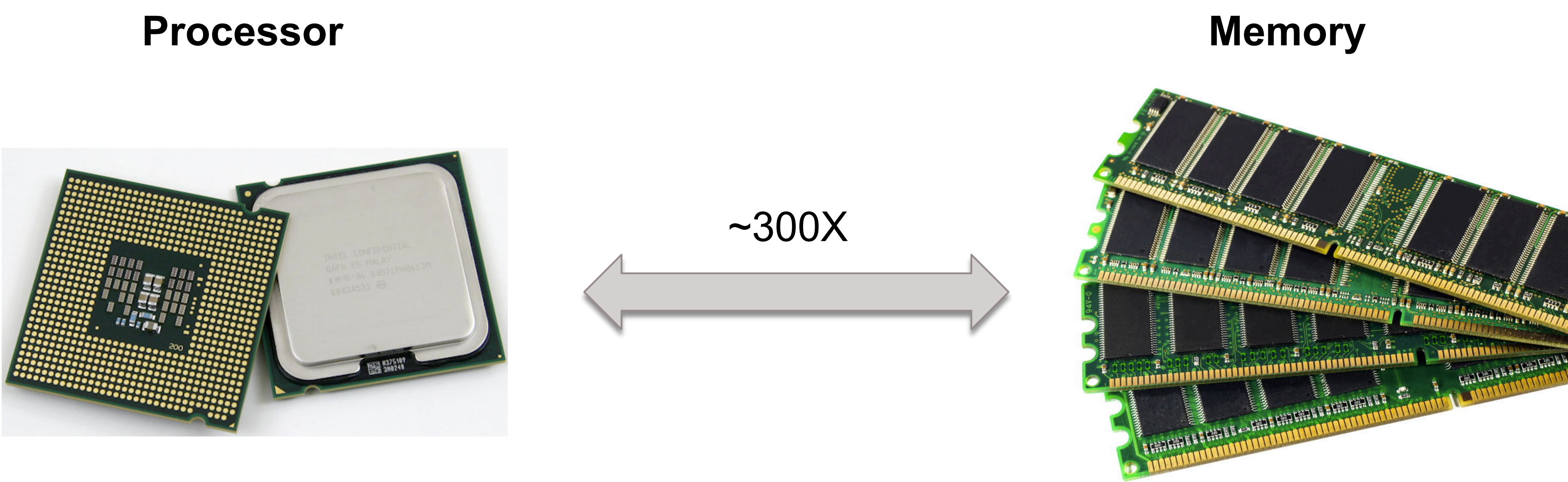
Computer Organization

- Classic components of a computing system



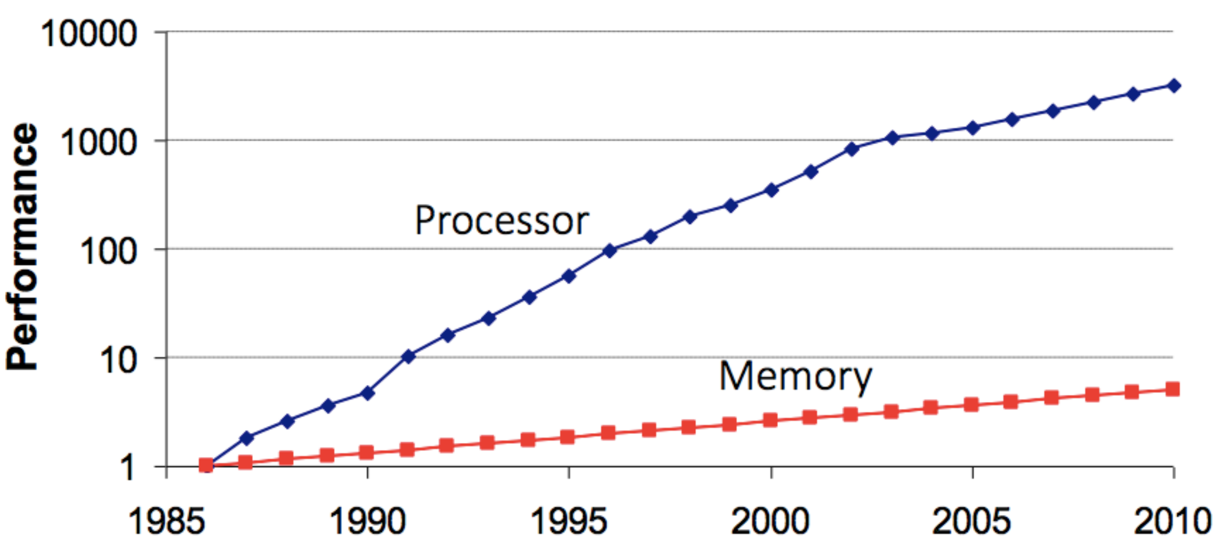
Memory System

- Data and instructions are stored on DRAM chips
  - DRAM has high bit density and low speed
  - An access DRAM may take about 300 processor cycles



The Memory Wall

- Processor-memory performance gap increased over 50% per year
  - Processor performance historically improved ~60% per year
  - Main memory access time improves ~5% per year



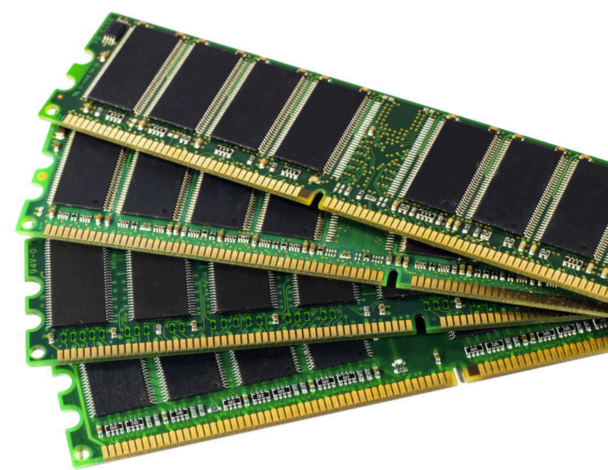
Memory System

- Data and instructions are stored on DRAM chips
  - DRAM has high bit density and low speed
  - An access DRAM may take about 300 processor cycles
- How to bridge the speed gap?

Processor

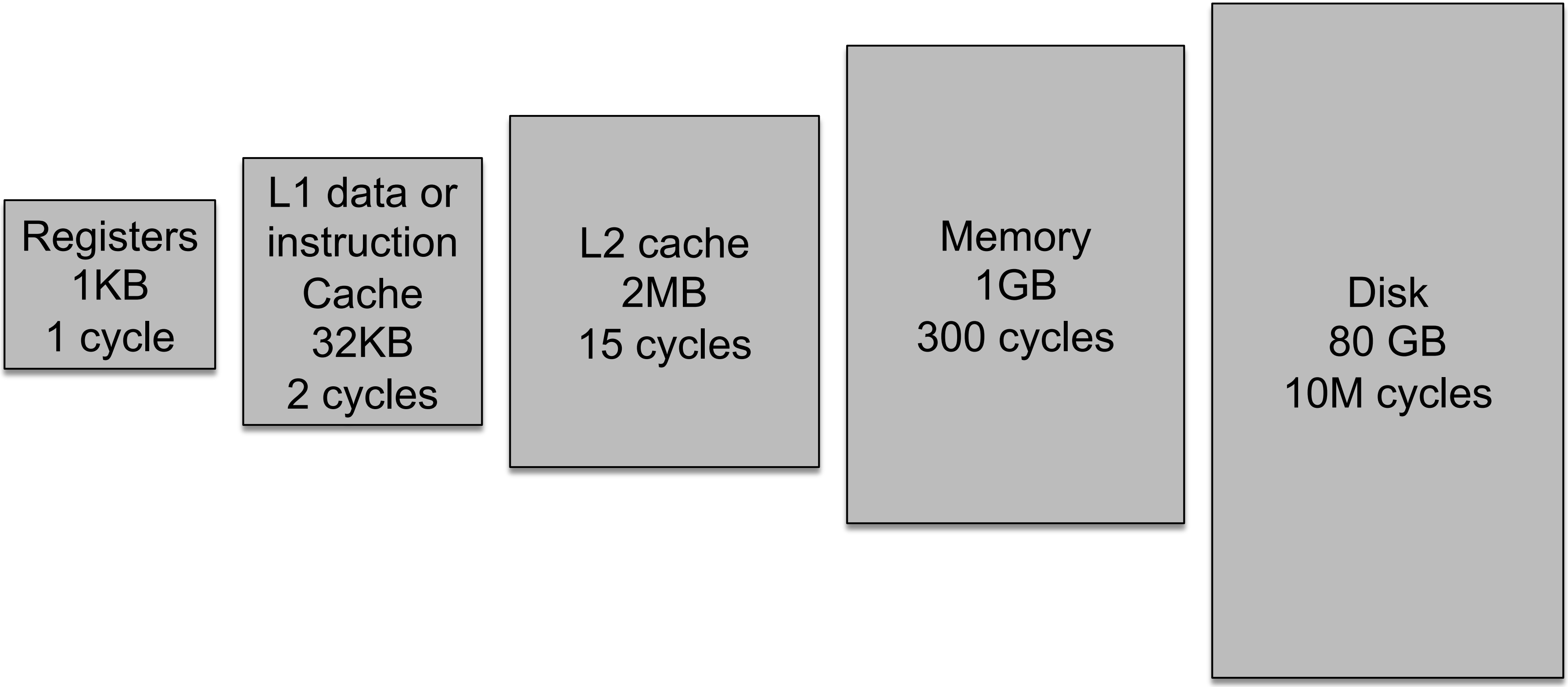
~300X

Memory



Memory Hierarchy

- The basic structure of a memory hierarchy.



Processor Cache

- Occupies a large fraction of die area in modern microprocessors

3-3.5 GHz
~$1000 (2014)

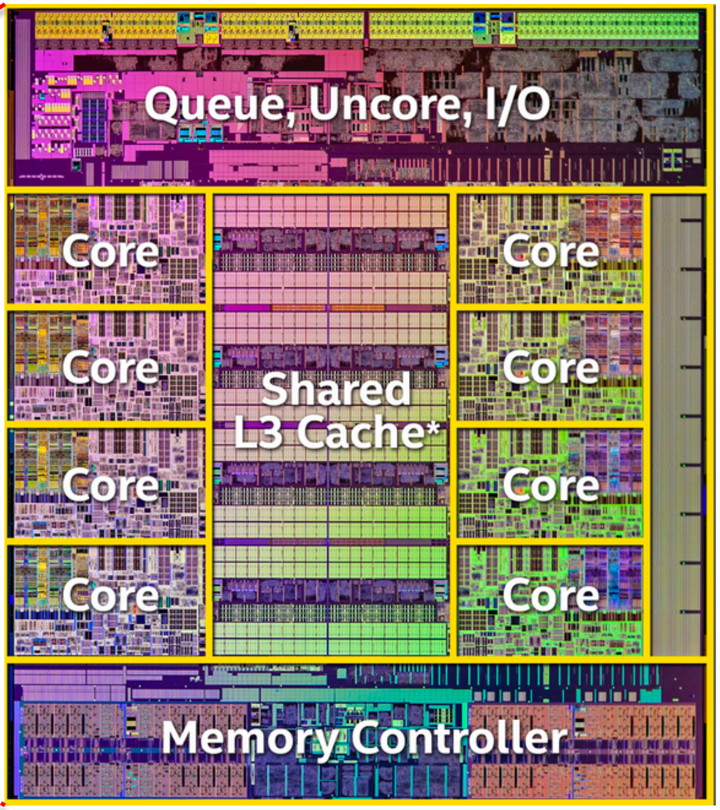
Source: Intel Core i7

Processor Cache

- Occupies a large fraction of die area in modern microprocessors

3-3.5 GHz
~$1000 (2014)

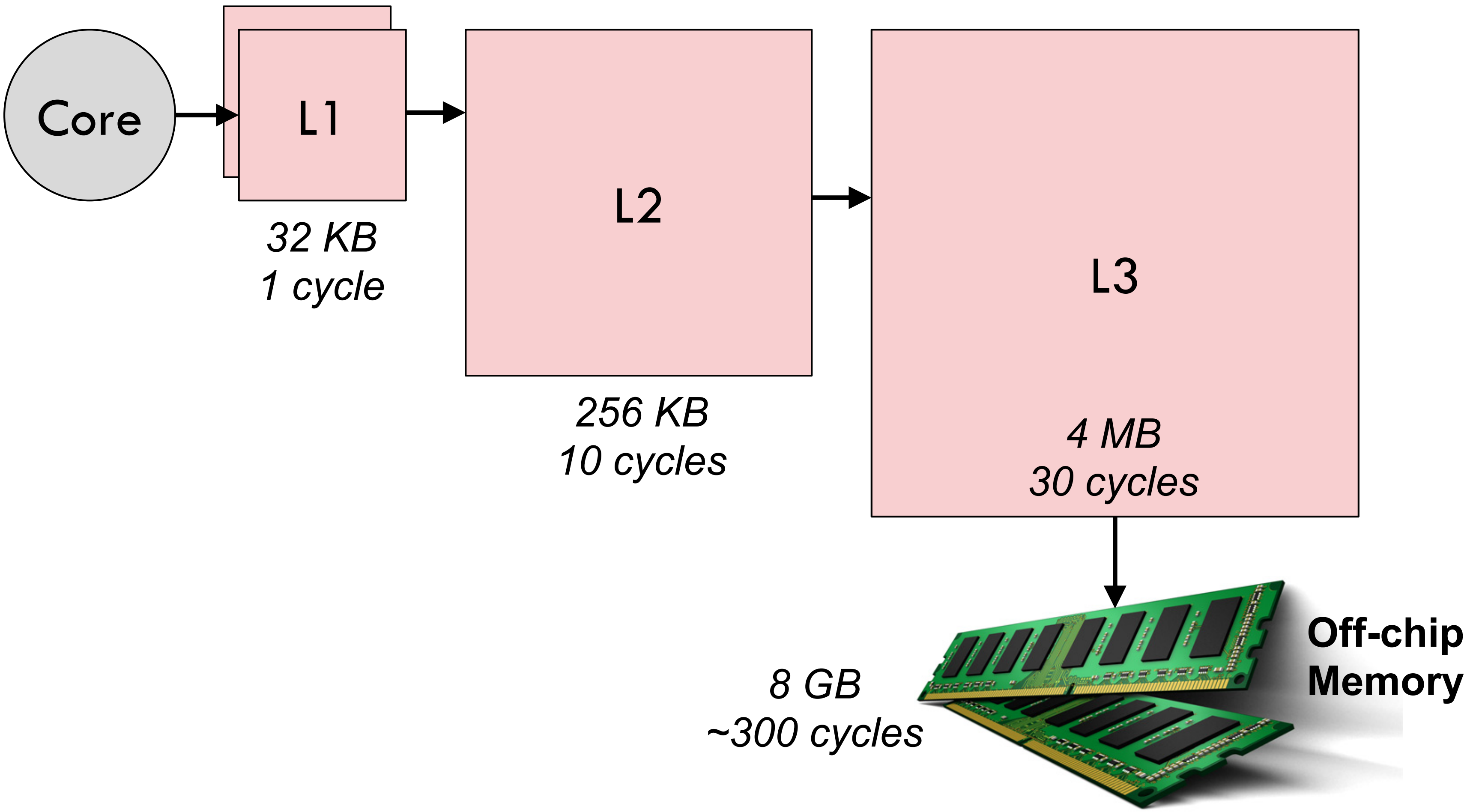
20MB of cache



Source: Intel Core i7

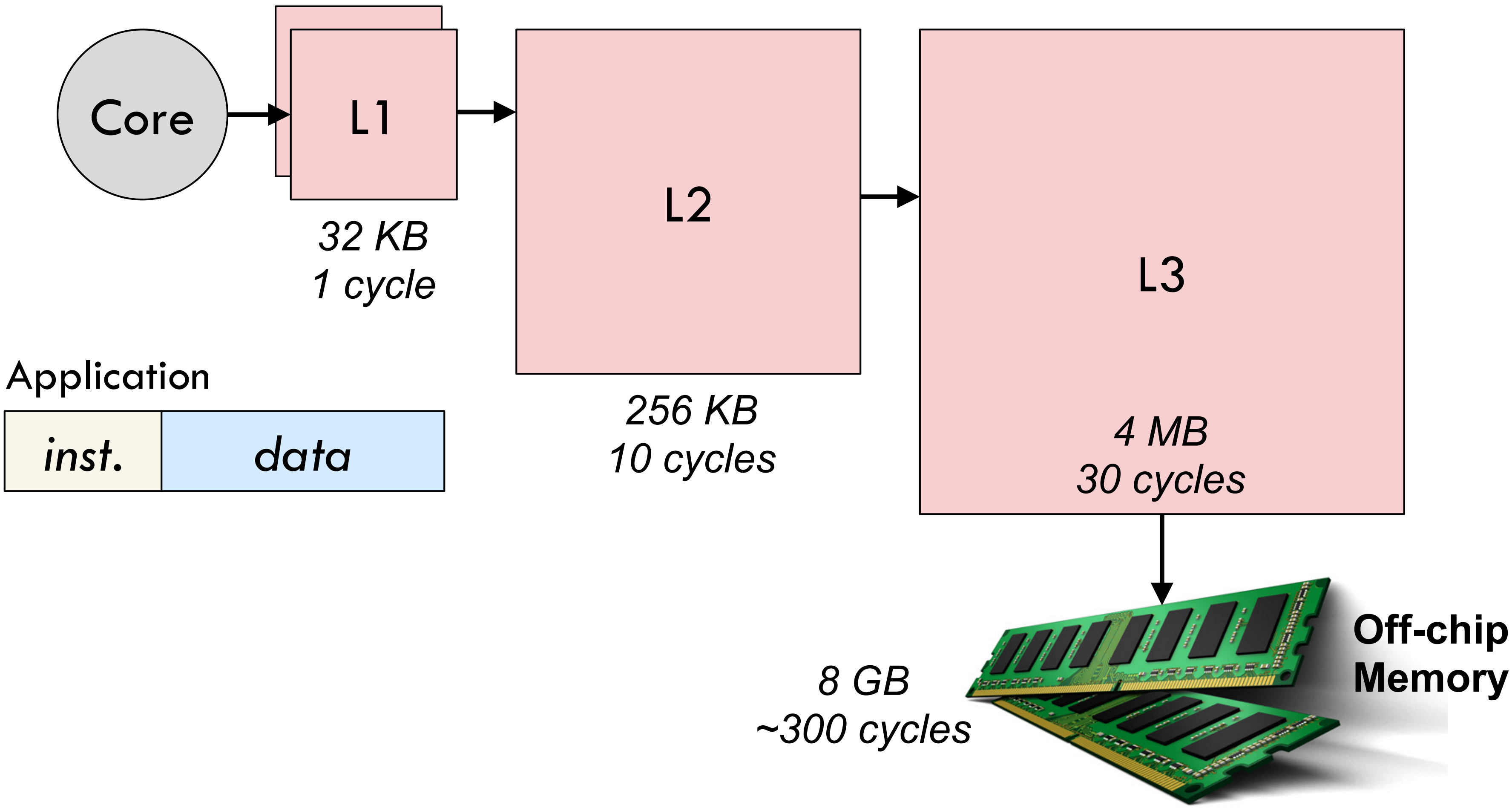
Cache Hierarchy

- Example three-level cache organization



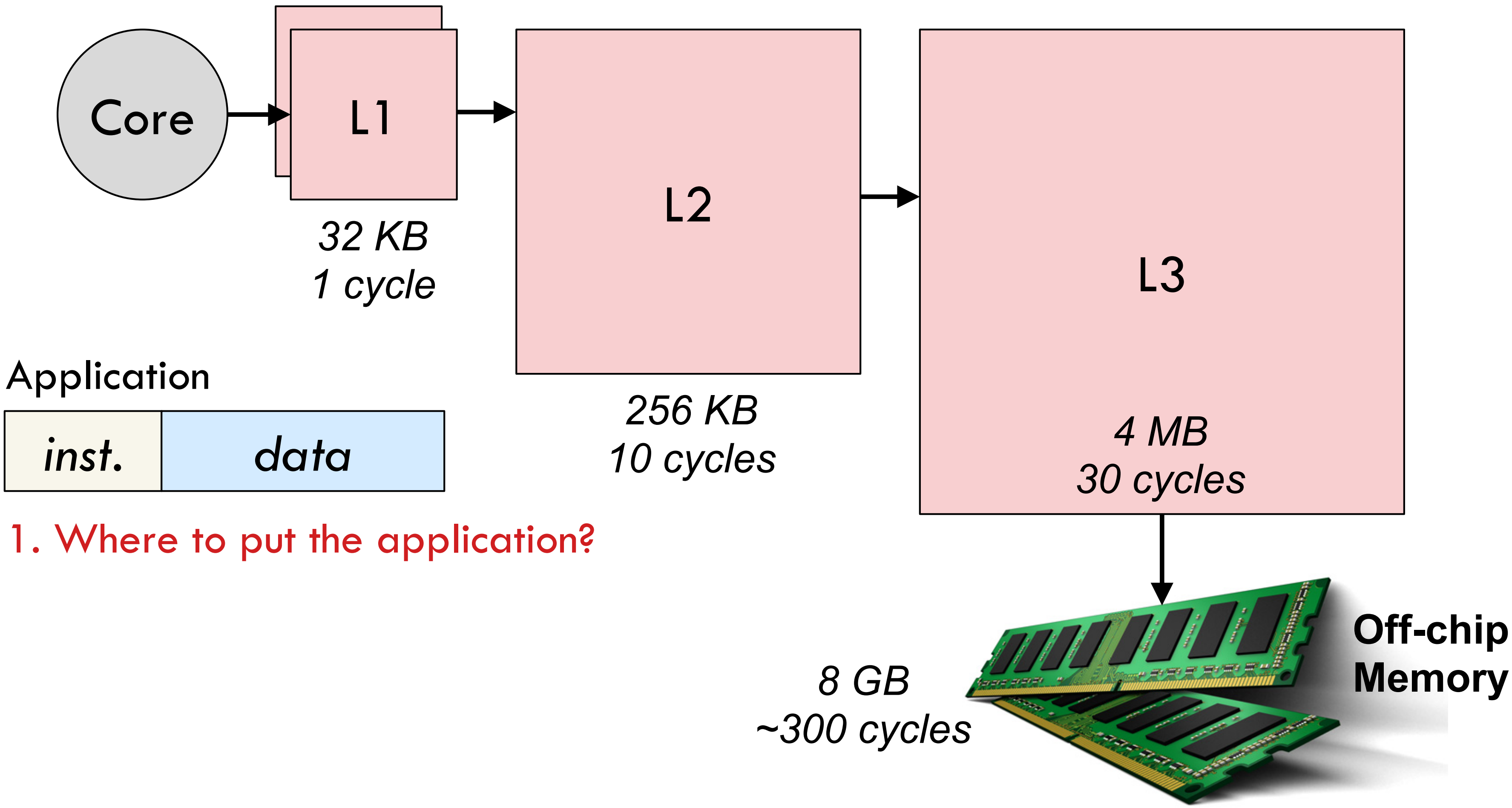
Cache Hierarchy

- Example three-level cache organization



Cache Hierarchy

- Example three-level cache organization

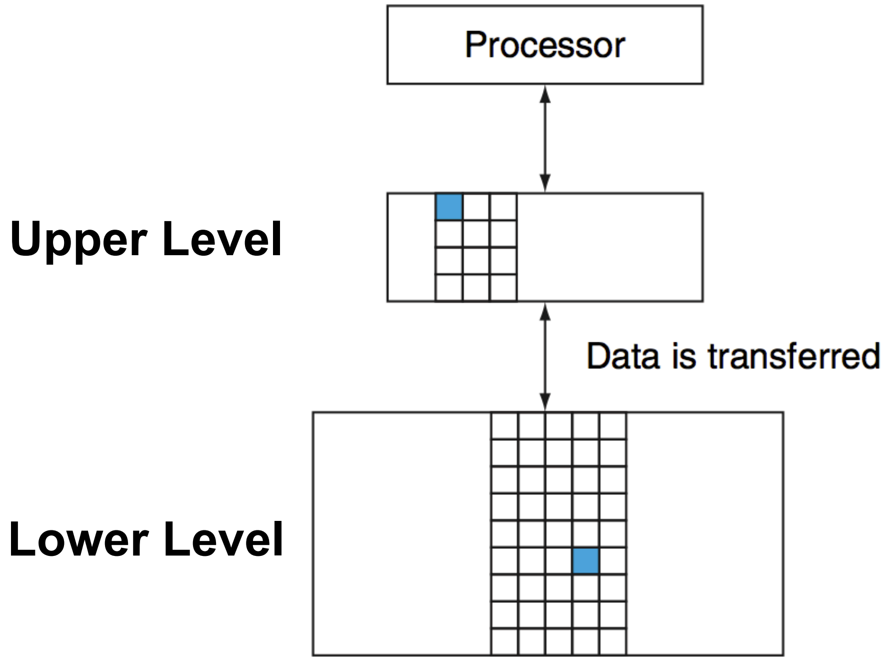


1. Where to put the application?

Memory Hierarchy

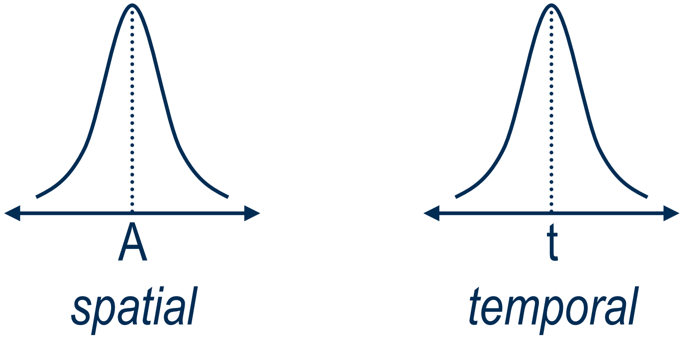
- The basic structure of a memory hierarchy.
- Multiple levels of the memory

Idea: keep important data closer to processor.



Principle of Locality

- Memory references exhibit localized accesses
- Types of locality
  - *spatial*: probability of access to $A+\delta$ at time $t+\varepsilon$ highest when $\delta \rightarrow 0$
  - *temporal*: probability of accessing $A+\varepsilon$ at time $t+\delta$ highest when $\delta \rightarrow 0$

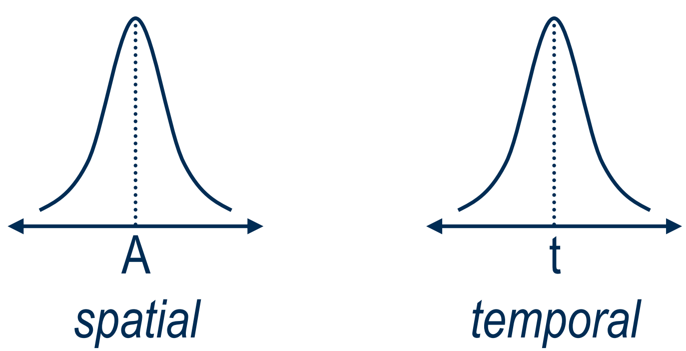



```
for (i=0; i<1000; ++i) {
        sum = sum + a[i];
}
```

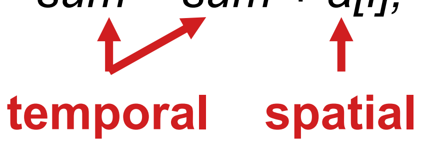
Key idea: store local data in fast cache levels

Principle of Locality

- Memory references exhibit localized accesses
- Types of locality
  - *spatial*: probability of access to $A+\delta$ at time $t+\varepsilon$ highest when $\delta \to 0$
  - *temporal*: probability of accessing $A+\varepsilon$ at time $t+\delta$ highest when $\delta \to 0$



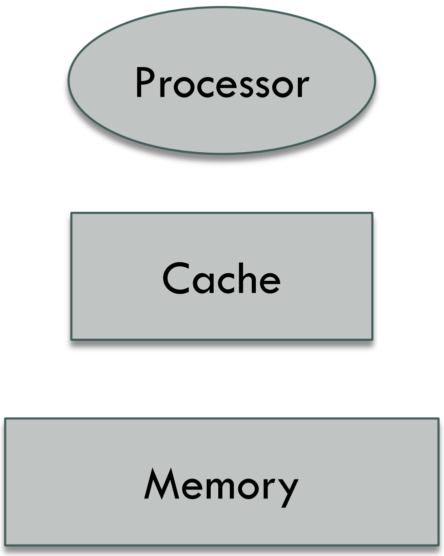

```
for (i=0; i<1000; ++i) {
        sum = sum + a[i];
}
```



Key idea: store local data in fast cache levels

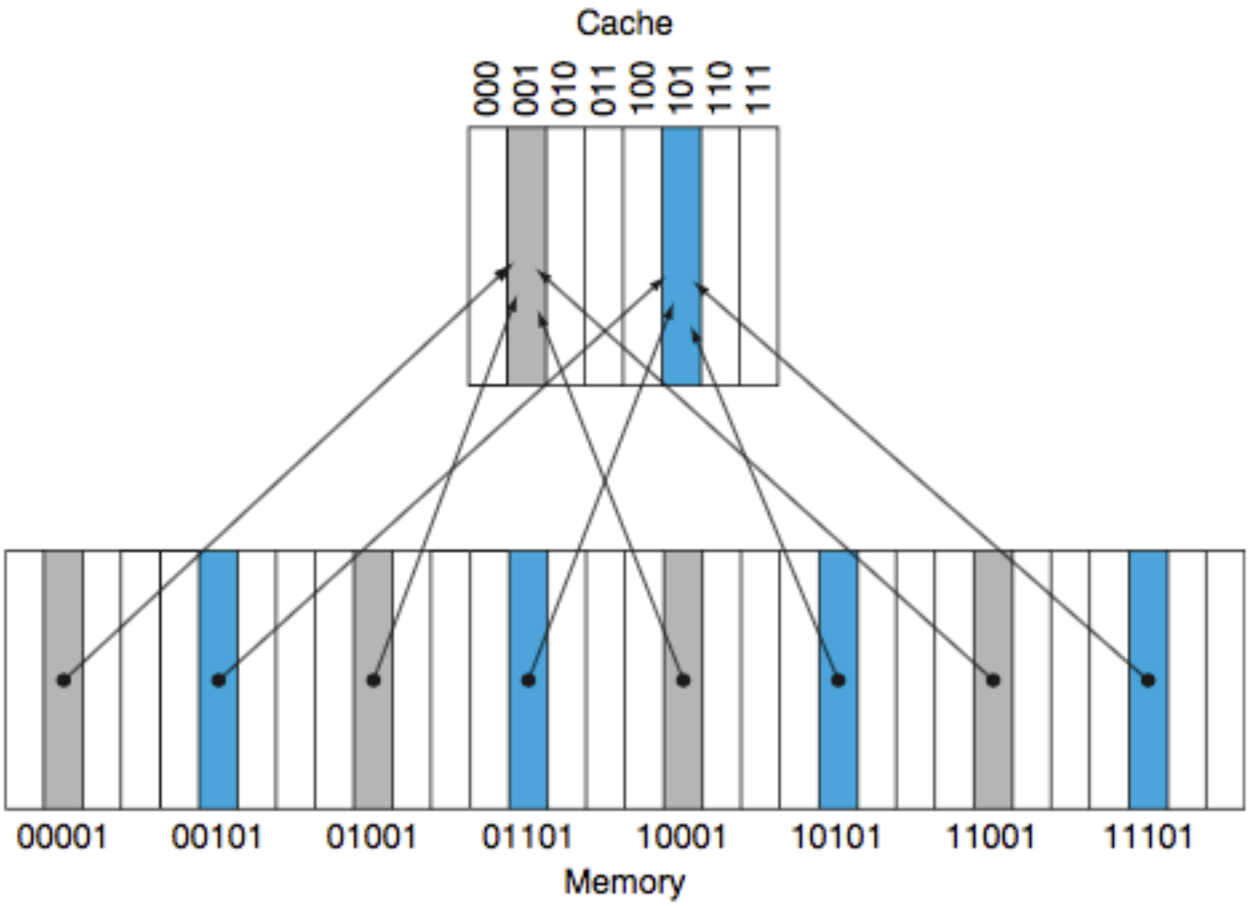
Cache Architecture

- Design principles
  - Temporal locality: if you used some data recently, you will likely use it again
  - Spatial locality: if you used some data recently, you will likely access its neighbors
- Cache terminology
  - Access time
  - Hit vs. miss
  - Miss penalty



Direct-Mapped Cache

- Cache address



Direct-Mapped Cache

- Cache lookup