CSE 520
Computer Architecture II

Memory Organization

Prof. Michel A. Kinsy

The course has 3 modules

Module 1

- Instruction Set Architecture (ISA)
- Simple Pipelining and Hazards
- Branch Prediction
- Superscalar Architectures
- Other Advanced Architectures

Module 2

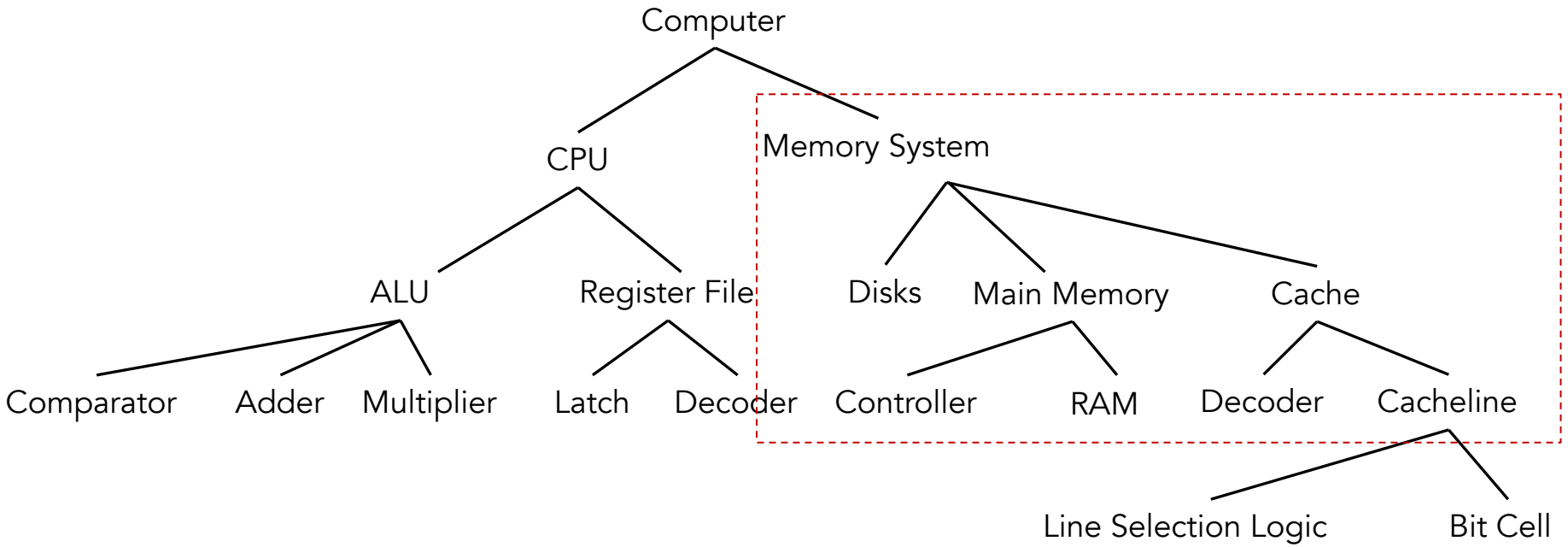
- Caches
- Memory Models & Synchronization
- Cache Coherence Protocols

Module 3

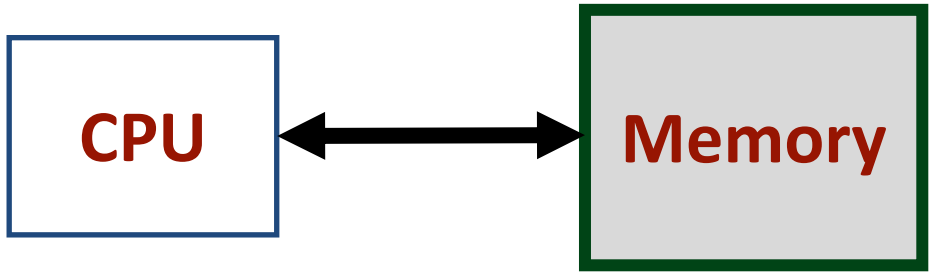
- On-Chip networks
- On-chip Network routing

Computing: Computer Architecture

- The DNA of Modern Computing



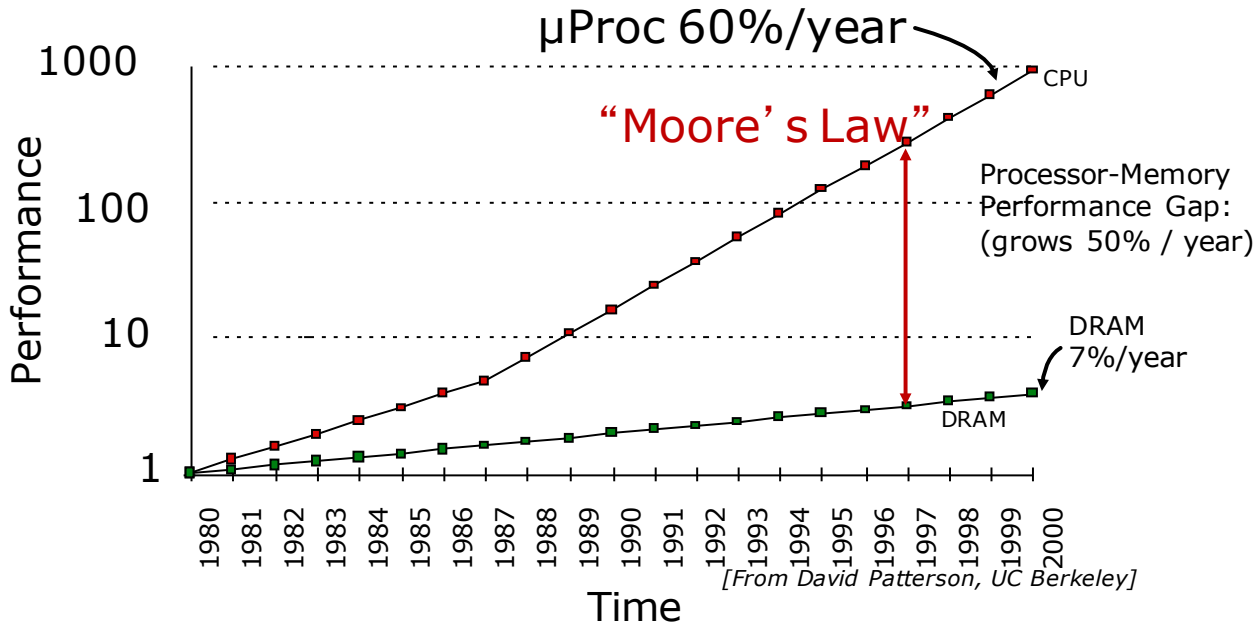
CPU-Memory Bottleneck



- Performance of high-speed computers is usually limited by memory bandwidth & latency
  - Latency (time for a single access) Memory access time >> Processor cycle time
  - Bandwidth (number of accesses per unit time) if fraction m of instructions access memory,
    - 1+m memory references / instruction
      - Ghost of the stored-program architecture
    - CPI = 1 requires 1+m memory refs / cycle

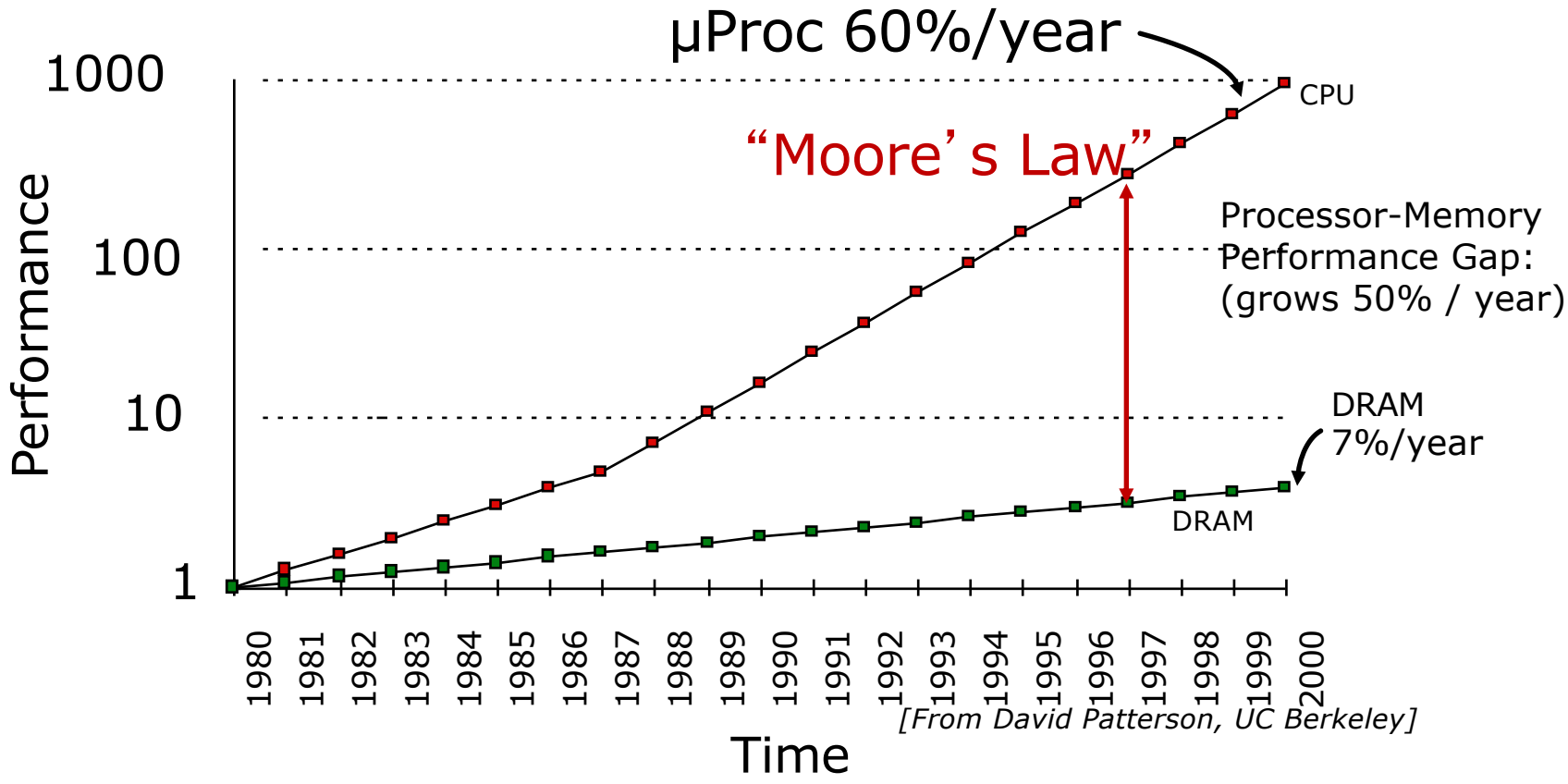
Processor- Memory Gap

- Performance gap: CPU (55% each year) vs. DRAM (7% each year)
  - Processor operations take of the order of 1 ns
  - Memory access requires 10s or even 100s of ns
  - Each instruction executed involves at least one memory access



Processor-DRAM Gap (latency)

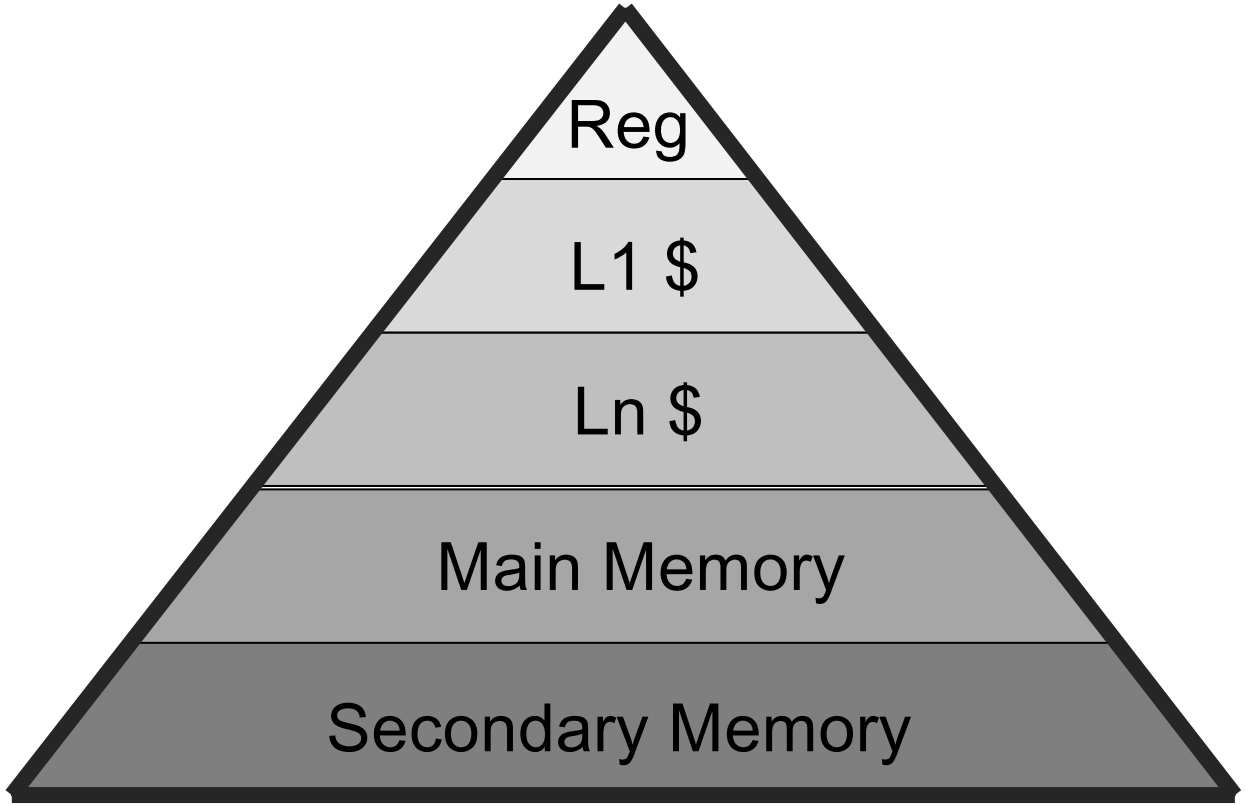
- Four-issue 2GHz superscalar accessing 100ns DRAM could execute 800 instructions during time for one memory access!



[From David Patterson, UC Berkeley]

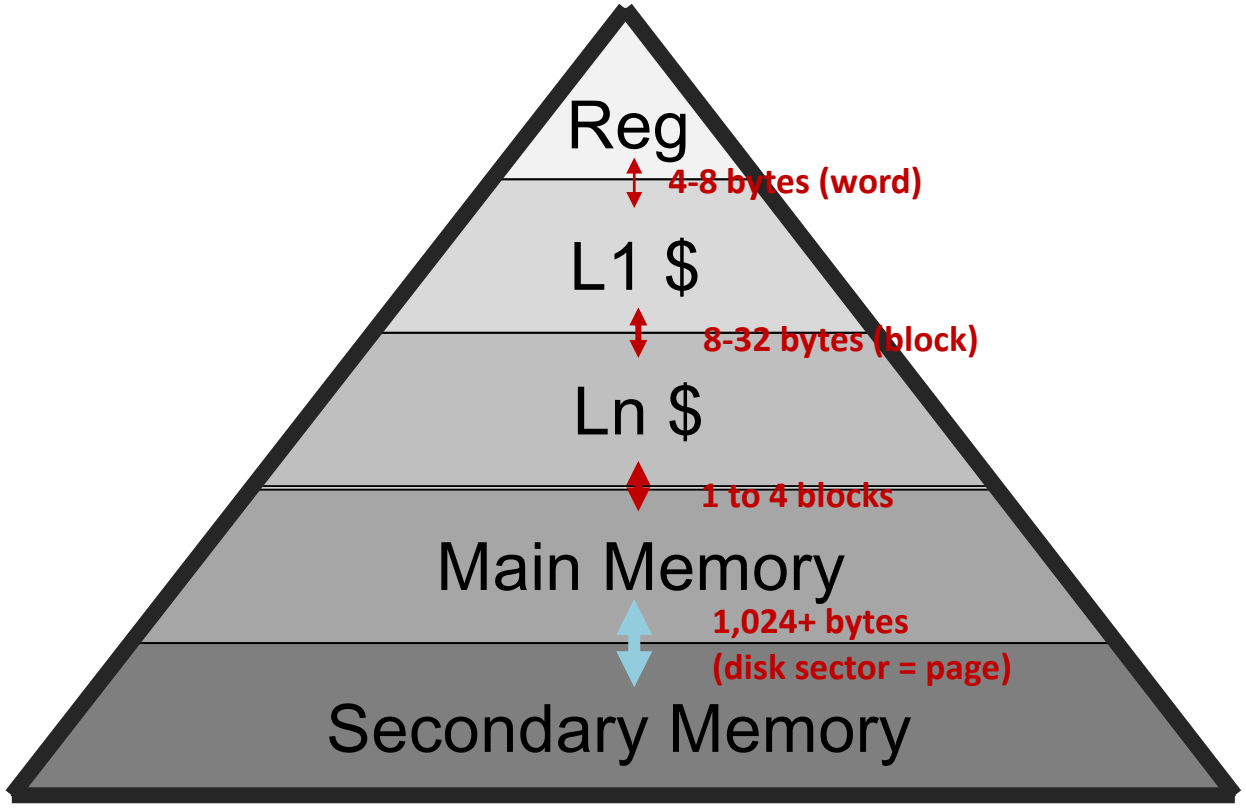
Memory Organization

- Memory is organized and accessed in ways to hide this gap



Memory Organization

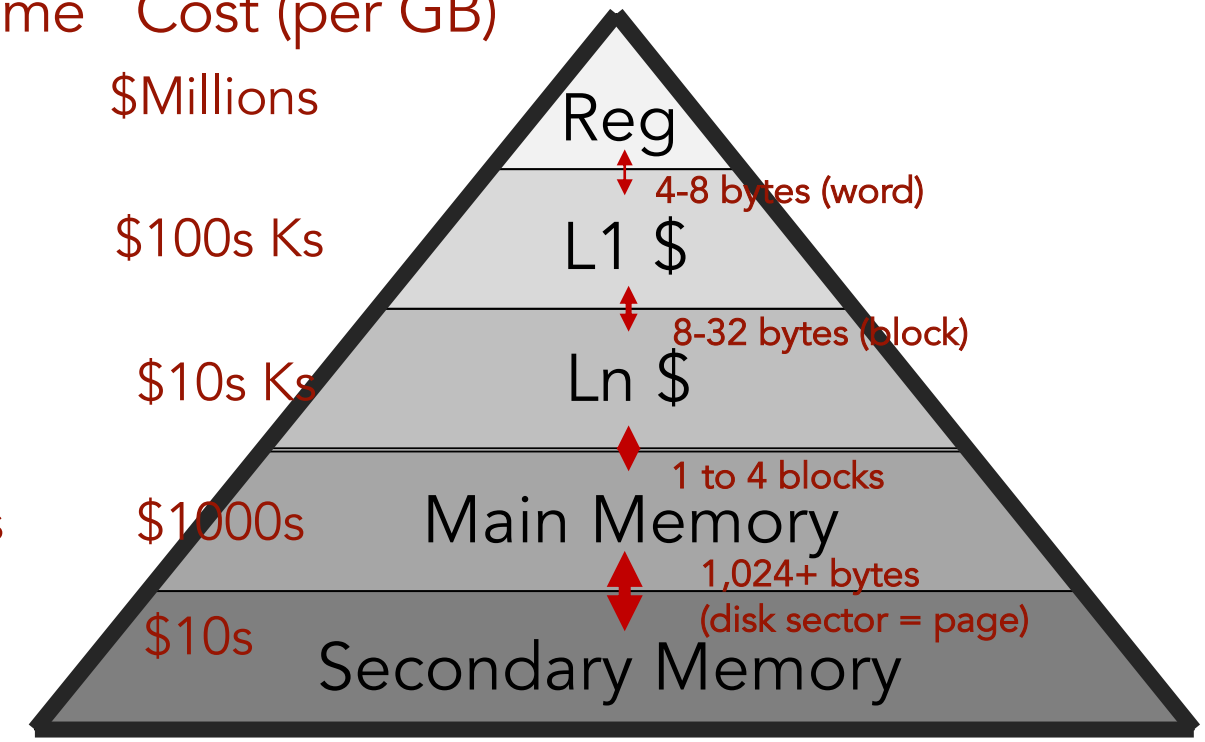
- Memory is organized and accessed in ways to hide this gap



Memory Trends

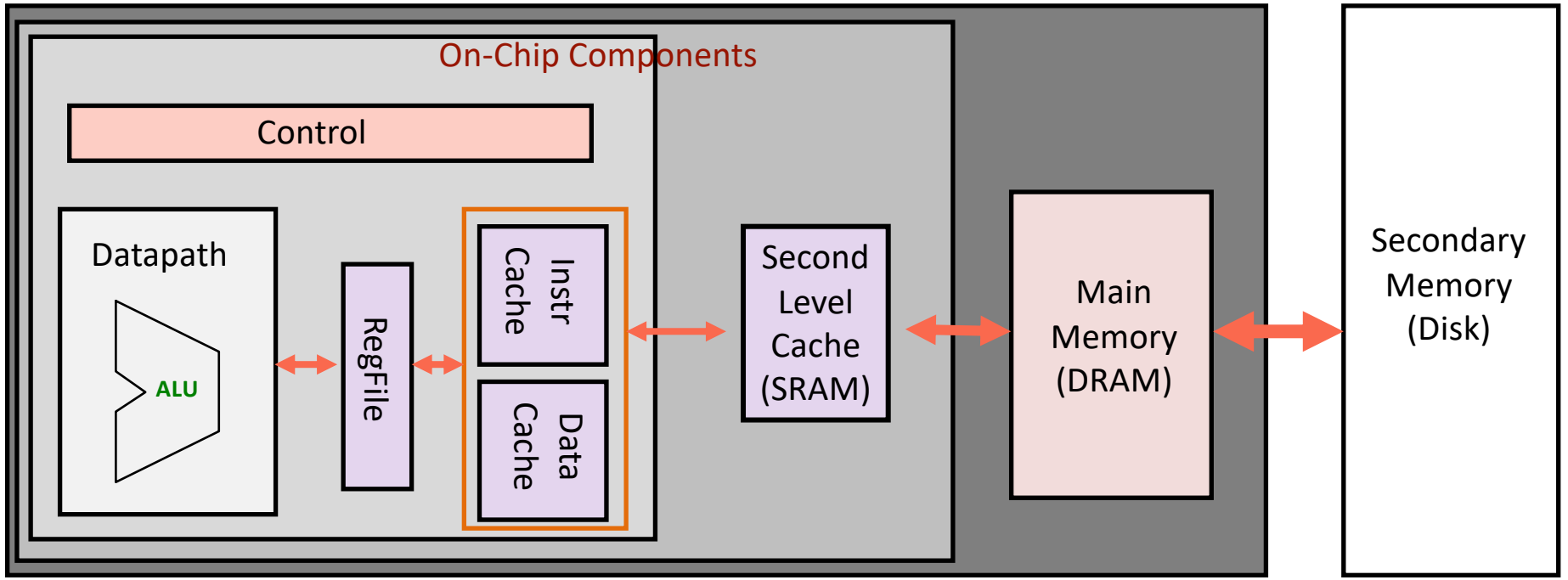
- The fastest memories are expensive and thus not very large

Capacity	Access Time	Cost (per GB)
100s B	ns	$Millions
10s KB	few ns	$100s Ks
MBs	10s ns	$10s Ks
100s MB	100s ns	$1000s
10s GB	10s ms	$10s

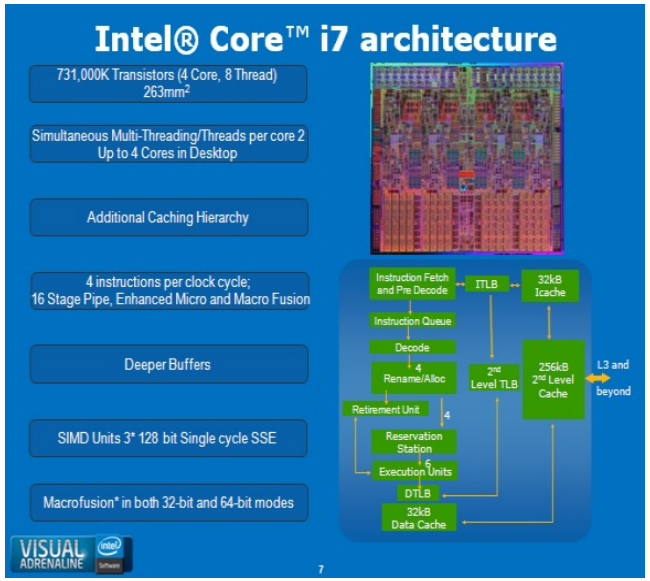


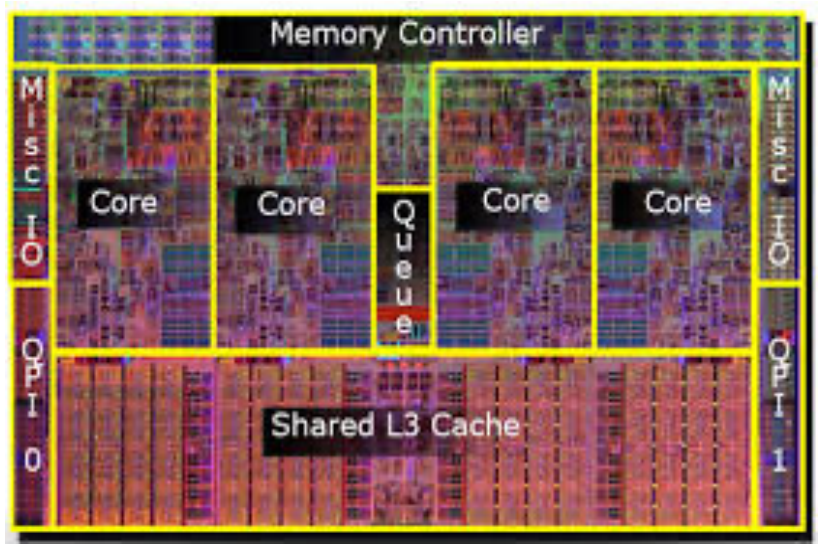
Illustrative View of Memory Organization

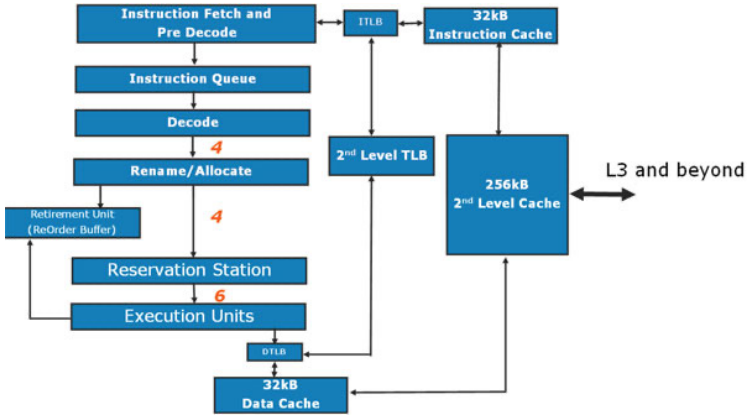
- A fast memory can help bridge the CPU-memory gap
- The fastest memories are expensive and thus not very large



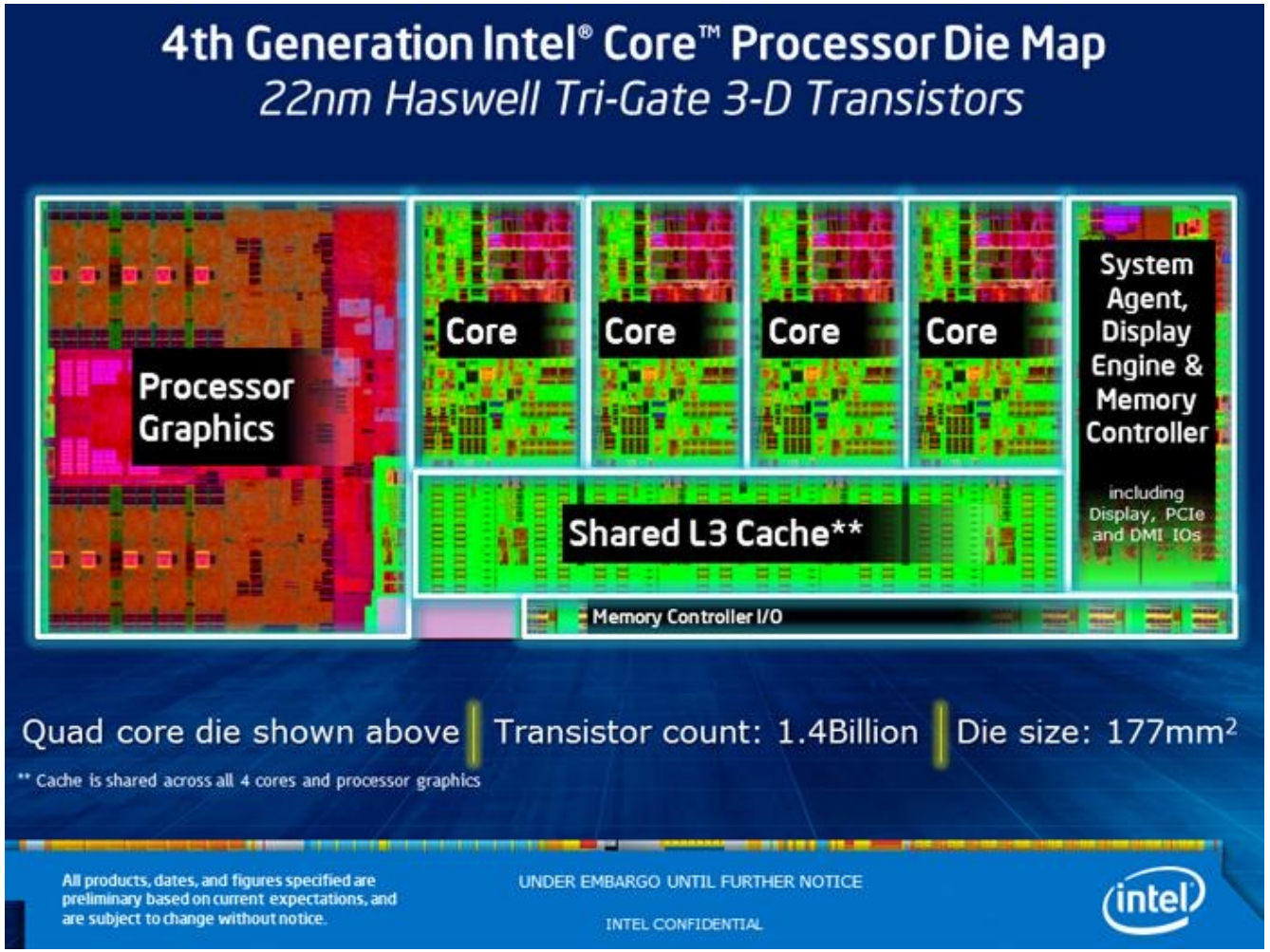
Intel Core i7 Organization







Intel Haswell



Memory Technology

- Early machines used a variety of memory technologies
  - Manchester Mark I used CRT Memory Storage
  - EDVAC used a mercury delay line
- Core memory was first large scale reliable main memory
  - Invented by Forrester in late 40s at MIT for Whirlwind project
  - Bits stored as magnetization polarity on small ferrite cores threaded onto 2 dimensional grid of wires

Memory Technology

- First commercial DRAM was Intel 1103
  - 1Kbit of storage on single chip
  - Charge on a capacitor used to hold value
- Semiconductor memory quickly replaced core in 1970s
  - Intel formed to exploit market for semiconductor memory
- Phase change memory (PCM) looking promising for the future
  - Slightly slower, but much denser than DRAM and non-volatile

Memory Technology

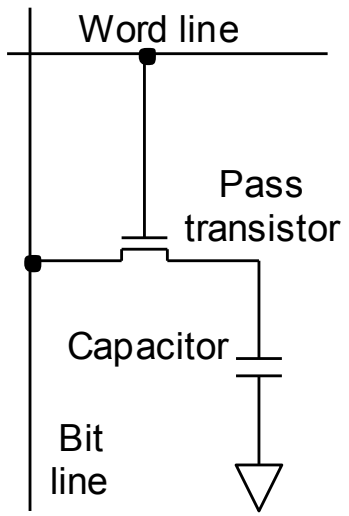
- Random Access Memory (RAM)
  - Any byte of memory can be accessed without touching the preceding bytes
  - RAM is the most common type of memory found in computers and other digital devices
  - There are two main types of RAM
    - DRAM (Dynamic Random Access Memory)
      - Needs to be "refreshed" regularly (~ every 8 ms)
      - 1% to 2% of the active cycles of the DRAM
      - Used for Main Memory
    - SRAM (Static Random Access Memory)

Memory Technology

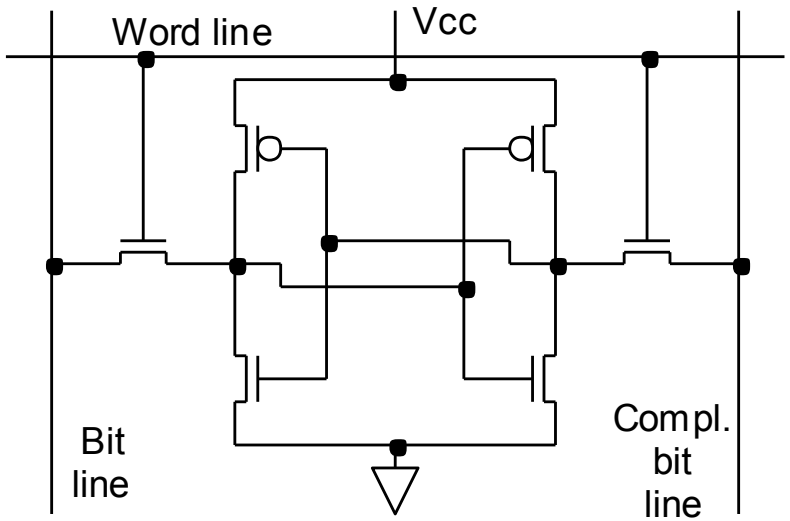
- Random Access Memory (RAM)
  - Any byte of memory can be accessed without touching the preceding bytes
  - RAM is the most common type of memory found in computers and other digital devices
  - There are two main types of RAM
    - DRAM (Dynamic Random Access Memory)
    - SRAM (Static Random Access Memory)
      - Content will last until power turned off
      - Low density (6 transistor cells), high power, expensive, fast
      - Used for caches

Memory Technology

- Single-transistor DRAM cell is considerably simpler than SRAM cell
- This leads to dense, high-capacity DRAM memory chips



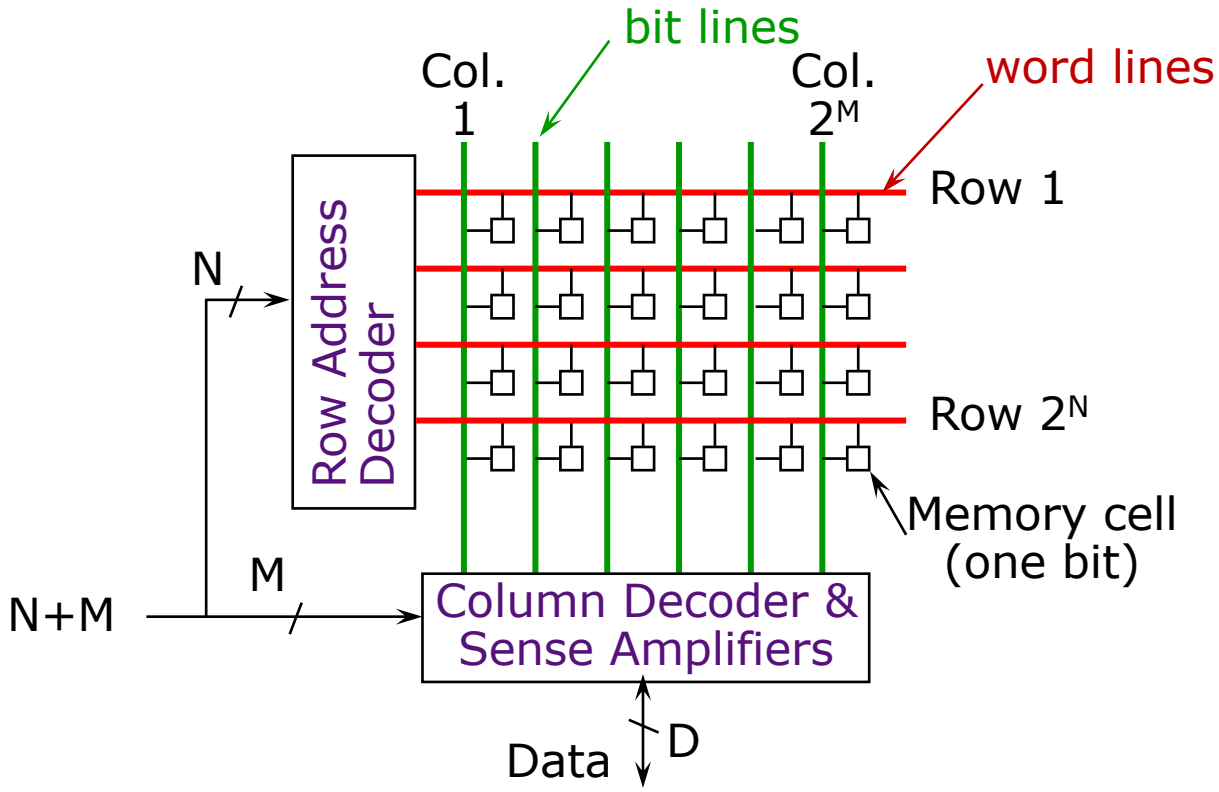
DRAM Cell



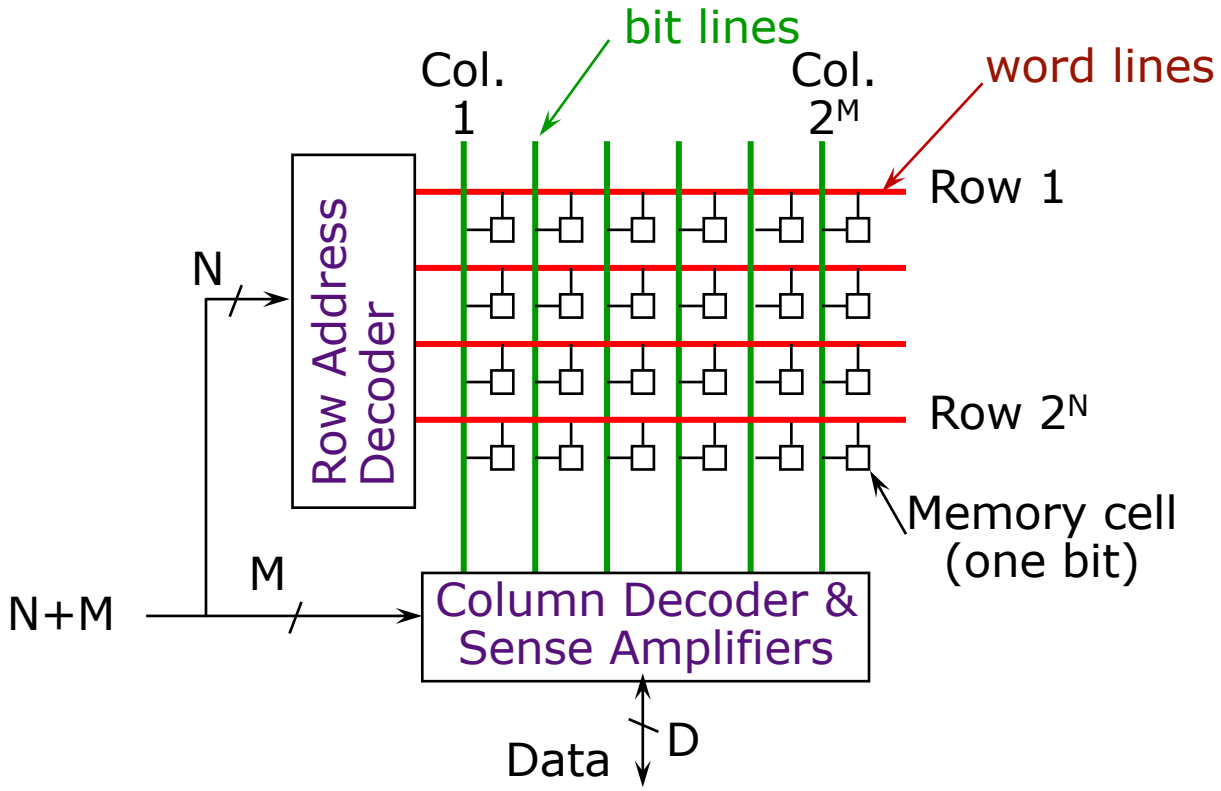
SRAM Cell

RAM Organization

- One memory row holds a block of data, so the column address selects the requested bit or word from that block



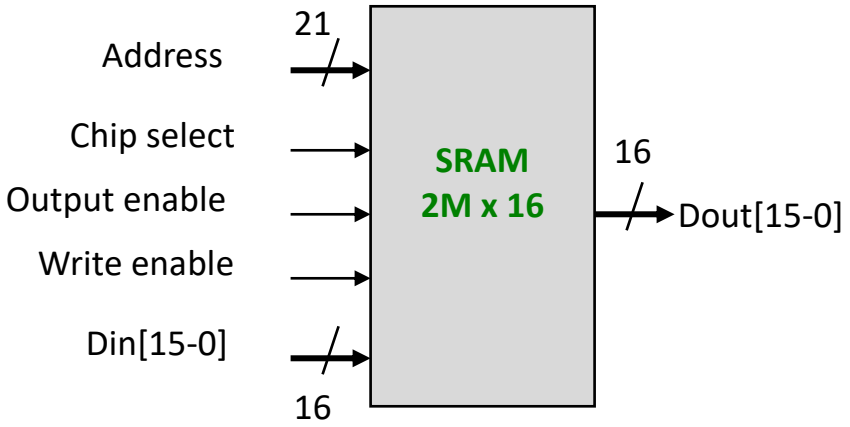
DRAM Architecture



- Modern chips have around 4 logical banks on each chip
  - Each logical bank physically implemented as many smaller arrays

RAM Organization

- One memory row holds a block of data, so the column address selects the requested bit or word from that block
- RAS or Row Access Strobe triggering row decoder
- CAS or Column Access Strobe triggering column selector

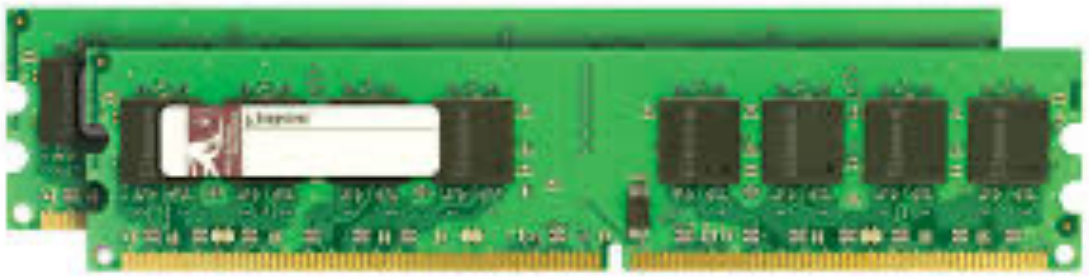


RAM Organization

- Latency: Time to access one word
  - Access time: time between the request and when the data is available (or written)
  - Cycle time: time between requests
  - Usually cycle time > access time
- Bandwidth: How much data from the memory can be supplied to the processor per unit time
  - Width of the data channel * The rate at which it can be used

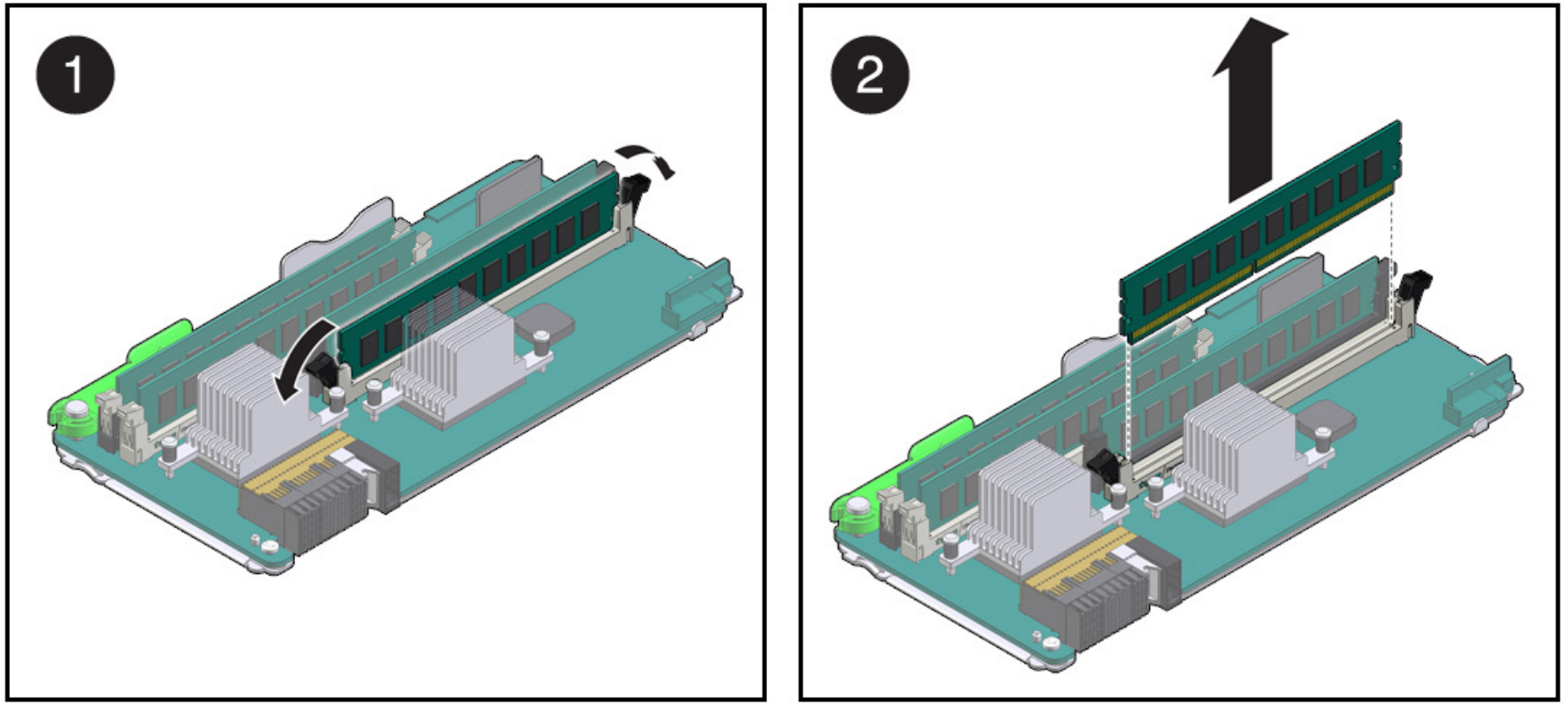
DRAM Packaging

- DIMM (Dual Inline Memory Module) contains multiple chips arranged in "ranks"
  - Each rank has clock/control/address signals connected in parallel (sometimes need buffers to drive signals to all chips), and data pins work together to return wide word
  - A modern DIMM usually has one or two ranks (occasionally 4 if high capacity)



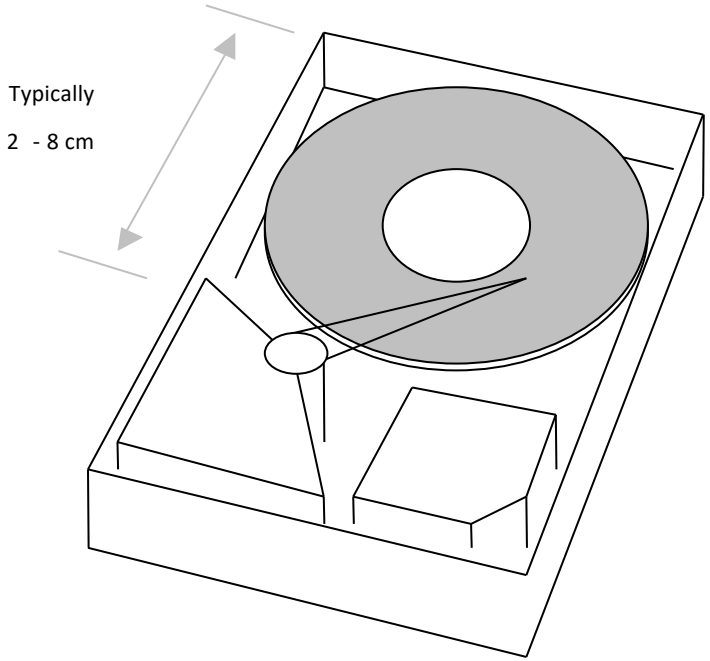
DRAM Packaging

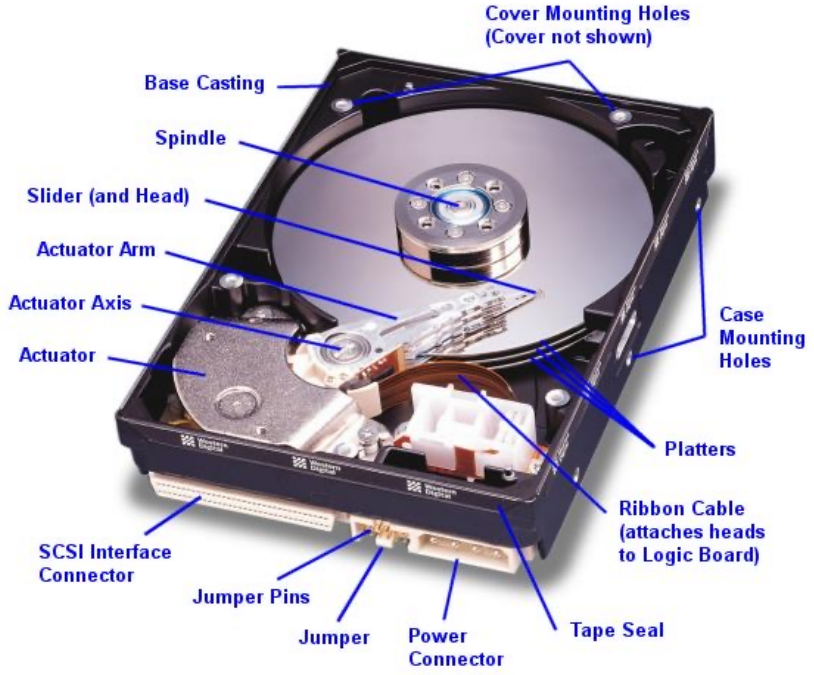
- DIMM (Dual Inline Memory Module) contains multiple chips arranged in "ranks"



Disk Memory Basics

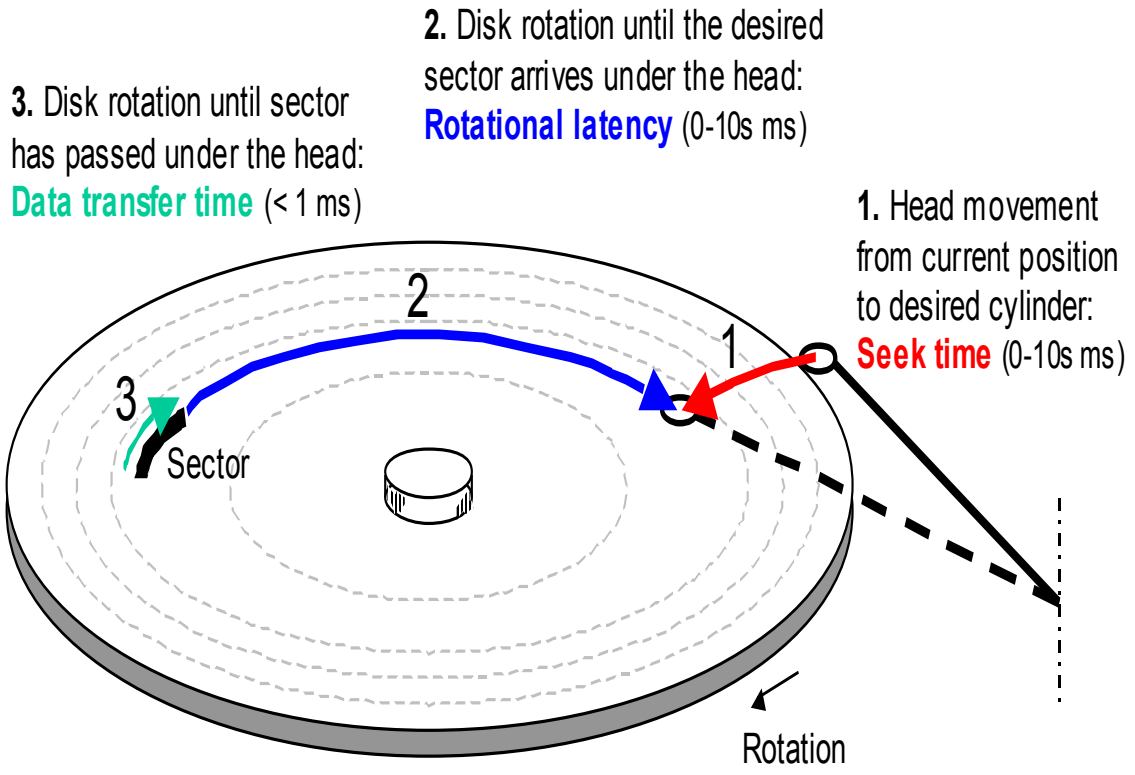
- Hard disk drive (HDD), hard disk, hard drive is the dominant secondary storage device in computer systems





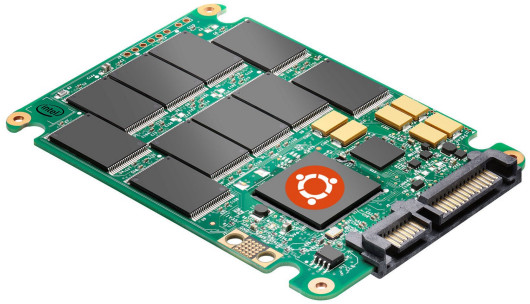
Disk Memory Basics

- Hard disk drive (HDD), hard disk, hard drive is the dominant secondary storage device in computer systems

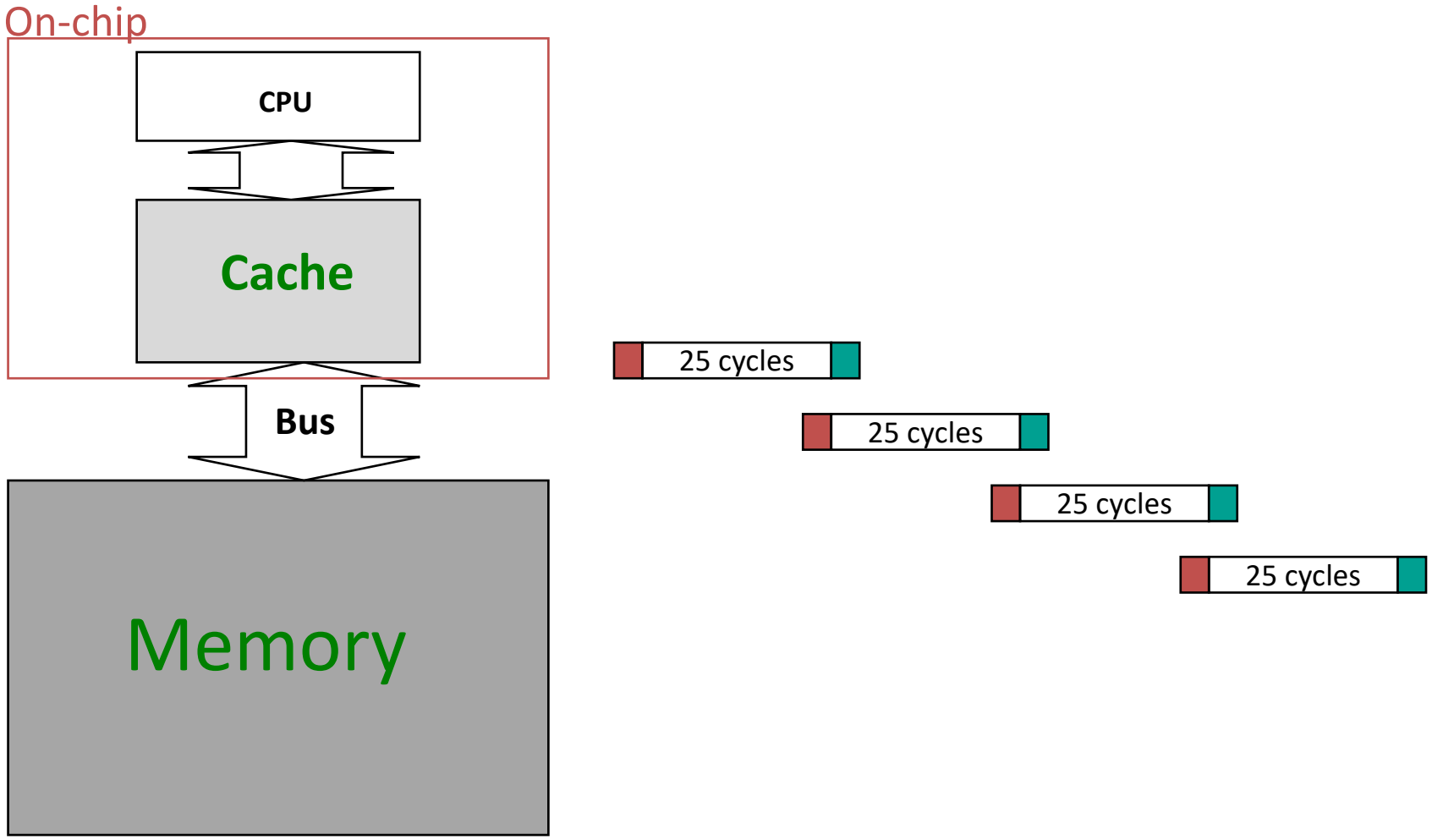


Disk Memory Basics

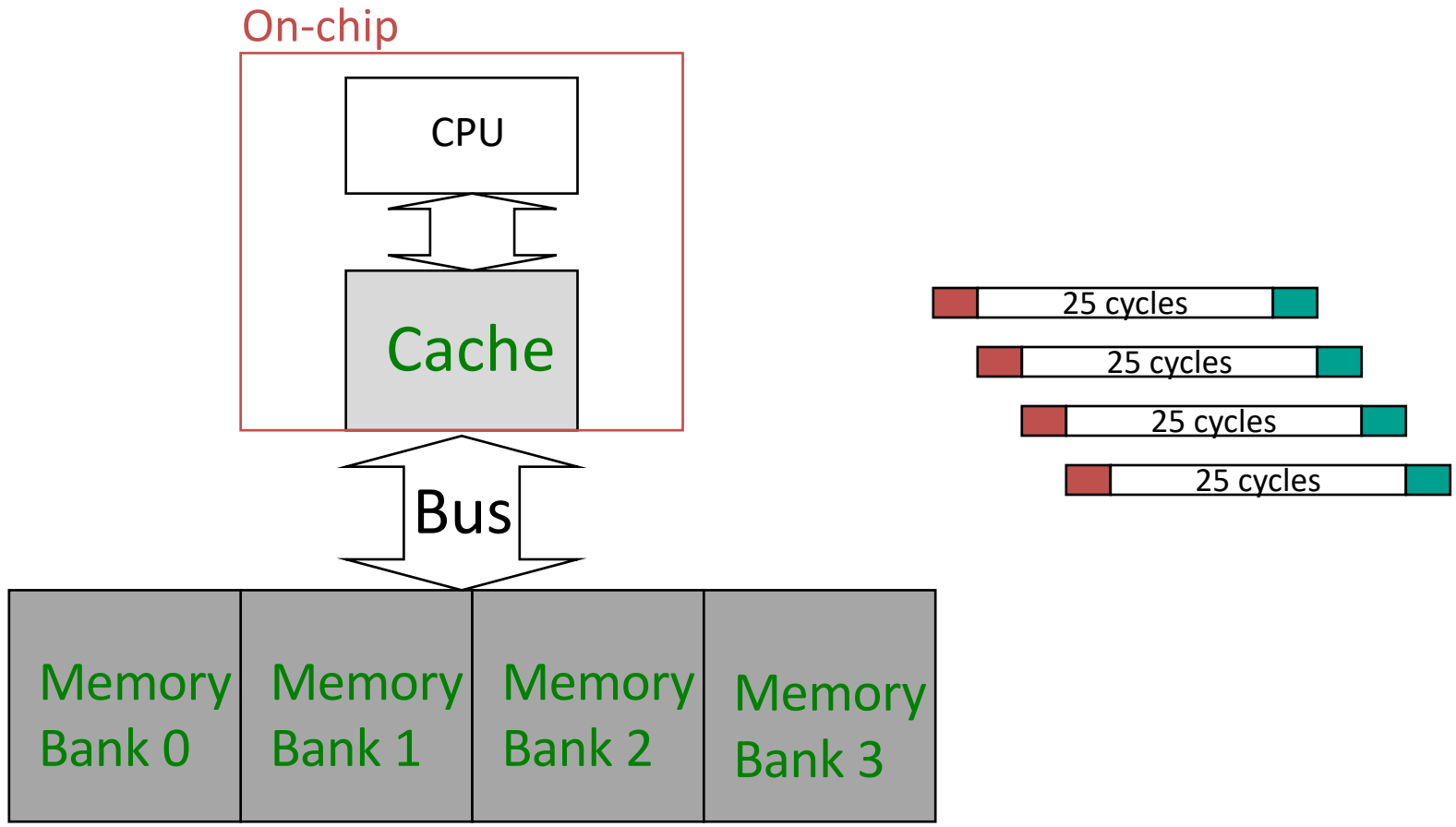
- Solid-State Drive (SSD) uses integrated circuits and has no moving mechanical components
  - Low latency
  - Low power
  - Solid state reliability
  - Widening application range
    - Embedded devices
    - Desktop and laptop PC
    - Server and supercomputer



Memory Organization

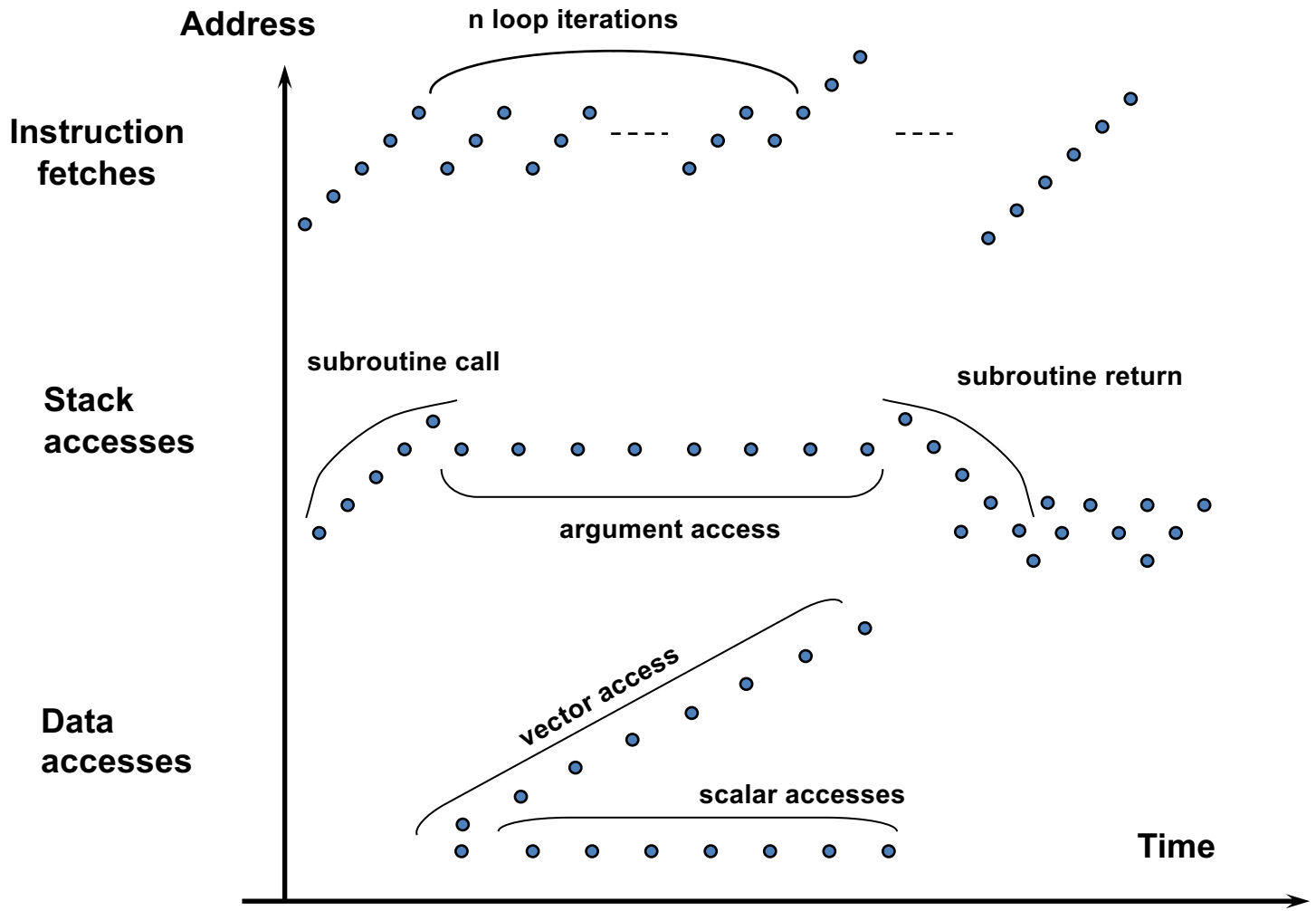


Memory Organization

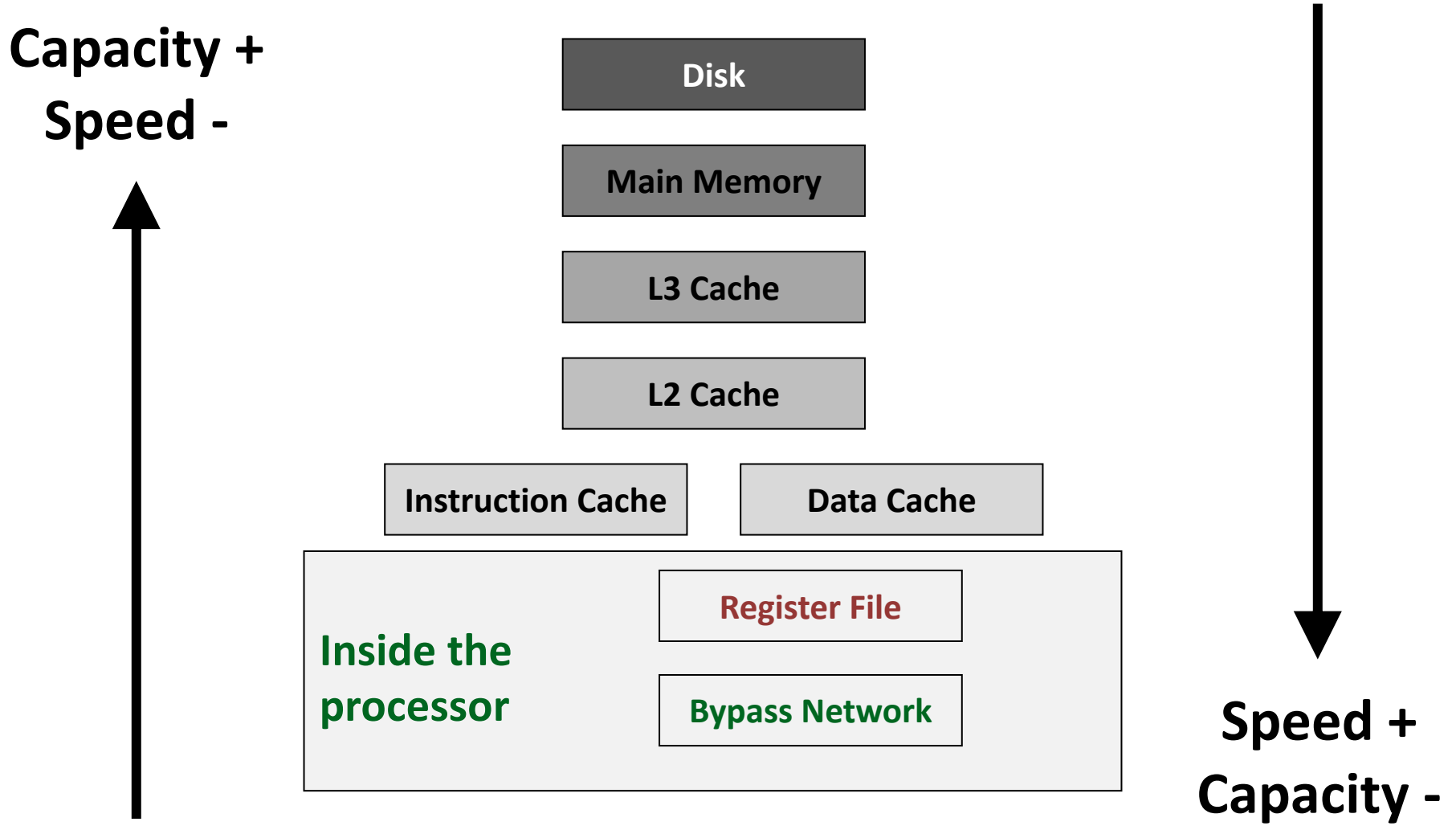


Interleaved Memory

Typical Memory Reference Patterns

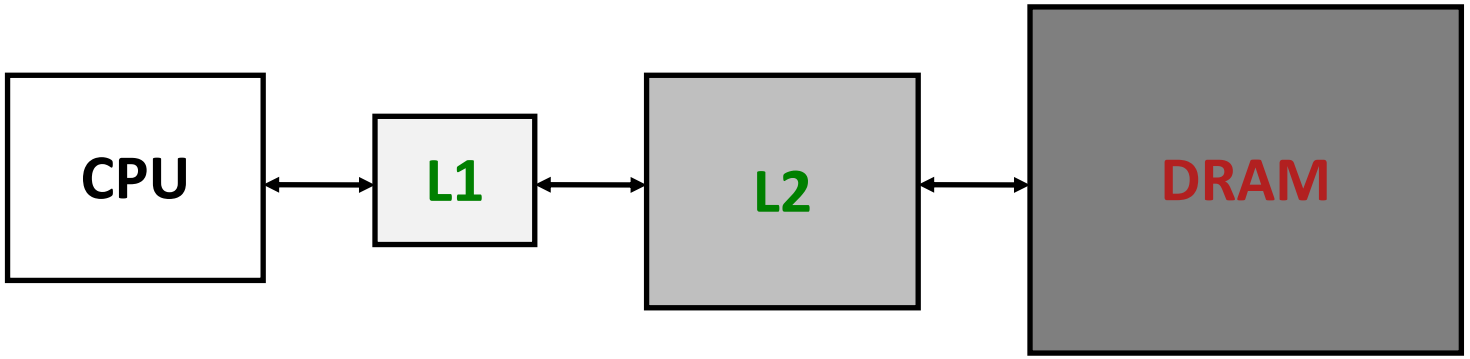


A Typical Memory Hierarchy



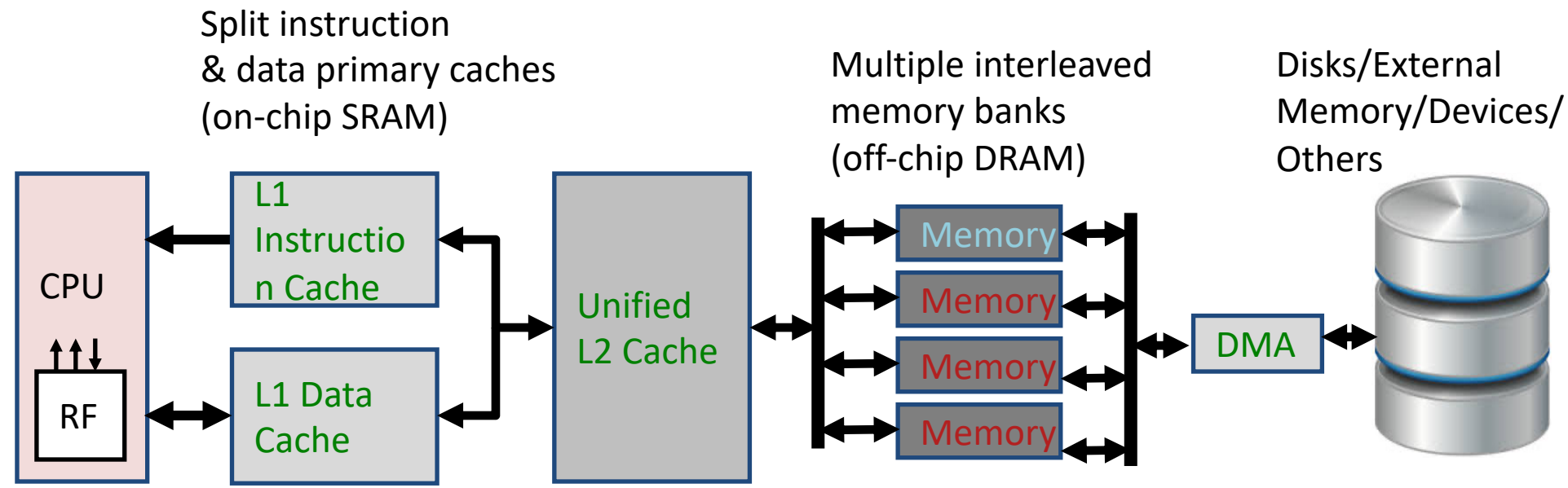
Memory Organization

- A memory cannot be large and fast
- Increasing sizes of cache at each level



  - A hit at a level occurs if that level of the memory contains the data needed by the CPU
  - A miss occurs if the level does not contain the requested data

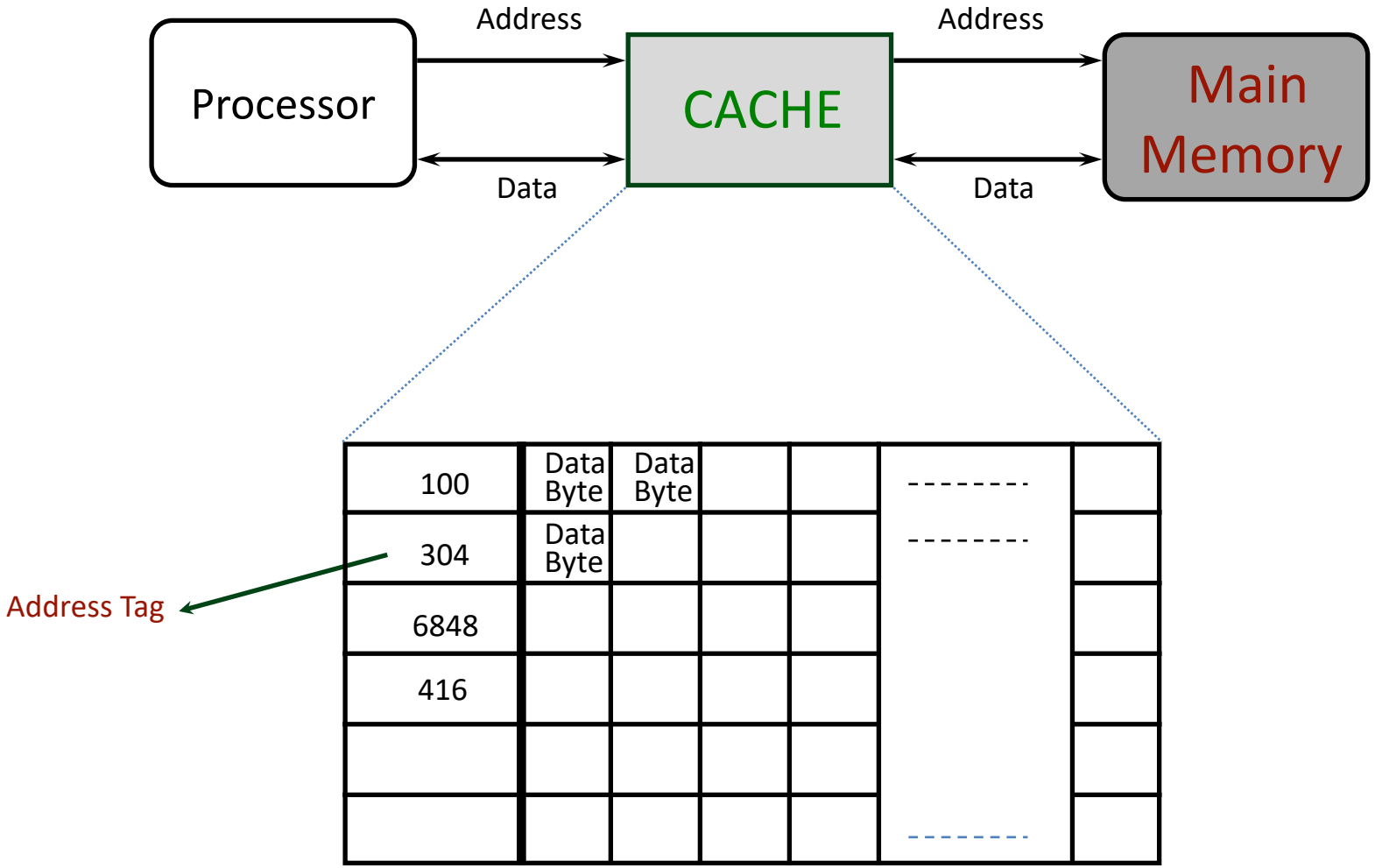
A Typical Memory Hierarchy



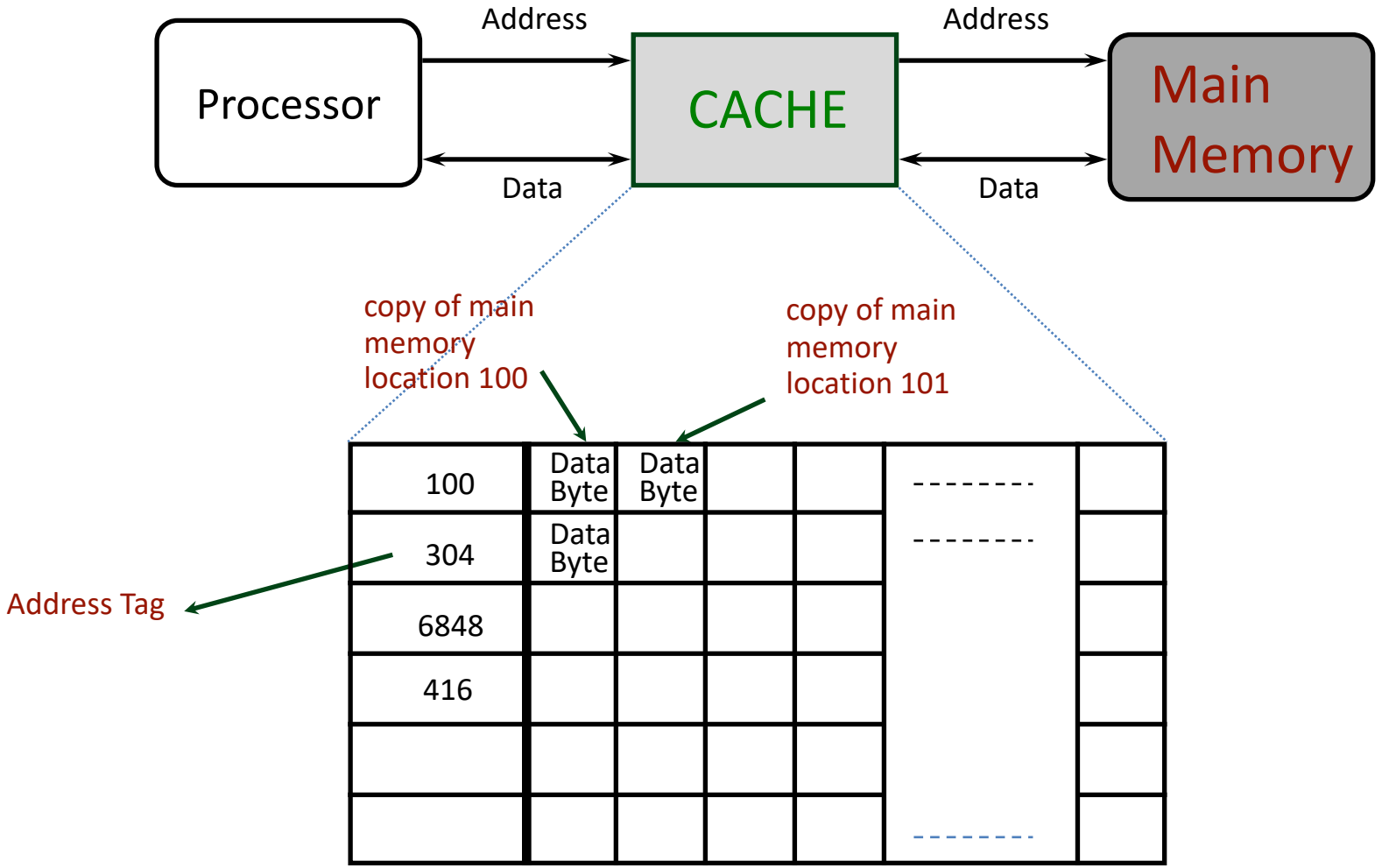
Definition of a Cache

- A cache is simply a copy of a small data segment residing in the main memory
  - Fast but small extra memory
  - Hold identical copies of main memory
  - Lower latency
  - Higher bandwidth
  - Usually several levels (1, 2 and 3)

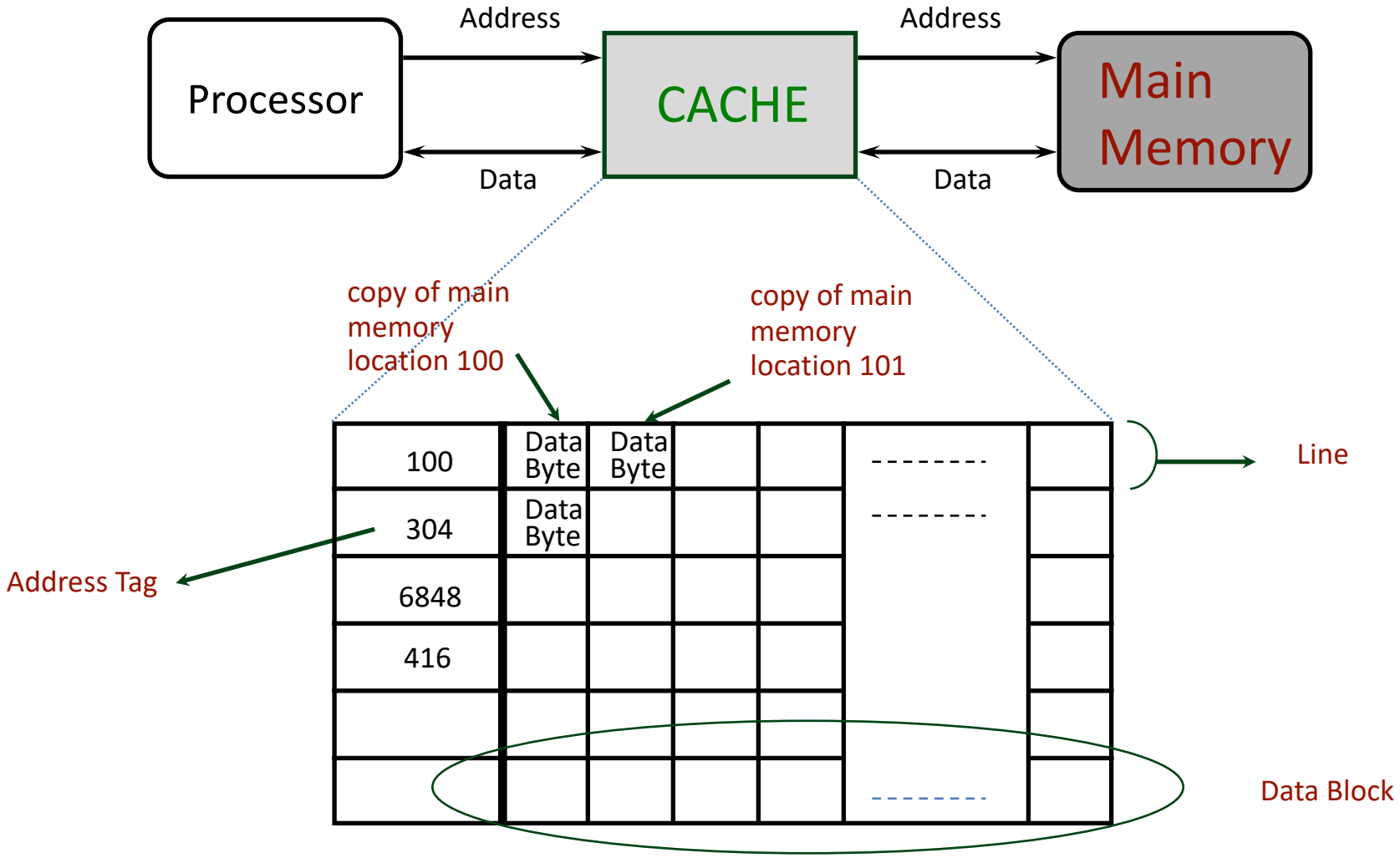
Cache Structures



Caching & Cache Structures

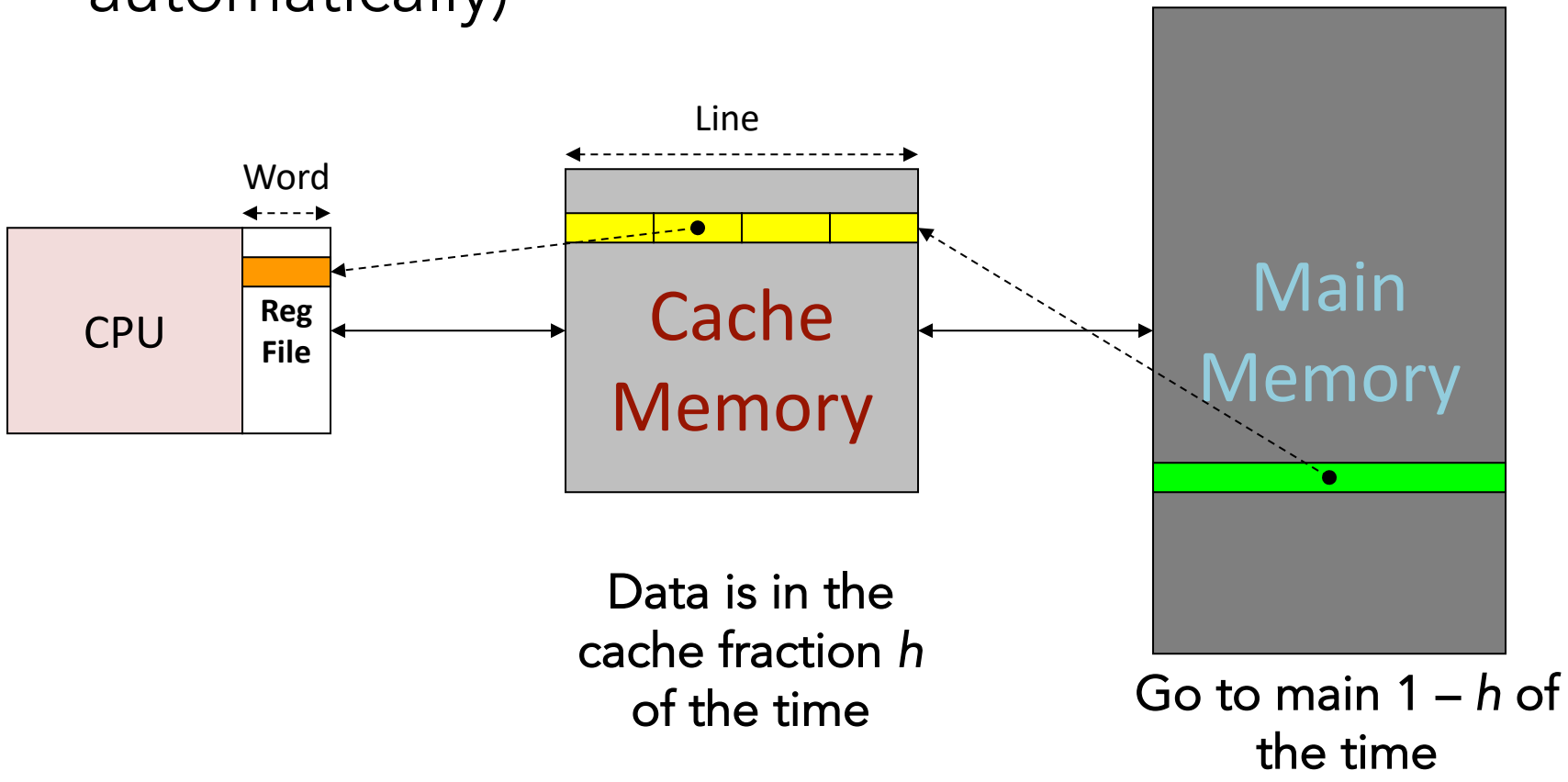


Caching & Cache Structures



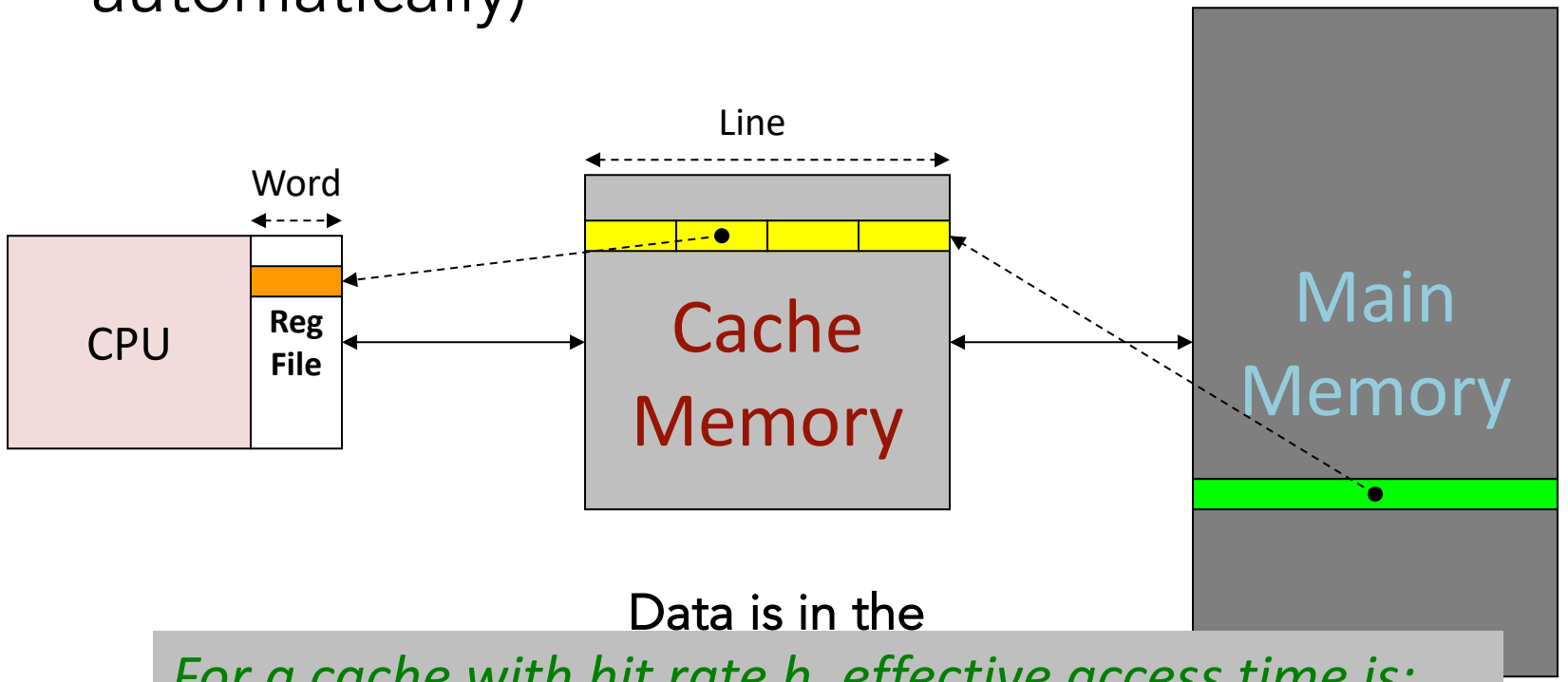
Multilevel Caches

- Cache is transparent to user (happens automatically)



Multilevel Caches

- Cache is transparent to user (happens automatically)



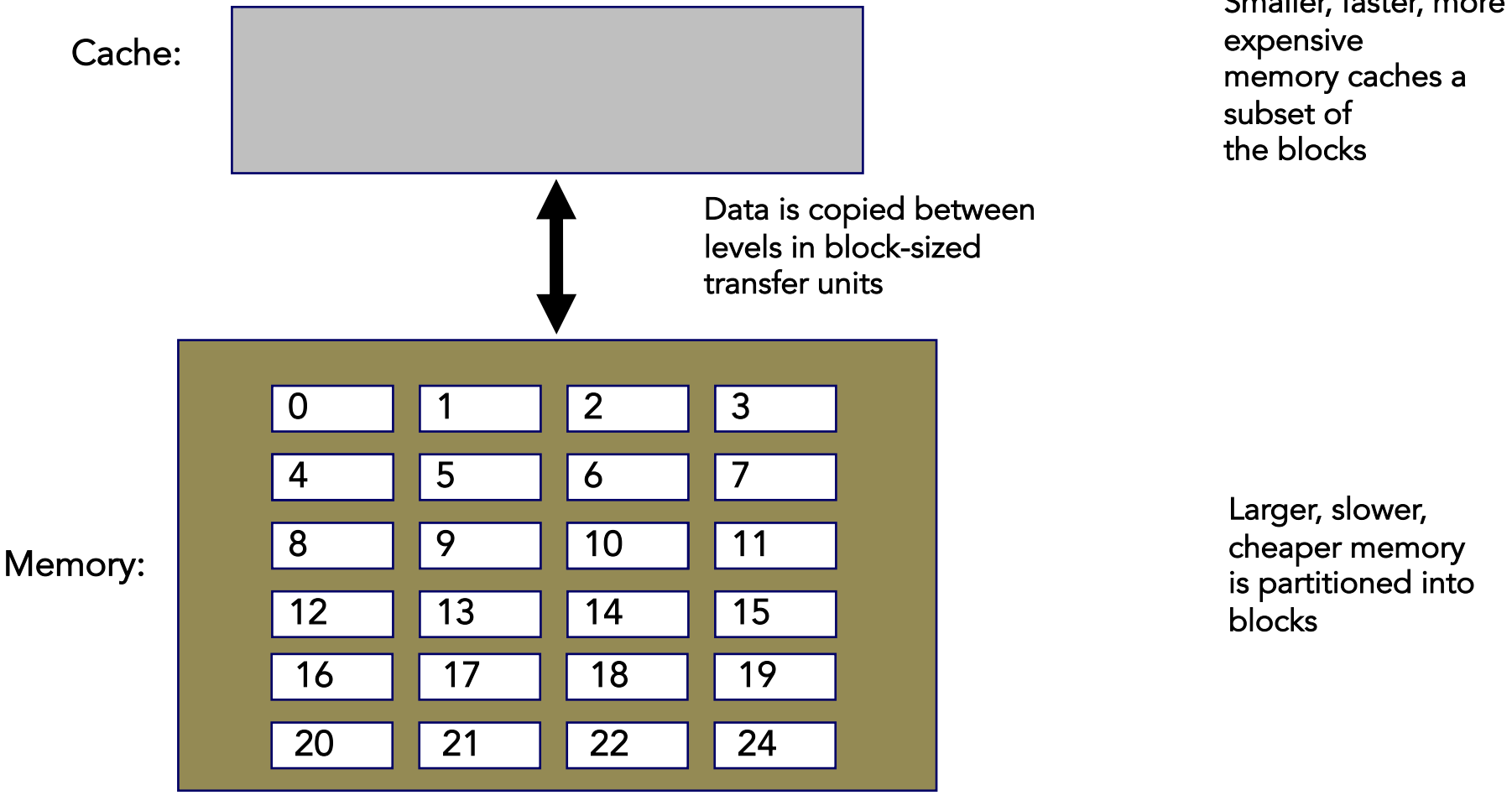
Data is in the

- *h* of

For a cache with hit rate h, effective access time is:

$C_{eff} = hC_{fast} + (1 - h)(C_{slow} + C_{fast}) = C_{fast} + (1 - h)C_{slow}$

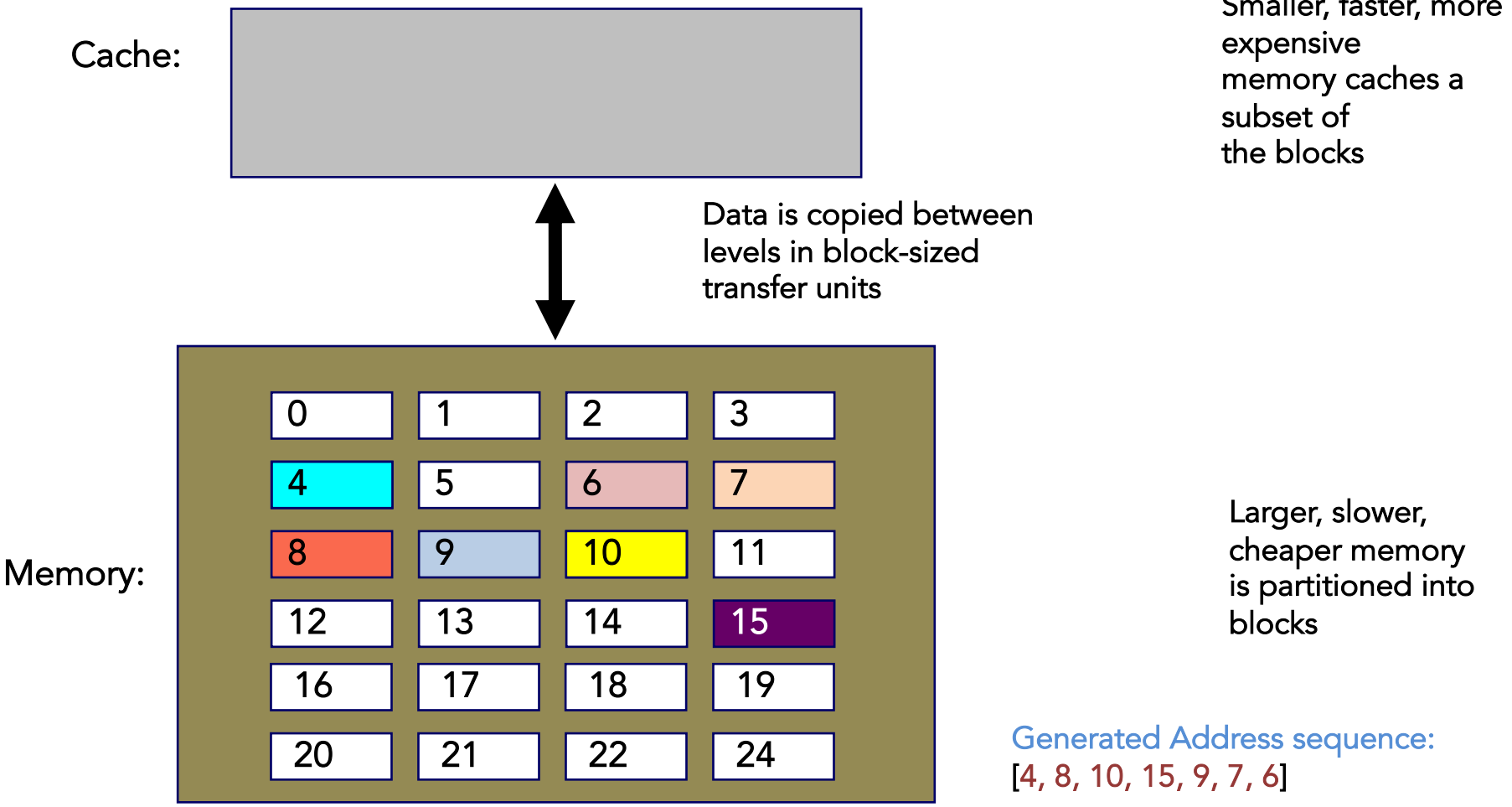
Caching Mechanism



Smaller, faster, more expensive memory caches a subset of the blocks

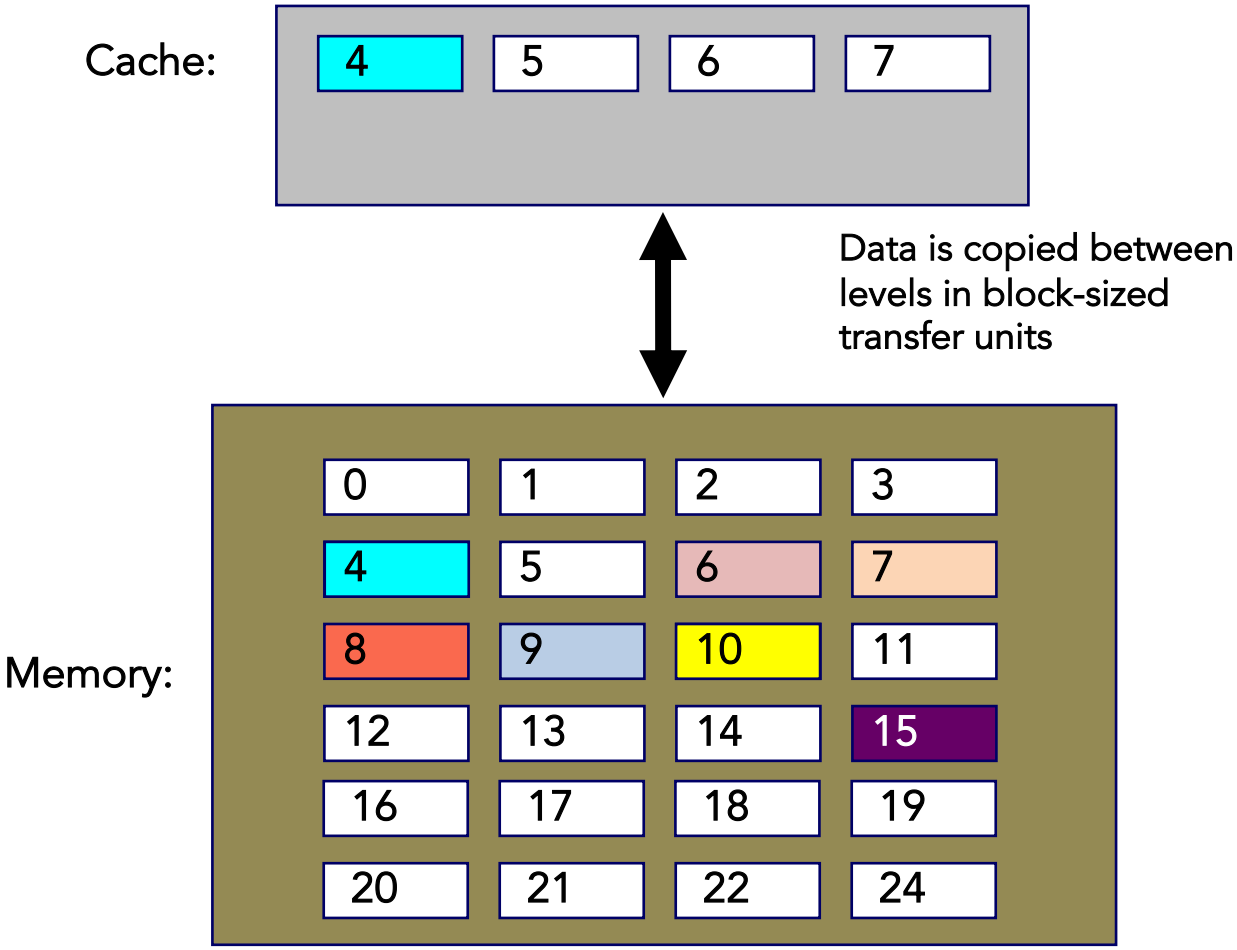
Larger, slower, cheaper memory is partitioned into blocks

Caching Mechanism



Generated Address sequence:
[4, 8, 10, 15, 9, 7, 6]

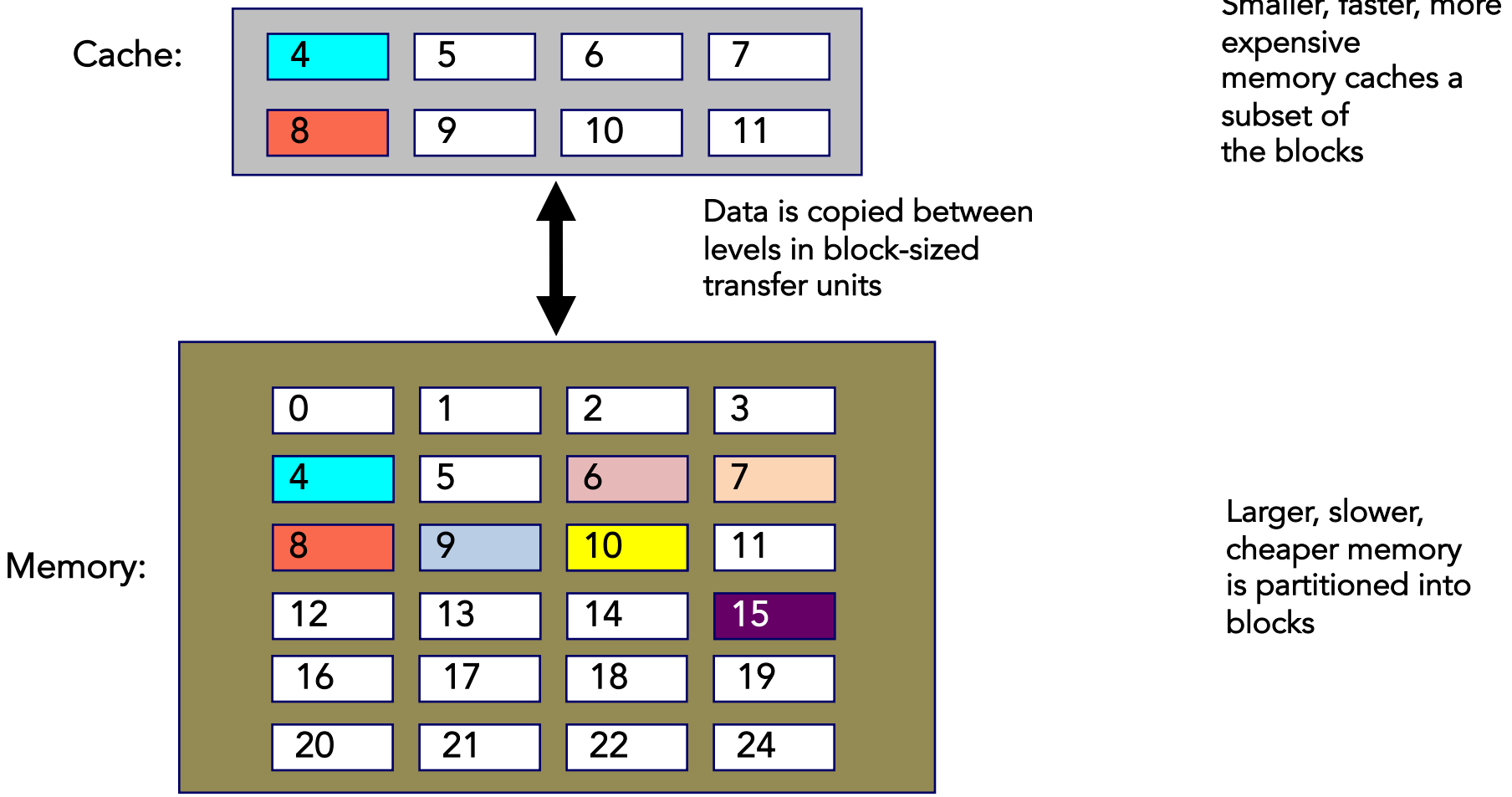
Caching Mechanism



Smaller, faster, more expensive memory caches a subset of the blocks

Larger, slower, cheaper memory is partitioned into blocks

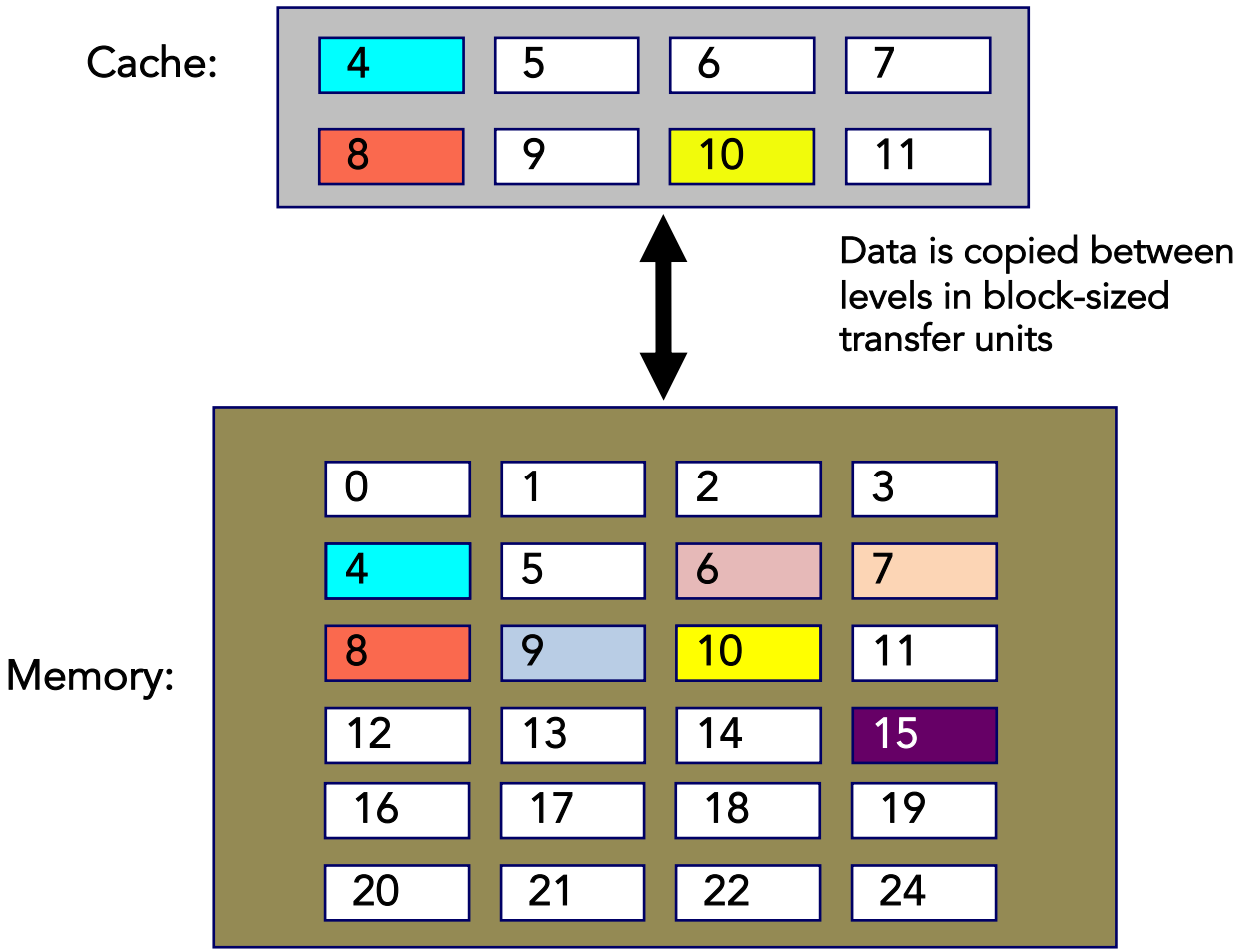
Caching Mechanism



Smaller, faster, more expensive memory caches a subset of the blocks

Larger, slower, cheaper memory is partitioned into blocks

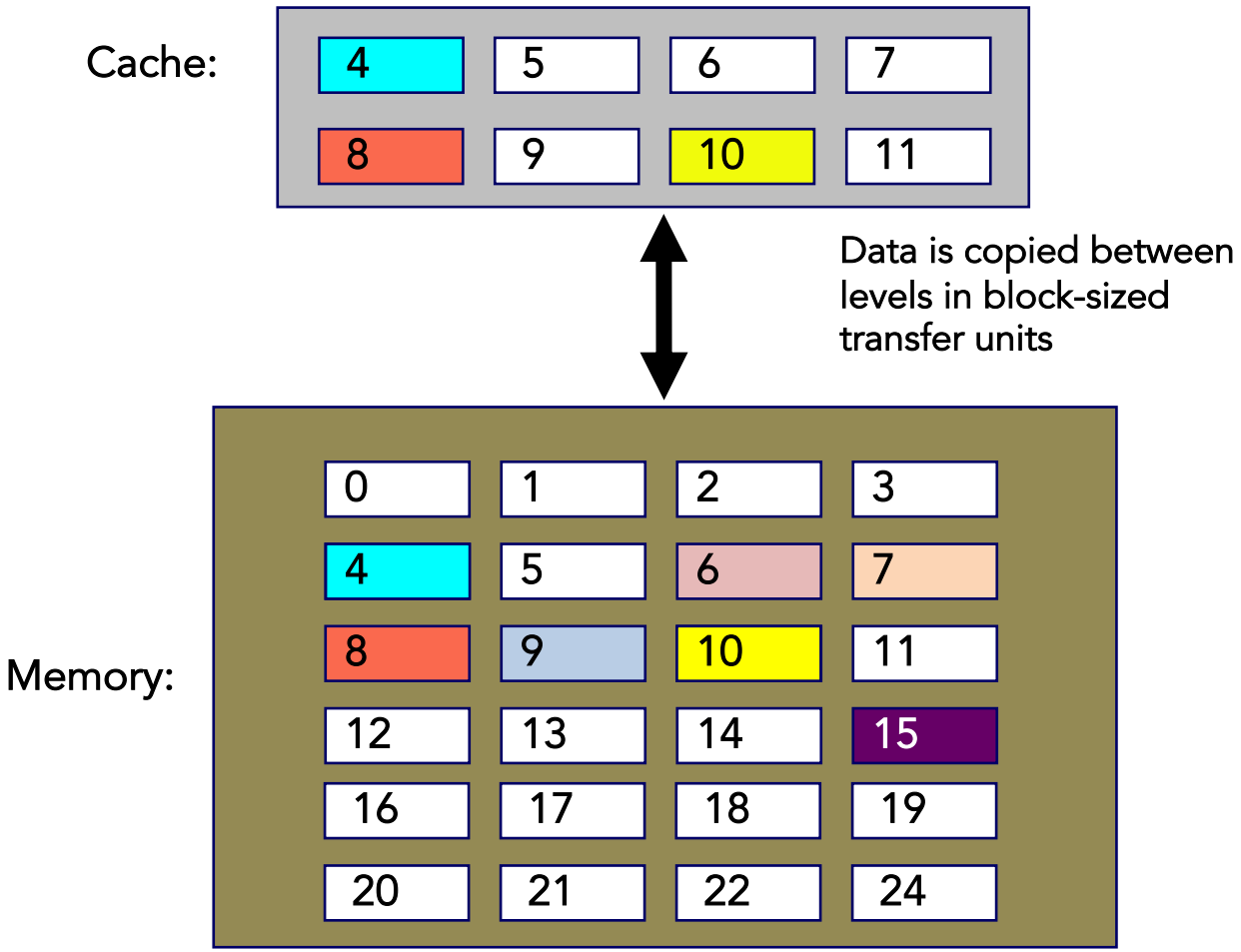
Caching Mechanism



Smaller, faster, more expensive memory caches a subset of the blocks

Larger, slower, cheaper memory is partitioned into blocks

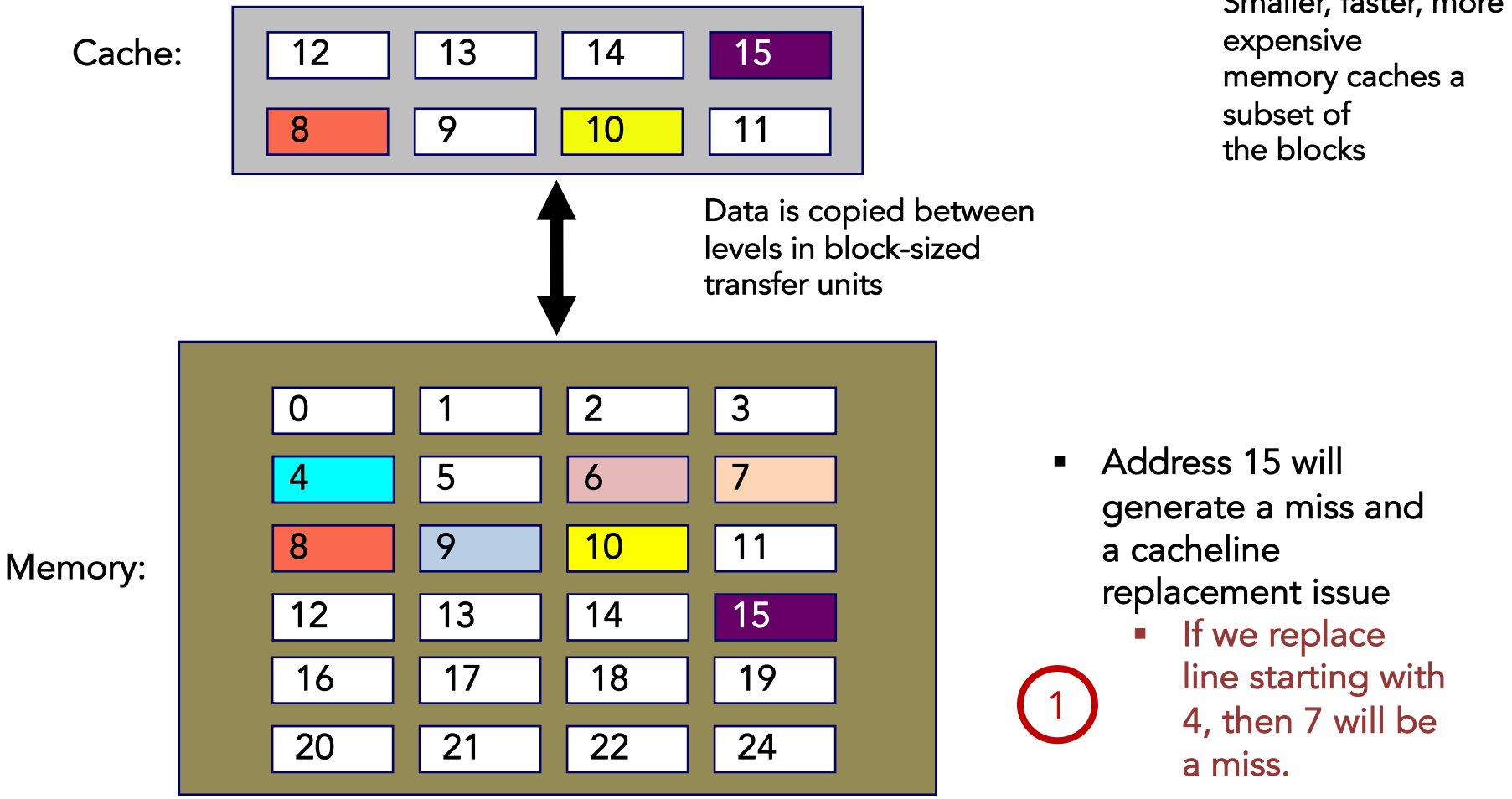
Caching Mechanism



Smaller, faster, more expensive memory caches a subset of the blocks

- Address 15 will generate a miss and a cacheline replacement issue

Caching Mechanism

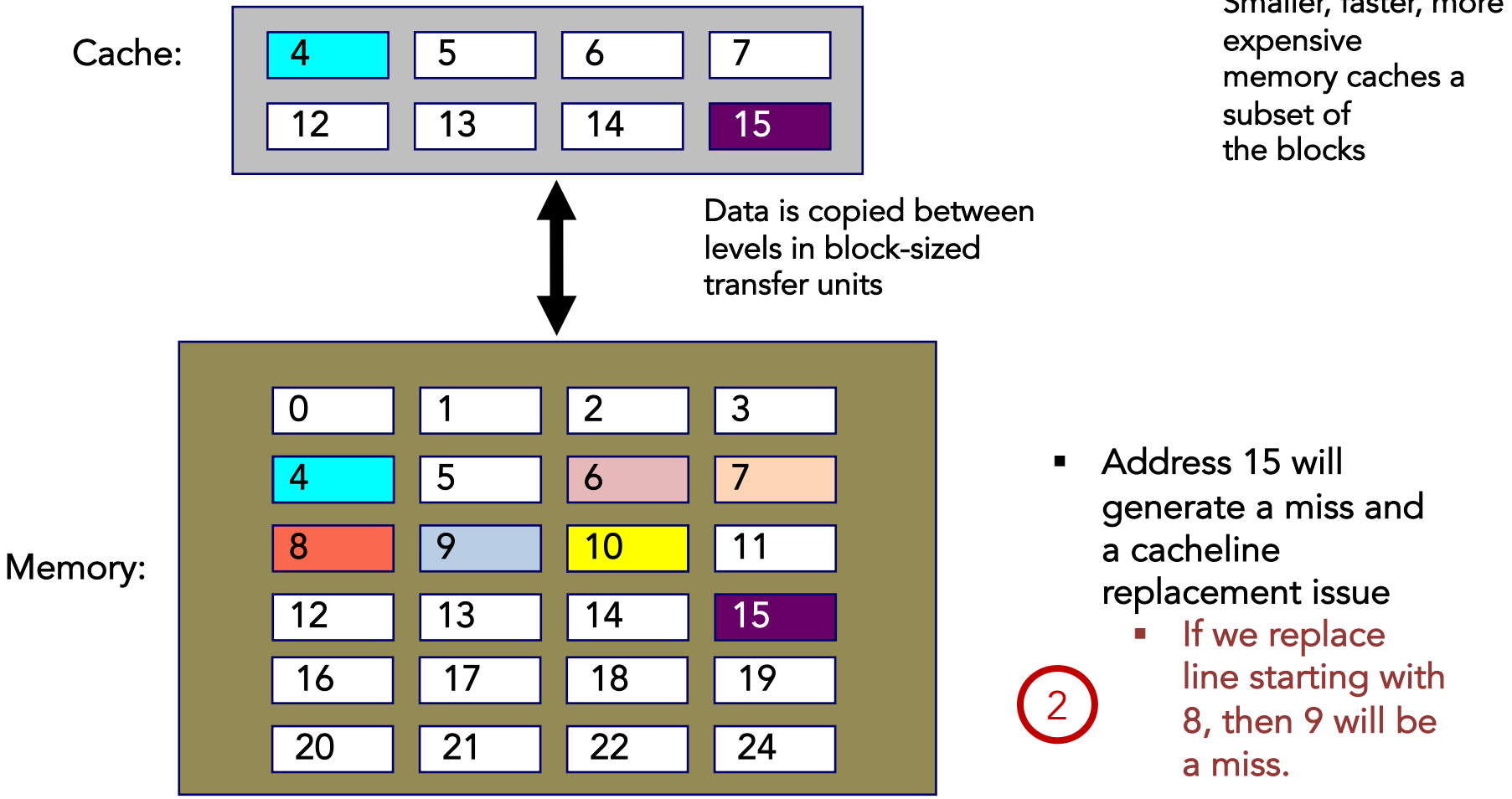


Smaller, faster, more expensive memory caches a subset of the blocks

- Address 15 will generate a miss and a cacheline replacement issue
  - If we replace line starting with 4, then 7 will be a miss.

1

Caching Mechanism

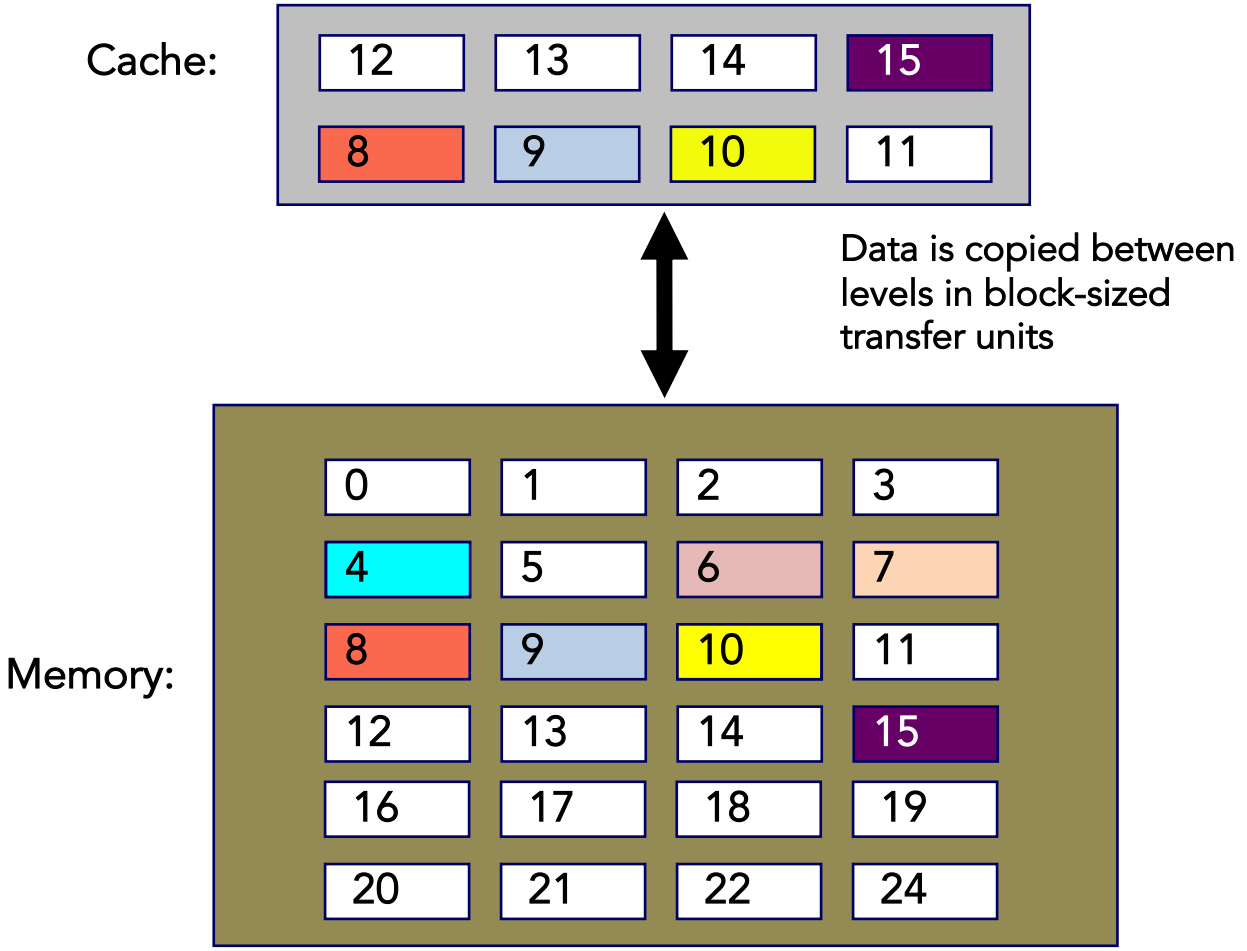


Smaller, faster, more expensive memory caches a subset of the blocks

- Address 15 will generate a miss and a cacheline replacement issue
  - If we replace line starting with 8, then 9 will be a miss.

2

Caching Mechanism

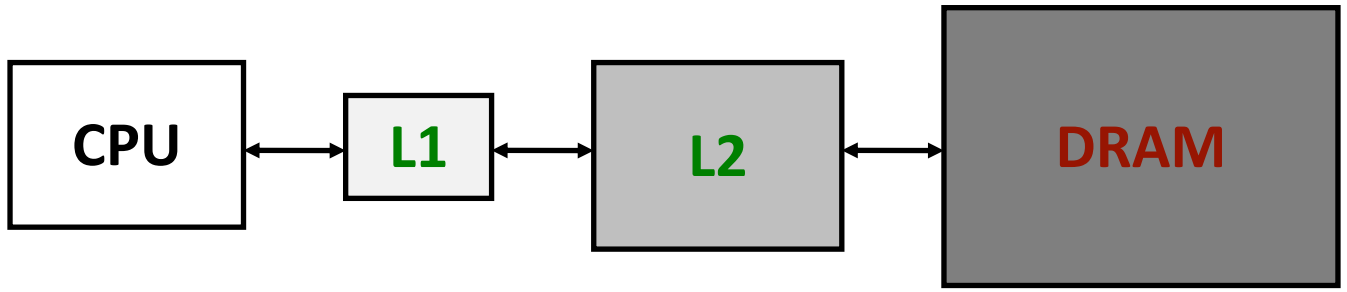


Smaller, faster, more expensive memory caches a subset of the blocks

- Address 15 will generate a miss and a cacheline replacement issue
  - Least Recently Used (LRU) replacement policy

Caches

- This organization works because most programs exhibit locality
  - The principle of temporal locality says that if a program accesses one memory address, there is a good chance that it will access the same address in the near future
  - The principle of spatial locality says that if a program accesses one memory address, there is a good chance that it will also access other nearby addresses



Caches

- Loops are excellent examples of temporal locality in programs

```
int i = 10;
while ( i > 0 )
{     j = i %2;
      sum += Array[j];
      i --;
 }
```

Caches

- Loops are excellent examples of temporal locality in programs
  - The loop body will be executed many times
  - The CPU will need to access those same few locations of the instruction memory repeatedly

```
loop:  lw   t2, 0(t1)      # place next element in t2
       add  a2, a2, t2     # sum = sum + array[i]
       addi t1, t1, 4      # point to next element
       addi t0, t0, -1     # i--
       bgtz t0, loop       # i > 0?
```

Caches

- Nearly every program exhibits spatial locality

```
student.name = "Albert Bitdiddle";
student.major = "Computer Engineering";
student.year    = "Junior";
Student.gps      = 3.95;
```

```
int sum = 0;
for (i = 0; i < N; i++)
        sum += Array[i];
return sum;
```

Caches

- Nearly every program exhibits spatial locality
  - Instructions are usually executed in sequence
- Loops exhibit *both* temporal and spatial locality

```
loop:  lw   t2, 0(t1)      # place next element in t2
       add  a2, a2, t2     # sum = sum + array[i]
       addi t1, t1, 4      # point to next element
       addi t0, t0, -1     # i--
       bgtz t0, loop       # i > 0?
```

Next Learning Module

- Caching Principles