The Memory Hierarchy

Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.

Seoul National University

Spring 2019

Random-Access Memory (RAM)

- Key features
  - RAM is traditionally packaged as a chip
  - Basic storage unit is a cell (one bit per cell). Multiple RAM chips form a memory.
- SRAM (Static RAM)
  - Each cell stores a bit with a four or six-transistor circuit
  - Retains value indefinitely, as long as it is kept powered
  - Faster and more expensive than DRAM
- DRAM (Dynamic RAM)
  - Each cell stores a bit with a capacitor. One transistor is used for access.
  - Value must be refreshed every 10 – 100 ms.
  - Slower and cheaper than SRAM

SRAM vs DRAM

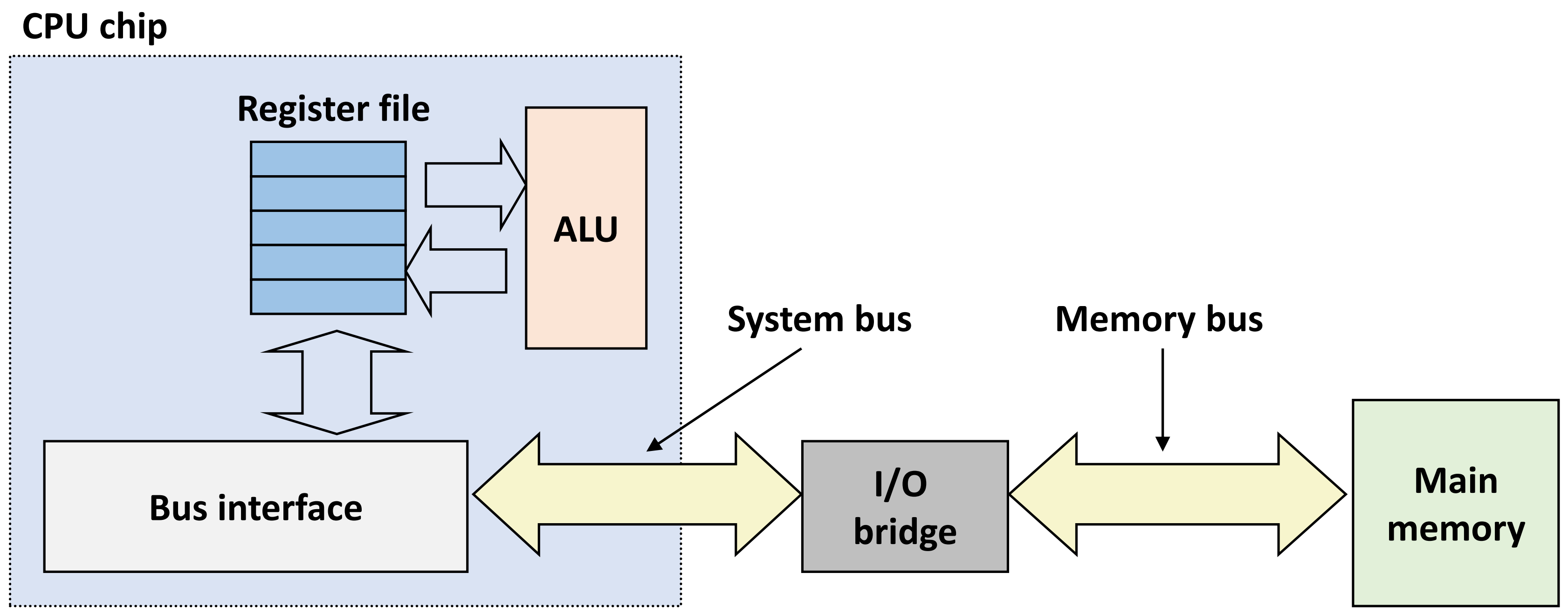
	TRs per bit	Access time	Needs Refresh?	Needs EDC?	Cost	Applications
SRAM	4 or 6	1x	No	Maybe	100x	Cache memories
DRAM	1	10x	Yes	Yes	1x	Main memories, Frame buffers

Nonvolatile Memories

- DRAM and SRAM are volatile memories
  - Lose information if powered off
- 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 memories: EEPROMs with partial (block-level) erase capability
- Uses for nonvolatile memories
  - Firmware programs stored in a ROM (BIOS, Disk/network/graphics controllers, …)
  - USB drives, smartphones, tablets, SSDs (Solid-State Drives), disk caches, …

Traditional Bus Structure

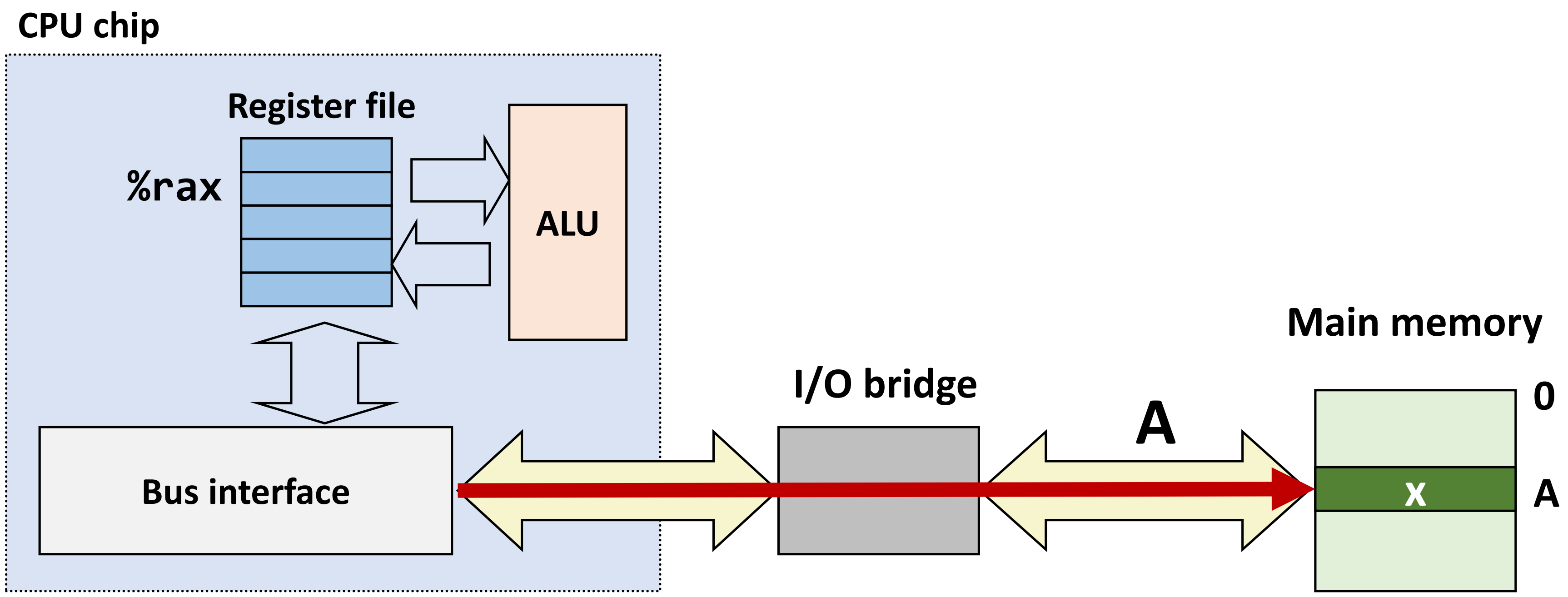
- A bus is a collection of parallel wires that carry address, data, and control signals among CPU, memory, and I/O devices
- Buses are typically shared by CPU, memory, and I/O devices



Memory Read Transaction (1)

- CPU places address A on the memory bus

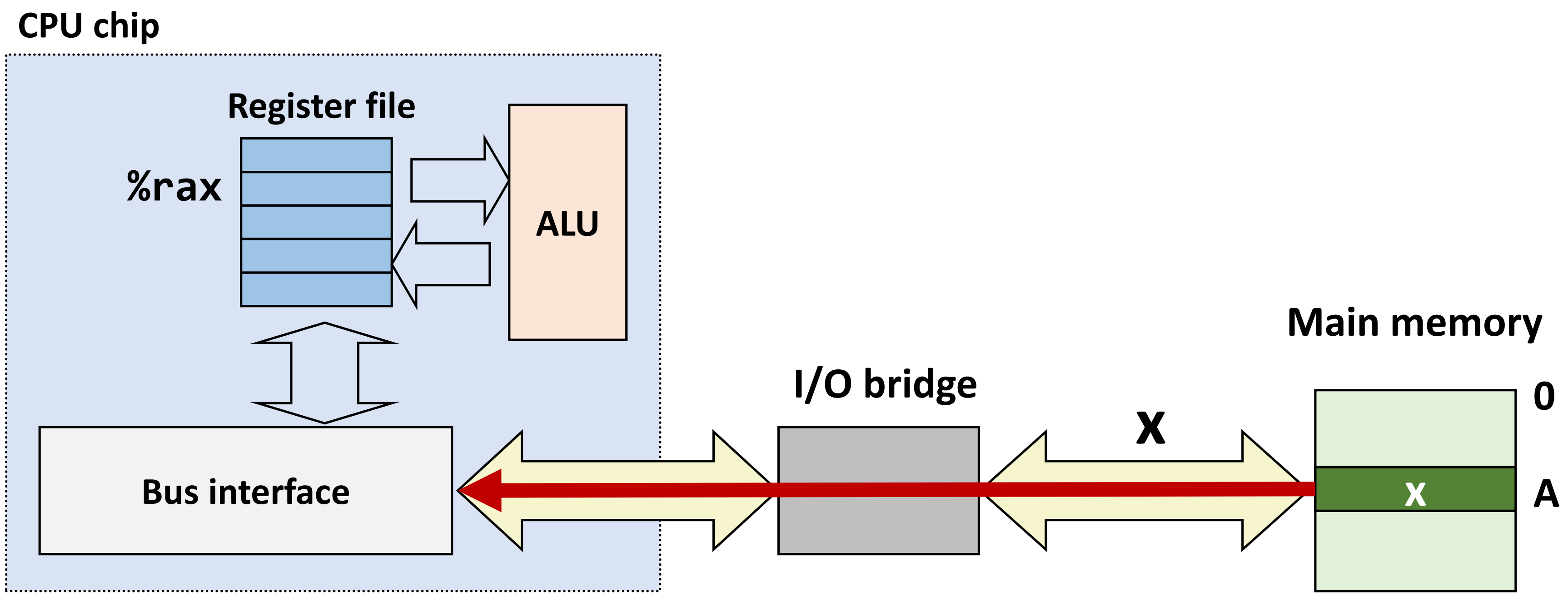
Load operation: `movq A, %rax`



Memory Read Transaction (2)

- Main memory reads A from the memory bus, retrieves word x, and places it on the bus

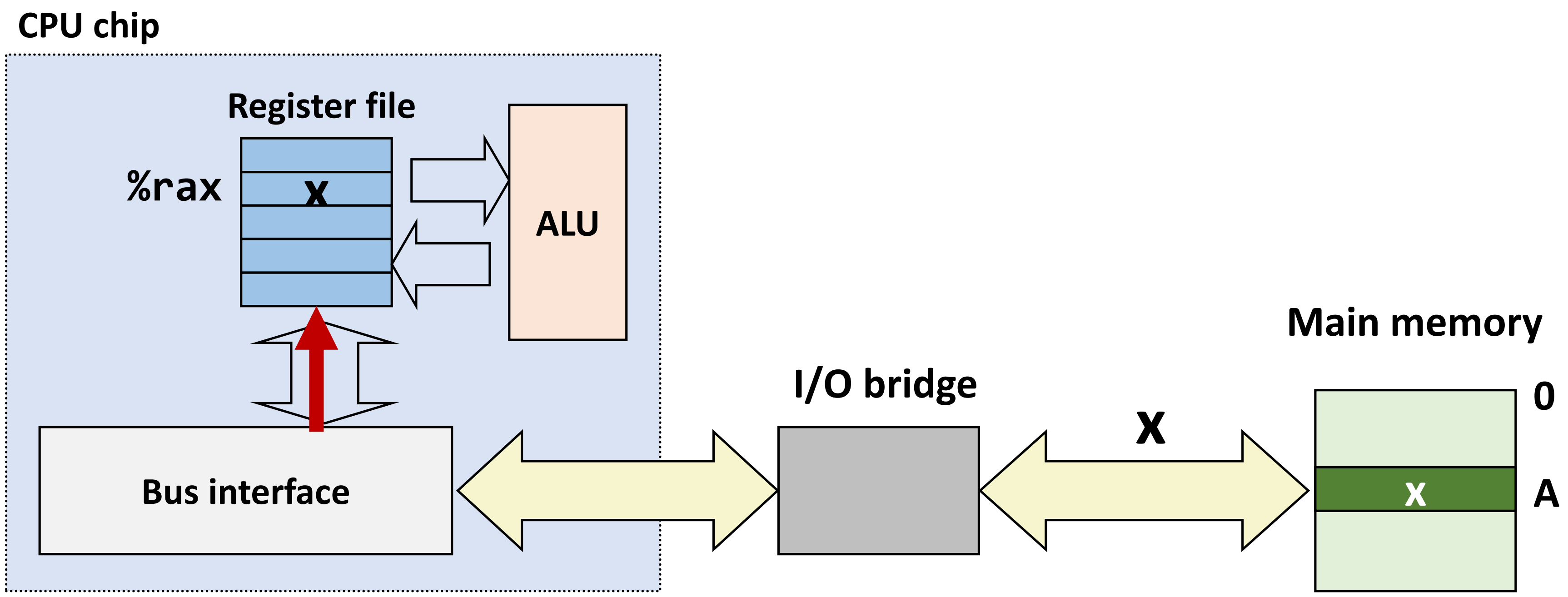
Load operation: `movq A, %rax`



Memory Read Transaction (3)

- CPU reads word x from the bus and copies it into register `%rax`

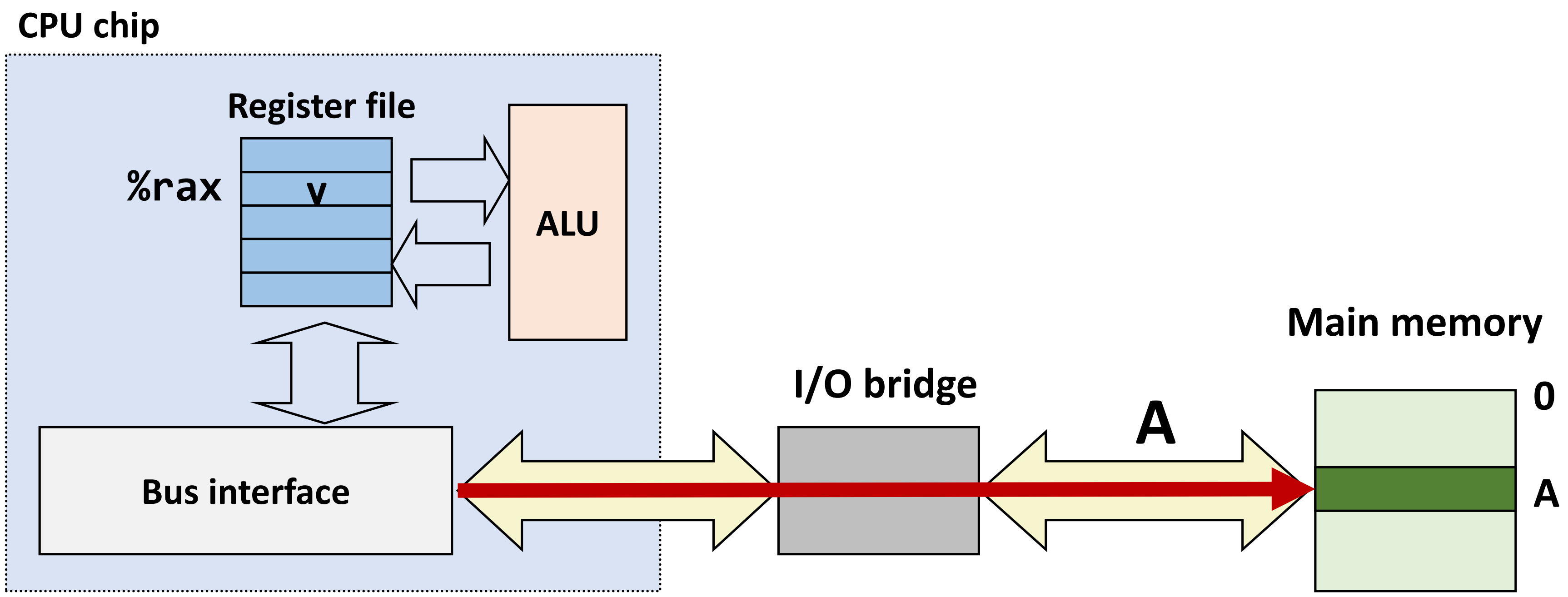
Load operation: **`movq A, %rax`**



Memory Write Transaction (1)

- CPU places address A on the bus. Main memory reads it and waits for the corresponding data word to arrive.

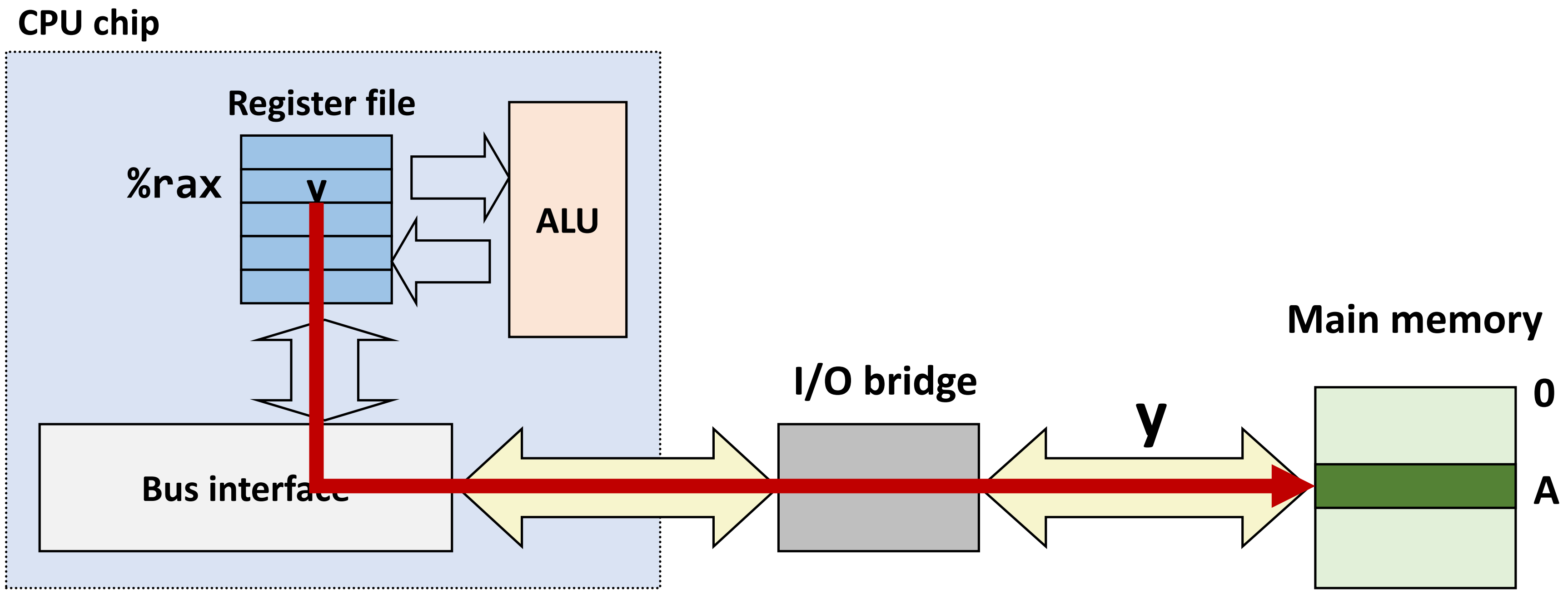
Store operation: `movq %rax, A`



Memory Write Transaction (2)

- CPU places data word y on the bus

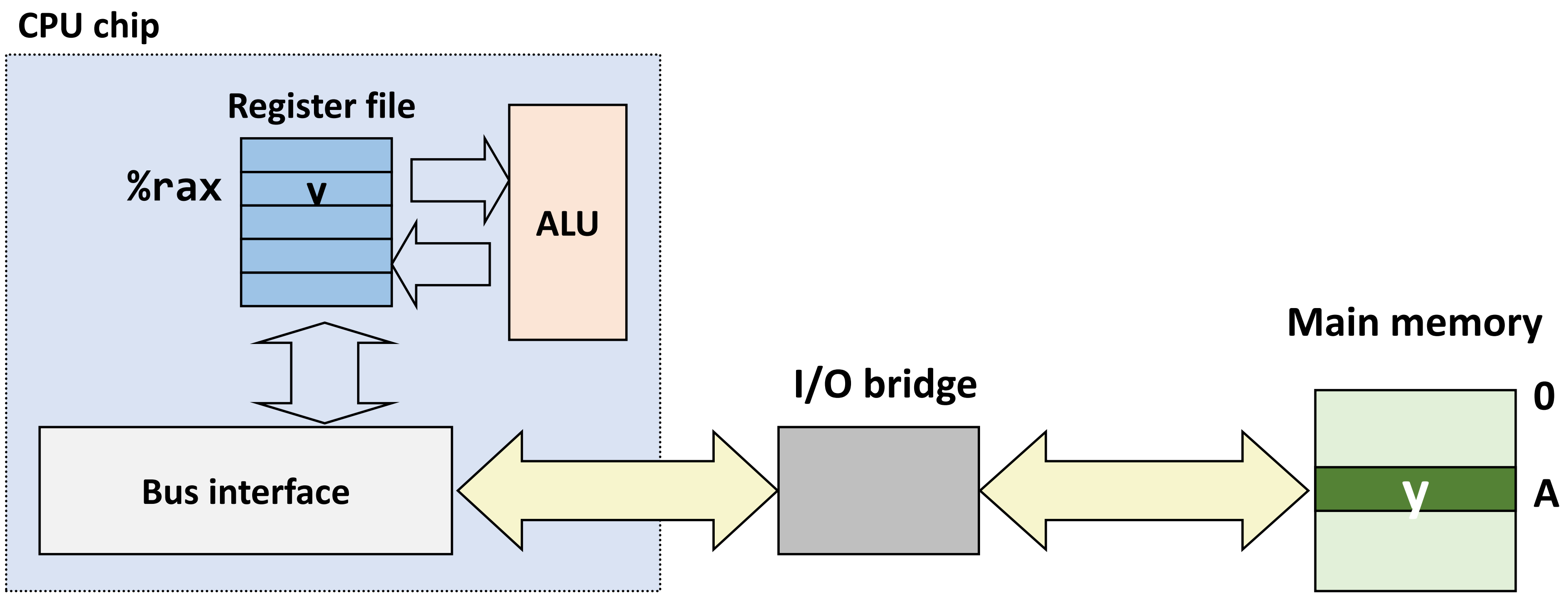
Store operation: `movq %rax, A`



Memory Write Transaction (3)

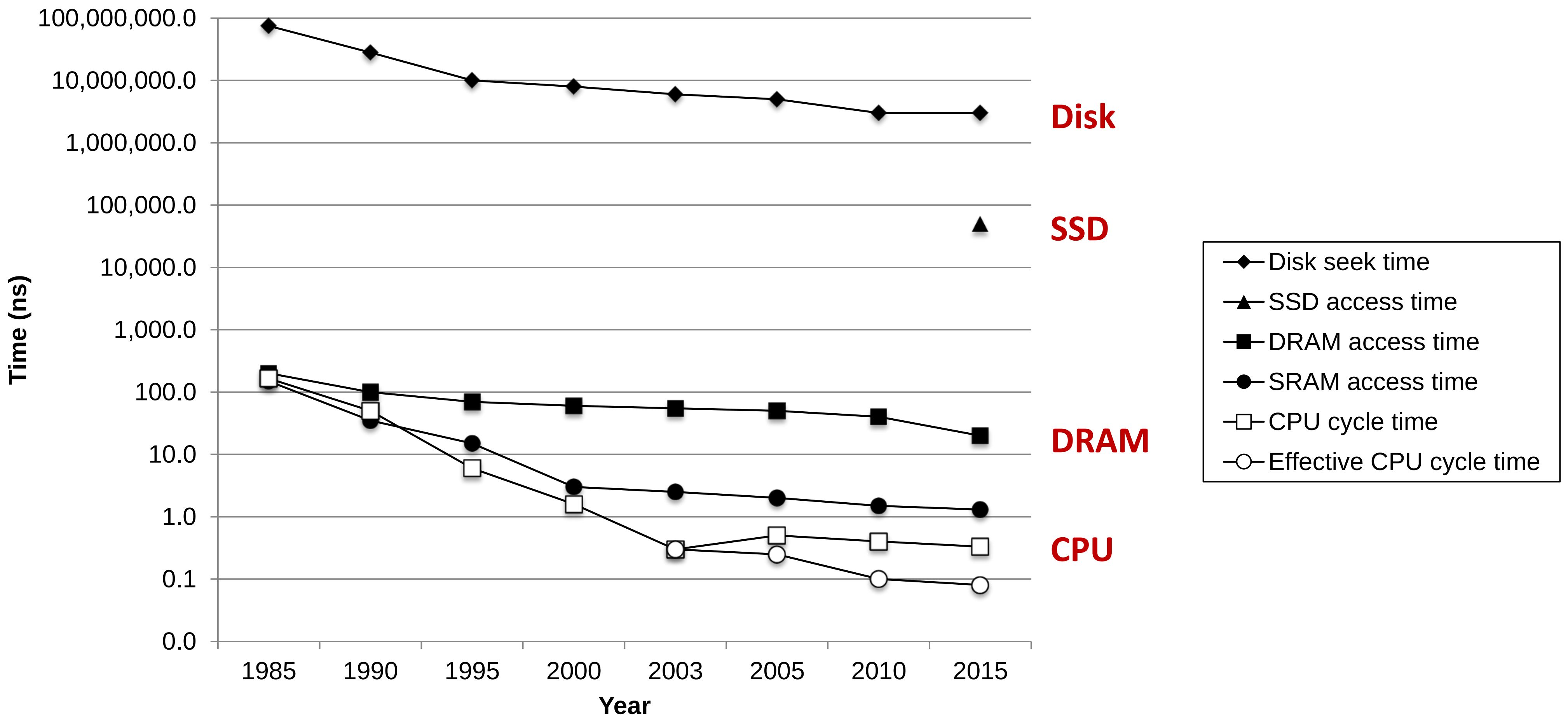
- Main memory reads data word y from the bus and stores it at address A

Store operation: `movq %rax, A`



The CPU-Memory Gap

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



Storage Trends

SRAM

Metric	**1985**	**1990**	**1995**	**2000**	**2005**	**2010**	**2015**	***2015:1985***
$/MB	2,900	320	256	100	75	60	*25*	*116*
access (ns)	150	35	15	3	2	1.5	*1.3*	*115*

DRAM

Metric	**1985**	**1990**	**1995**	**2000**	**2005**	**2010**	**2015**	***2015:1985***
$/MB	880	100	30	1	0.1	0.06	0.02	*44,000*
access (ns)	200	100	70	60	50	40	20	*10*
typical size (MB)	0.256	4	16	64	2,000	8,000	16.000	*62,500*

Disk

Metric	**1985**	**1990**	**1995**	**2000**	**2005**	**2010**	**2015**	***2015:1985***
$/GB	100,000	8,000	300	10	5	0.3	0.03	*3,333,333*
access (ms)	75	28	10	8	*5*	*3*	*3*	*25*
typical size (GB)	0.01	0.16	1	20	160	1,500	3,000	*300,000*

CPU Clock Rates

Inflection point in computer history when designers hit the "Power Wall"

	1985	1990	1995	2003	2005	2010	2015	2015:1985
CPU	80286	80386	Pentium	P-4	Core 2	Core i7(n)	Core i7(h)	
Clock rate (MHz)	6	20	150	3,300	2,000	2,500	3,000	500
Cycle time (ns)	166	50	6	0.30	0.50	0.4	0.33	500
Cores	1	1	1	1	2	4	4	4
Effective cycle time (ns)	166	50	6	0.30	0.25	0.10	0.08	2,075

(n) Nehalem processor (h) Haswell processor

Locality to the Rescue!

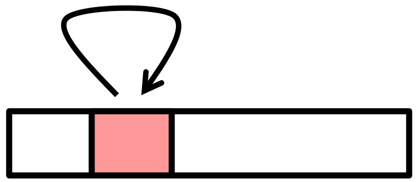
- Question:
  How can we make a memory as fast as SRAM and as cheap as DRAM (or even disk)?

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

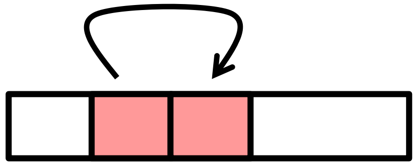
Principle of Locality

- 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



Principle of Locality: Example

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

- Data
  - Reference array elements in succession (stride-1 reference pattern) — Spatial locality
  - Reference **`sum`** each iteration — Temporal locality
- Instructions
  - Reference instructions in sequence — Spatial locality
  - Cycle through loop repeatedly — Temporal locality

Memory Hierarchy (1)

- 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

- These fundamental properties complement each other beautifully

- They suggest an approach for organizing memory and storage systems known is a memory hierarchy

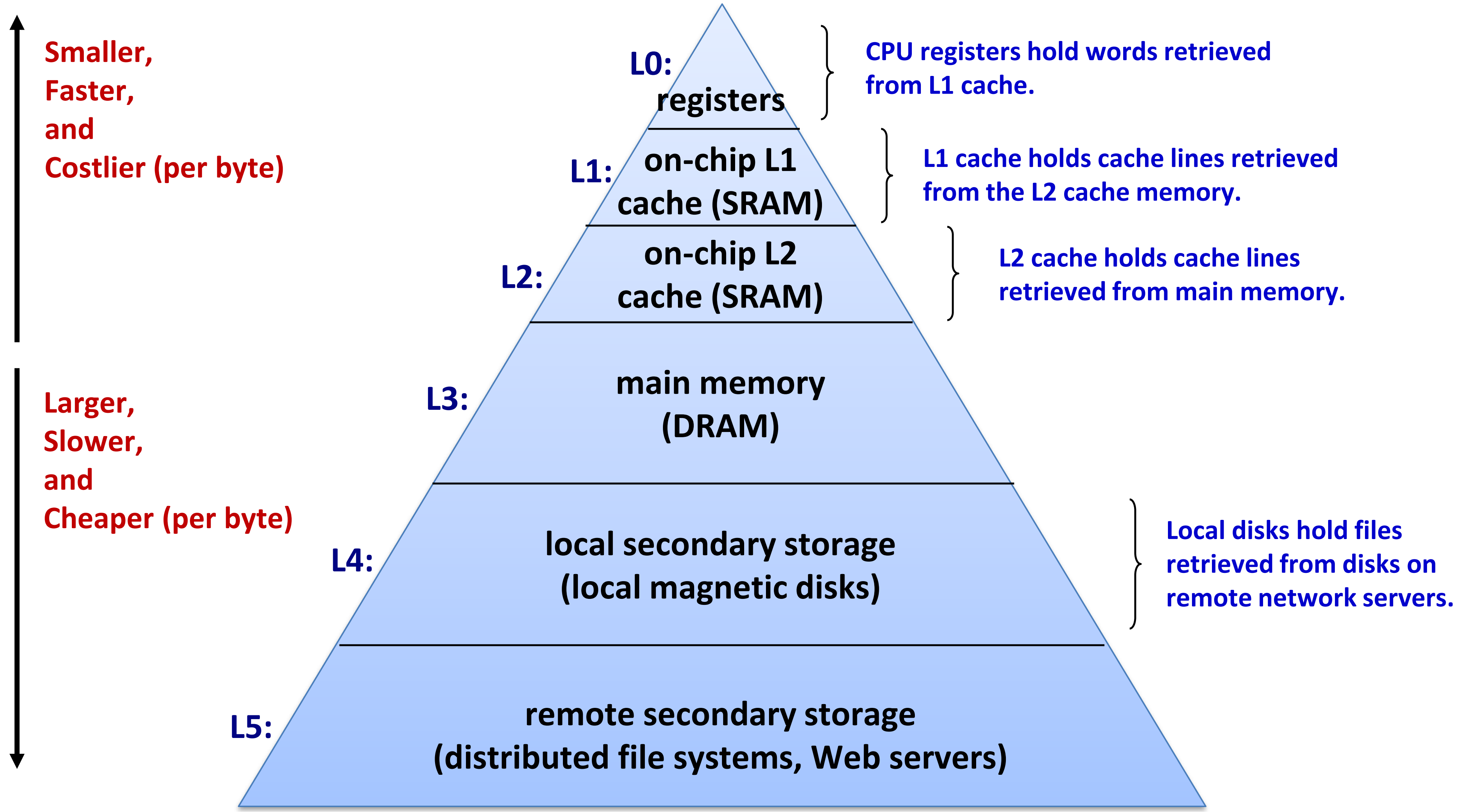
Memory Hierarchy (2)

“We are therefore forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible.”

-- A. W. Burks, H. H. Goldstein, J. von Neumann, Preliminary Discussion of the Logical Design of Electronic Computing Instrument, June 1946.

- Taking advantage of locality
  - Store everything on disk
  - Copy recently accessed (and nearby) items from disk to smaller DRAM memory (main memory)
  - Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory (cache memory)

Memory Hierarchy: Example



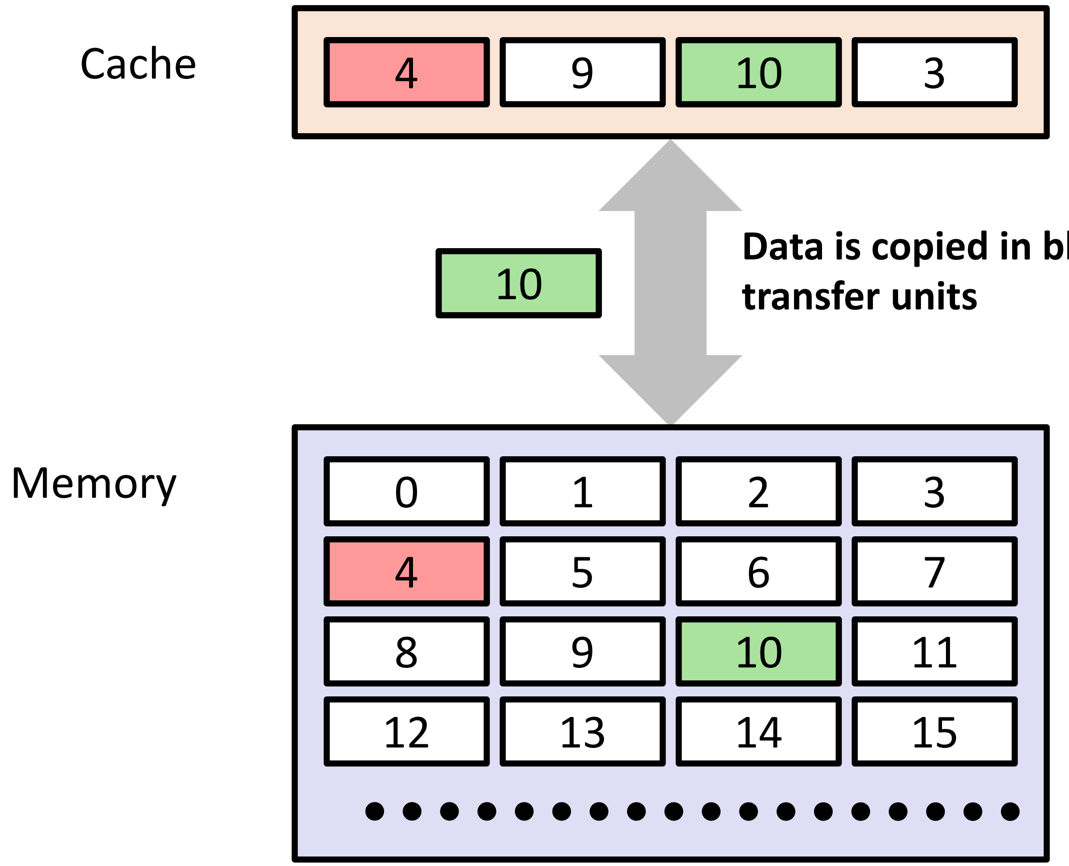
Smaller,
Faster,
and
Costlier (per byte)
Larger,
Slower,
and
Cheaper (per byte)
L0:
registers
L1:
on-chip L1 cache (SRAM)
L2:
on-chip L2 cache (SRAM)
L3:
main memory (DRAM)
L4:
local secondary storage (local magnetic disks)
L5:
remote secondary storage (distributed file systems, Web servers)
CPU registers hold words retrieved from L1 cache.
L1 cache holds cache lines retrieved from the L2 cache memory.
L2 cache holds cache lines retrieved from main memory.
Local disks hold files retrieved from disks on remote network servers.

Caches

- 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 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*

Big Idea: The memory hierarchy creates a large pool of storage that costs as much as the cheap storage near the bottom, but that serves data to programs at the rate of the fast storage near the top

General Cache Concepts

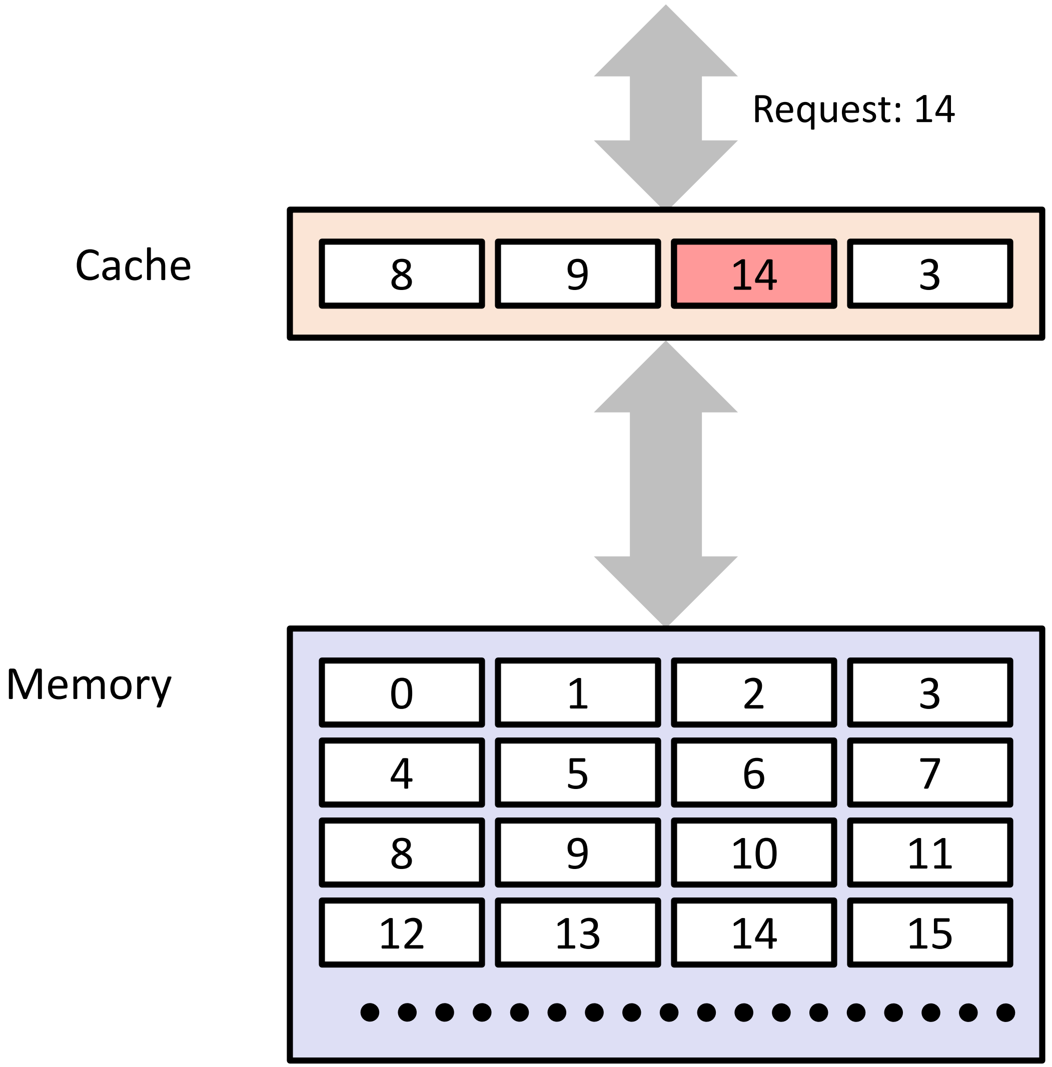


Smaller, faster, more expensive memory caches a subset of the blocks

Data is copied in block-sized transfer units

Larger, slower, cheaper memory viewed as partitioned into "blocks"

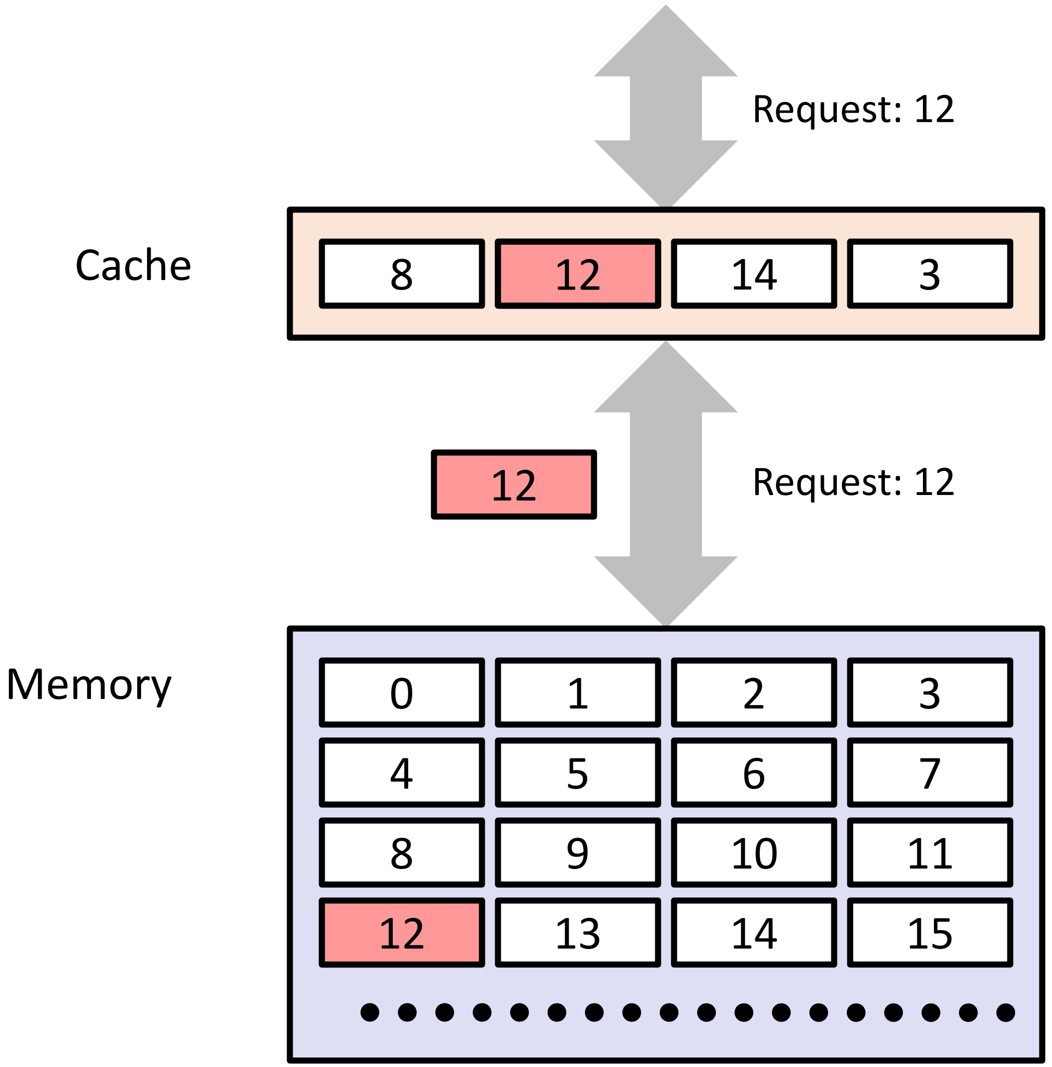
General Cache Concepts: Hit



Data in block b is needed

Block b is in cache:
Hit!

General Cache Concepts: Miss



Data in block b is needed

Block b is not in cache:
Miss!

Block b is fetched from *memory*

Block b is stored in cache
- **Placement policy:** determines where *b* goes
- **Replacement policy:** determines which block gets evicted (victim)

Types of Cache Misses

- **Cold (compulsory) miss**
  - Cold misses occur because the cache is empty
- **Conflict miss**
  - Most caches limit blocks at level *k+1* to a small subset (sometimes a singleton) of the block positions at level *k*
    - e.g. Block *i* at level *k+1* must be placed in block (*i* mod 4) at level *k*
  - Conflict misses occur when the level *k* cache is large enough, but multiple data objects all map to the same level *k* block
    - e.g. Referencing blocks 0, 8, 0, 8, 0, 8, … would miss every time
- **Capacity miss**
  - Occurs when the set of active cache blocks (working set) is larger than the cache

Caching Examples

Cache type	What is cached?	Where is it cached?	Latency (cycles)	Managed by
Registers	4-8 bytes words	CPU core	0	Compiler
TLB	Address translation	On-chip TLB	0	Hardware MMU
L1 cache	64-byte blocks	On-chip L1	4	Hardware
L2 cache	64-byte blocks	On-chip L2	10	Hardware
Virtual memory	4-KB pages	Main memory	100	Hardware + OS
Buffer cache	Parts of files	Main memory	100	OS
Disk cache	Disk sectors	Disk controller	100,000	Disk firmware
Network buffer cache	Parts of files	Local disk	10,000,000	NFS client
Browser cache	Web pages	Local disk	10,000,000	Web browser
Web cache	Web pages	Remote server disks	1,000,000,000	Web proxy server

Summary

- The speed gap between CPU, memory and mass storage continues to widen
- Well-written programs exhibit a property called locality
- Memory hierarchies based on caching close the gap by exploiting locality