

CS 33

Memory Hierarchy

Most of the slides in this lecture are either from or adapted from slides provided by the authors of the textbook “Computer Systems: A Programmer’s Perspective,” 2nd Edition and are provided from the website of Carnegie-Mellon University, course 15-213, taught by Randy Bryant and David O’Hallaron in Fall 2010. These slides are indicated “Supplied by CMU” in the notes section of the slides.

Today

- Storage technologies and trends
- Locality of reference
- Caching in the memory hierarchy

Supplied by CMU.

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
- Static RAM (SRAM)
 - each cell stores a bit with a four- or six-transistor circuit
 - retains value indefinitely, as long as it is kept powered
 - relatively insensitive to electrical noise (EMI), radiation, etc.
 - faster and more expensive than DRAM
- Dynamic RAM (DRAM)
 - each cell stores bit with a capacitor; transistor is used for access
 - value must be refreshed every 10-100 ms
 - more sensitive to disturbances (EMI, radiation,...) than SRAM
 - slower and cheaper than SRAM

Supplied by CMU.

SRAM vs DRAM Summary

	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

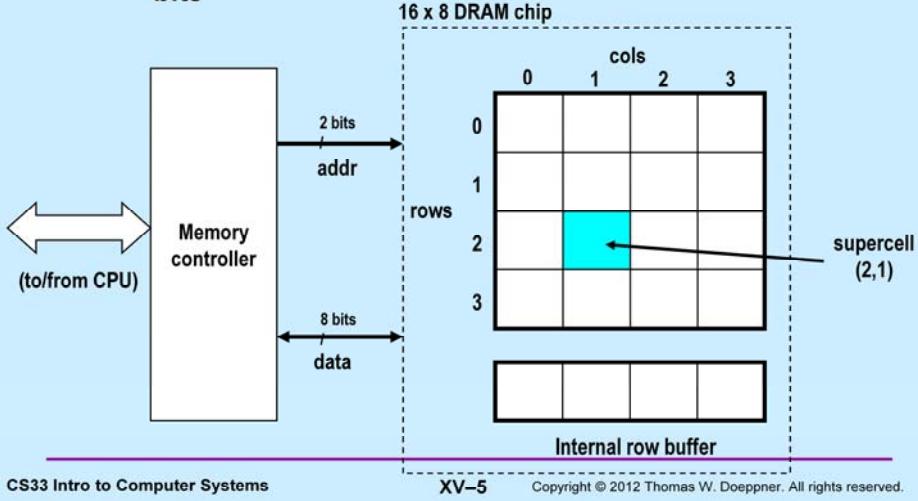
- **EDC = error detection and correction**
 - to cope with noise, etc.

Supplied by CMU.

Conventional DRAM Organization

- **d x w DRAM:**

- dw total bits organized as d **supercells** of size w bits

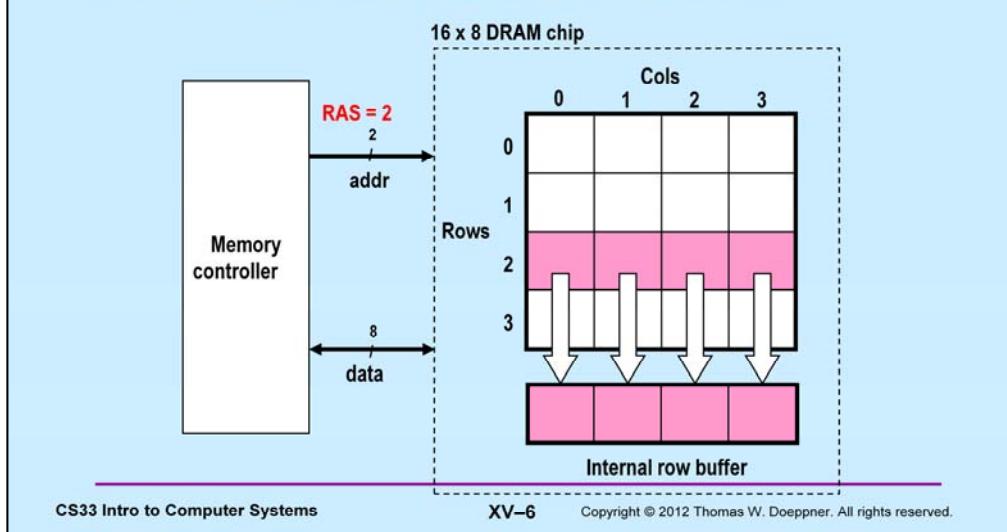


Supplied by CMU.

Reading DRAM Supercell (2,1)

Step 1(a): row access strobe (**RAS**) selects row 2

Step 1(b): row 2 copied from DRAM array to row buffer

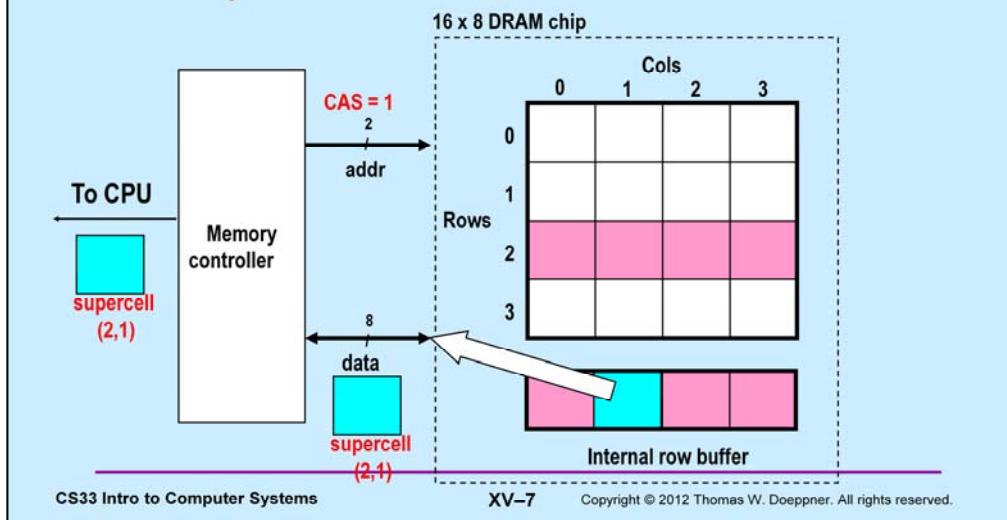


Supplied by CMU.

Reading DRAM Supercell (2,1)

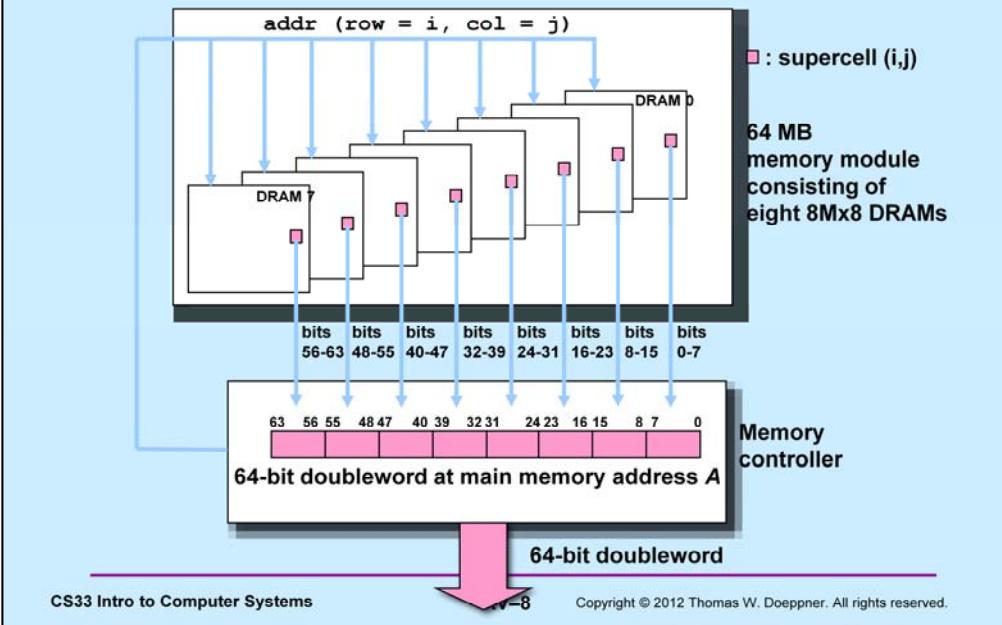
Step 2(a): column access strobe (**CAS**) selects column 1

Step 2(b): supercell (2,1) copied from buffer to data lines, and eventually back to the CPU



Supplied by CMU.

Memory Modules



Supplied by CMU.

Enhanced DRAMs

- Basic DRAM cell has not changed since its invention in 1966
 - commercialized by Intel in 1970
- DRAM cores with better interface logic and faster I/O:
 - synchronous DRAM (**SDRAM**)
 - » uses a conventional clock signal instead of asynchronous control
 - » allows reuse of the row addresses (e.g., RAS, CAS, CAS, CAS)
 - double data-rate synchronous DRAM (**DDR SDRAM**)
 - » double edge clocking sends two bits per cycle per pin
 - » different types distinguished by size of small prefetch buffer:
 - **DDR** (2 bits), **DDR2** (4 bits), **DDR3** (8 bits)
 - » by 2010, standard for most server and desktop systems
 - » Intel Core i7 supports only DDR3 SDRAM

Supplied by CMU.

The “prefetch buffer” hold cells from adjacent rows when a row is fetched. The idea is that, in most cases, if the processor requests one row, it will need the next row soon.

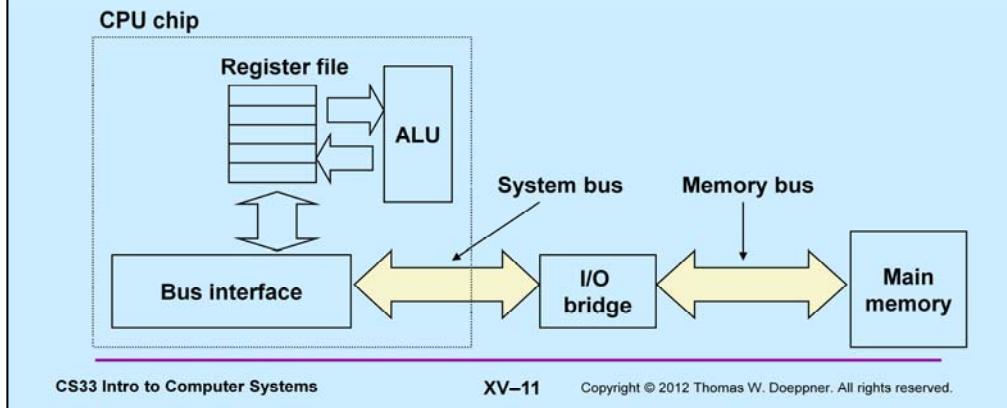
Nonvolatile Memories

- 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 memory: EEPROMs with partial (sector) erase capability
 - » wears out after about 100,000 erasings
- Uses for nonvolatile memories
 - firmware programs stored in a ROM (BIOS, controllers for disks, network cards, graphics accelerators, security subsystems,...)
 - solid state disks (replace rotating disks in thumb drives, smart phones, mp3 players, tablets, laptops,...)
 - disk caches

Supplied by CMU.

Traditional Bus Structure Connecting CPU and Memory

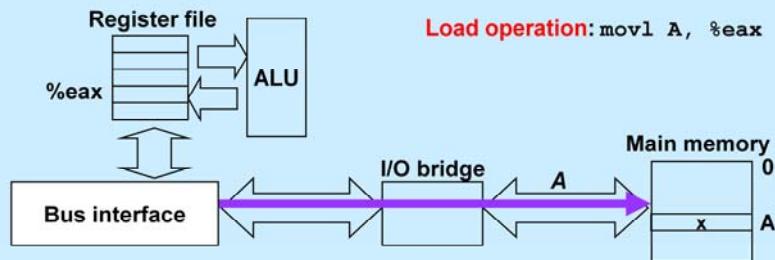
- A **bus** is a collection of parallel wires that carry address, data, and control signals
- Buses are typically shared by multiple devices



Supplied by CMU.

Memory Read Transaction (1)

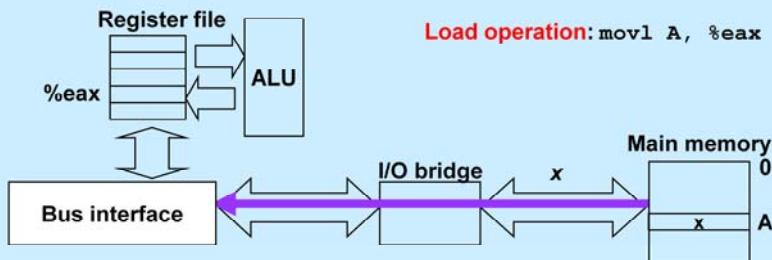
- CPU places address A on the memory bus



Supplied by CMU.

Memory Read Transaction (2)

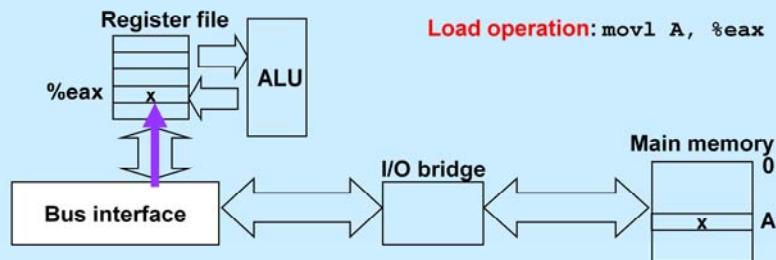
- Main memory reads A from the memory bus, retrieves word x, and places it on the bus



Supplied by CMU.

Memory Read Transaction (3)

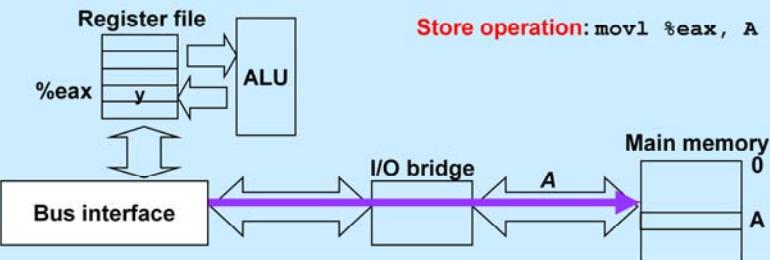
- CPU reads word x from the bus and copies it into register $\%eax$



Supplied by CMU.

Memory Write Transaction (1)

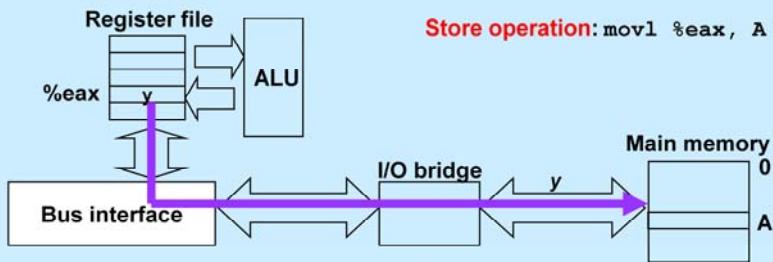
- CPU places address A on bus. Main memory reads it and waits for the corresponding data word to arrive



Supplied by CMU.

Memory Write Transaction (2)

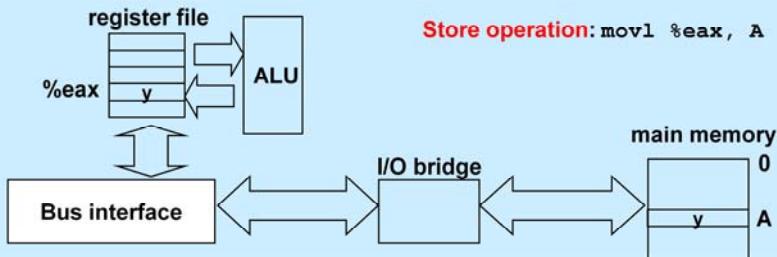
- CPU places data word y on the bus



Supplied by CMU.

Memory Write Transaction (3)

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



Supplied by CMU.

What's Inside A Disk Drive?

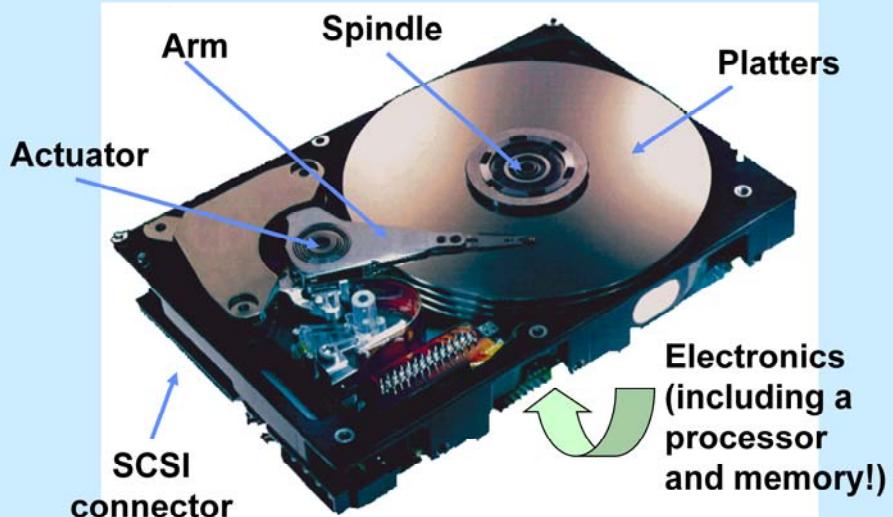
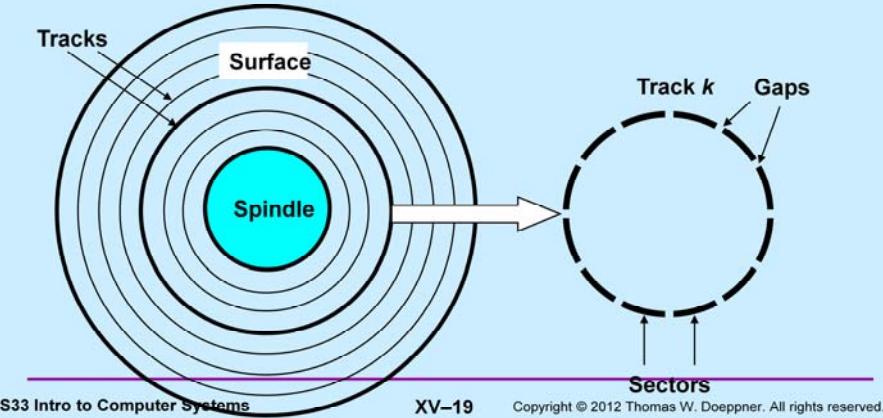


Image courtesy of Seagate Technology

Supplied by CMU.

Disk Geometry

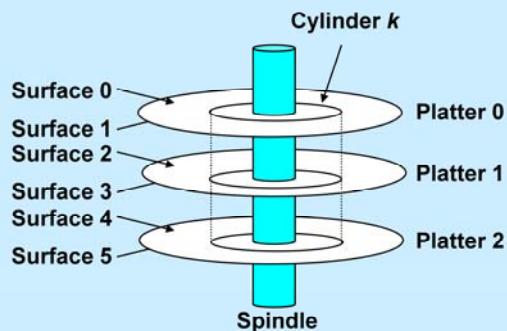
- Disks consist of **platters**, each with two **surfaces**
- Each surface consists of concentric rings called **tracks**
- Each track consists of **sectors** separated by **gaps**



Supplied by CMU.

Disk Geometry (Multiple-Platter View)

- Aligned tracks form a cylinder



Supplied by CMU.

Disk Capacity

- **Capacity:** maximum number of bits that can be stored
 - capacity expressed in units of gigabytes (GB), where $1 \text{ GB} = 10^9 \text{ Bytes}$
- Capacity is determined by these technology factors:
 - **recording density** (bits/in): number of bits that can be squeezed into a 1 inch segment of a track
 - **track density** (tracks/in): number of tracks that can be squeezed into a 1 inch radial segment
 - **areal density** (bits/in²): product of recording and track density
- Modern disks partition tracks into disjoint subsets called **recording zones**
 - each track in a zone has the same number of sectors, determined by the circumference of innermost track
 - each zone has a different number of sectors/track

Supplied by CMU.

Computing Disk Capacity

**Capacity = (# bytes/sector) x (avg. # sectors/track) x
(# tracks/surface) x (# surfaces/platter) x
(# platters/disk)**

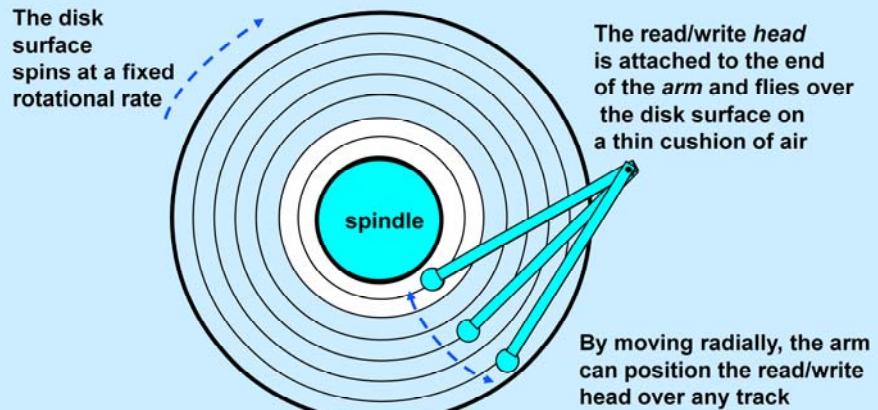
Example:

- 512 bytes/sector
- 600 sectors/track (on average)
- 40,000 tracks/surface
- 2 surfaces/platter
- 5 platters/disk

$$\begin{aligned}\text{Capacity} &= 512 \times 600 \times 40000 \times 2 \times 5 \\ &= 122,280,000,000 \\ &= 122.28 \text{ GB}\end{aligned}$$

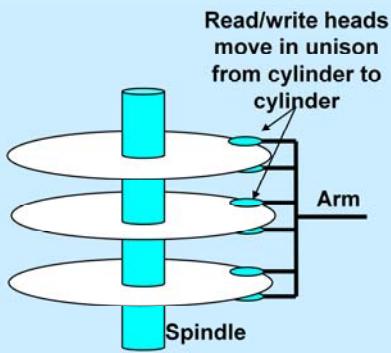
Supplied by CMU.

Disk Operation (Single-Platter View)



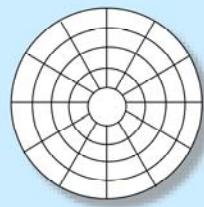
Supplied by CMU.

Disk Operation (Multi-Platter View)



Supplied by CMU.

Disk Structure: Top View of Single Platter

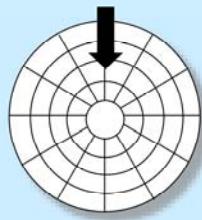


Surface organized into tracks

Tracks divided into sectors

Supplied by CMU.

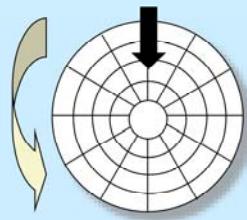
Disk Access



Head in position above a track

Supplied by CMU.

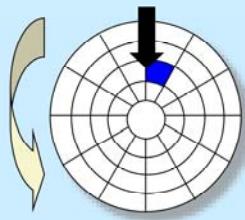
Disk Access



Rotation is counter-clockwise

Supplied by CMU.

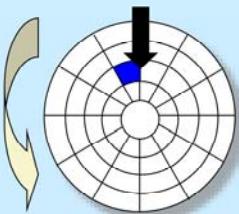
Disk Access – Read



About to read blue sector

Supplied by CMU.

Disk Access – Read

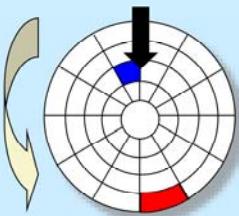


After **BLUE**
read

After reading blue sector

Supplied by CMU.

Disk Access – Read

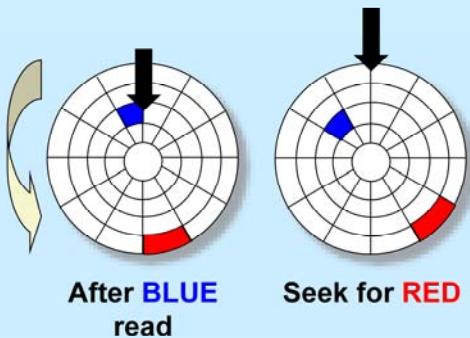


After **BLUE**
read

Red request scheduled next

Supplied by CMU.

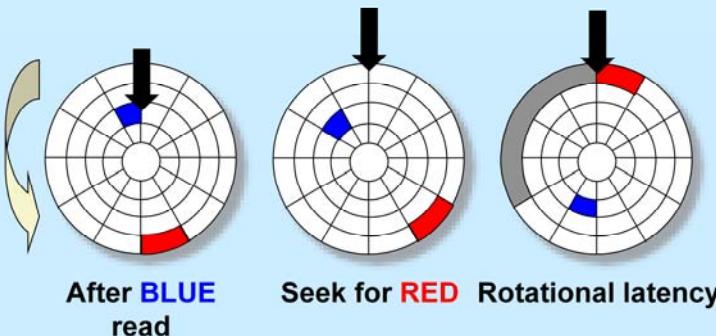
Disk Access – Seek



Seek to red's track

Supplied by CMU.

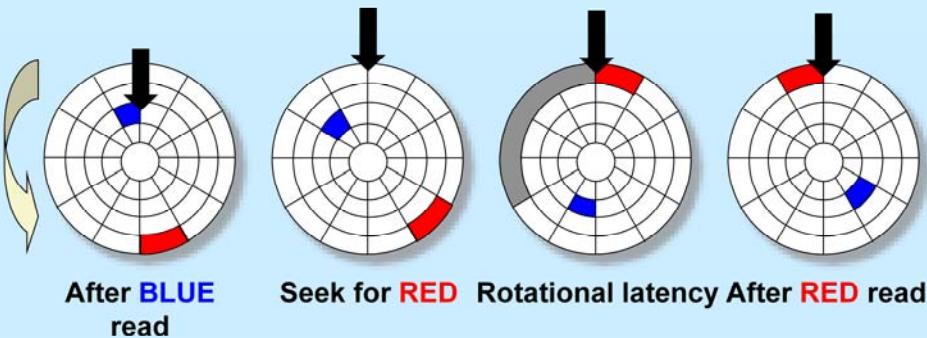
Disk Access – Rotational Latency



Wait for red sector to rotate around

Supplied by CMU.

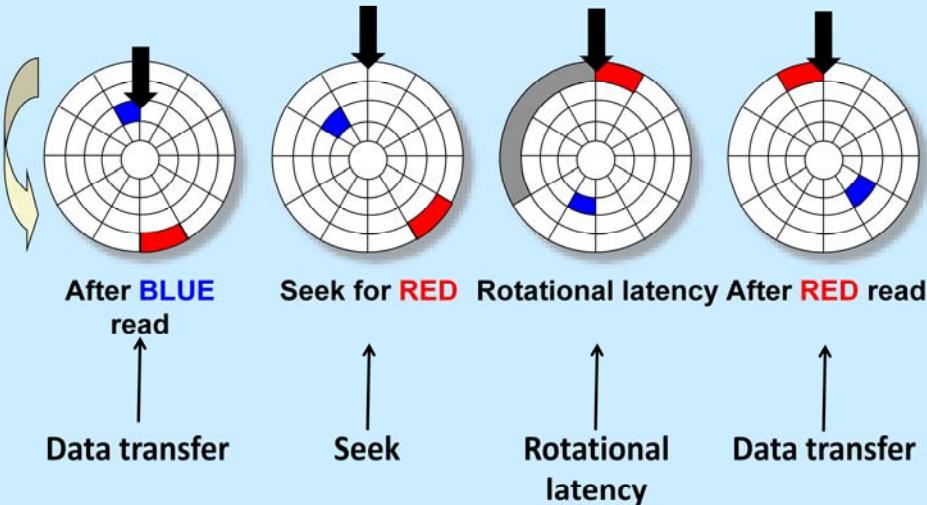
Disk Access – Read



Complete read of red

Supplied by CMU.

Disk Access – Service Time Components



Supplied by CMU.

Disk Access Time

- Average time to access some target sector approximated by :
 - $T_{access} = T_{avg\ seek} + T_{avg\ rotation} + T_{avg\ transfer}$
- Seek time (Tavg seek)
 - time to position heads over cylinder containing target sector
 - typical Tavg seek is 3–9 ms
- Rotational latency (Tavg rotation)
 - time waiting for first bit of target sector to pass under r/w head
 - $T_{avg\ rotation} = 1/2 \times 1/\text{RPMs} \times 60\ \text{sec}/1\ \text{min}$
 - typical Tavg rotation = 7200 RPMs
- Transfer time (Tavg transfer)
 - time to read the bits in the target sector
 - $T_{avg\ transfer} = 1/\text{RPM} \times 1/(\text{avg\ # sectors/track}) \times 60\ \text{secs}/1\ \text{min}$

Supplied by CMU.

Disk Access Time Example

- Given:
 - rotational rate = 7,200 RPM
 - average seek time = 9 ms
 - avg # sectors/track = 400
- Derived:
 - $T_{avg\ rotation} = 1/2 \times (60\ secs/7200\ RPM) \times 1000\ ms/sec = 4\ ms$
 - $T_{avg\ transfer} = 60/7200\ RPM \times 1/400\ secs/track \times 1000\ ms/sec = 0.02\ ms$
 - $T_{access} = 9\ ms + 4\ ms + 0.02\ ms$
- Important points:
 - access time dominated by seek time and rotational latency
 - first bit in a sector is the most expensive, the rest are free
 - SRAM access time is about 4 ns/doubleword, DRAM about 60 ns
 - » disk is about 40,000 times slower than SRAM
 - » 2,500 times slower than DRAM

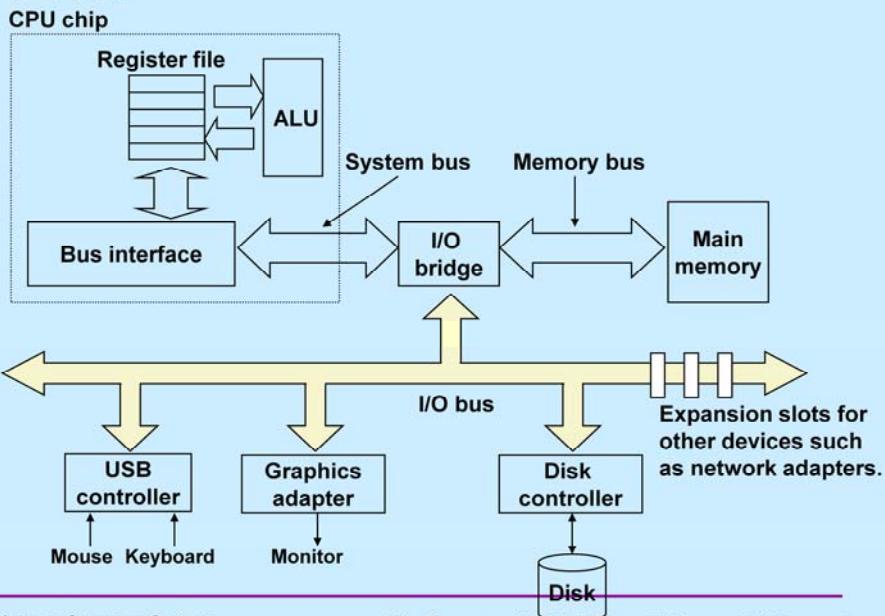
Supplied by CMU.

Logical Disk Blocks

- Modern disks present a simpler abstract view of the complex sector geometry:
 - the set of available sectors is modeled as a sequence of b-sized **logical blocks** (0, 1, 2, ...)
- Mapping between logical blocks and actual (physical) sectors
 - maintained by hardware/firmware device called disk controller
 - converts requests for logical blocks into (surface,track,sector) triples
- Allows controller to set aside spare cylinders for each zone
 - accounts for the difference in “formatted capacity” and “maximum capacity”

Supplied by CMU.

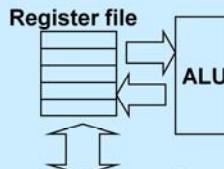
I/O Bus



Supplied by CMU.

Reading a Disk Sector (1)

CPU chip



CPU initiates a disk read by writing a command, logical block number, and destination memory address to a **port** (address) associated with disk controller

Bus interface

Main memory

I/O bus

USB controller

Graphics adapter

Disk controller

Mouse Keyboard

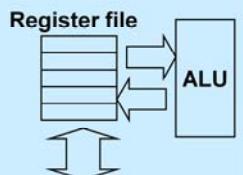
Monitor



Supplied by CMU.

Reading a Disk Sector (2)

CPU chip



Disk controller reads the sector and performs a direct memory access (DMA) transfer into main memory

Bus interface

Main memory

I/O bus

USB controller
Mouse Keyboard

Graphics adapter
Monitor

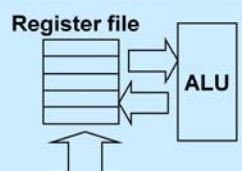
Disk controller

Disk

Supplied by CMU.

Reading a Disk Sector (3)

CPU chip



When the DMA transfer completes, the disk controller notifies the CPU with an *interrupt* (i.e., asserts a special "interrupt" pin on the CPU)

Bus interface

Main memory

I/O bus

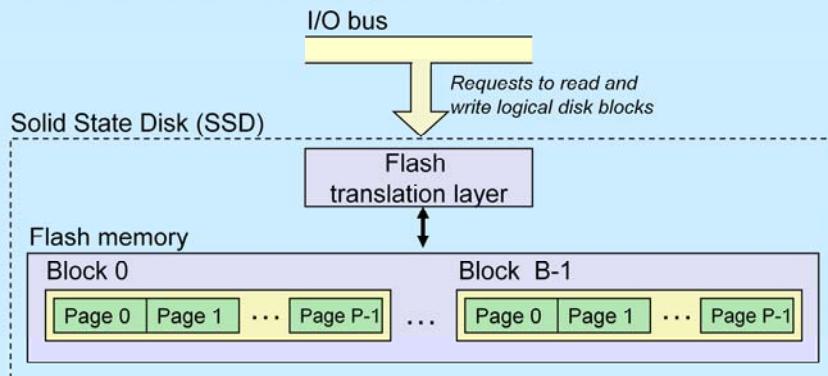
USB controller
Mouse Keyboard

Graphics adapter
Monitor

Disk controller
Disk

Supplied by CMU.

Solid State Disks (SSDs)



- **Pages: 512KB to 4KB; blocks: 32 to 128 pages**
- **Data read/written in units of pages**
- **Page can be written only after its block has been erased**
- **A block wears out after 100,000 repeated writes**

Supplied by CMU.

SSD Performance Characteristics

Sequential read tput	250 MB/s	Sequential write tput	170 MB/s
Random read tput	140 MB/s	Random write tput	14 MB/s
Random read access	30 us	Random write access	300 us

- Why are random writes so slow?
 - erasing a block is slow (around 1 ms)
 - write to a page triggers a copy of all useful pages in the block
 - » find a used block (new block) and erase it
 - » write the page into the new block
 - » copy other pages from old block to the new block

Supplied by CMU.

SSD Tradeoffs vs Rotating Disks

- **Advantages**
 - no moving parts → faster, less power, more rugged
- **Disadvantages**
 - have the potential to wear out
 - » mitigated by “wear-leveling logic” in flash translation layer
 - » e.g. Intel X25 guarantees 1 petabyte (10^{15} bytes) of random writes before they wear out
 - in 2010, about 100 times more expensive per byte
- **Applications**
 - MP3 players, smart phones, laptops
 - beginning to appear in desktops and servers

Supplied by CMU.

Storage Trends

SRAM

Metric	1980	1985	1990	1995	2000	2005	2010	2010:1980
\$/MB	19,200	2,900	320	256	100	75	60	320
access (ns)	300	150	35	15	3	2	1.5	200

DRAM

Metric	1980	1985	1990	1995	2000	2005	2010	2010:1980
\$/MB	8,000	880	100	30	1	0.1	0.06	130,000
access (ns)	375	200	100	70	60	50	40	9
typical size (MB)	0.064	0.256	4	16	64	2,000	8,000	125,000

Disk

Metric	1980	1985	1990	1995	2000	2005	2010	2010:1980
\$/MB	500	100	8	0.30	0.01	0.005	0.0003	1,600,000
access (ms)	87	75	28	10	8	4	3	29
typical size (MB)	1	10	160	1,000	20,000	160,000	1,500,000	1,500,000

Supplied by CMU.

CPU Clock Rates

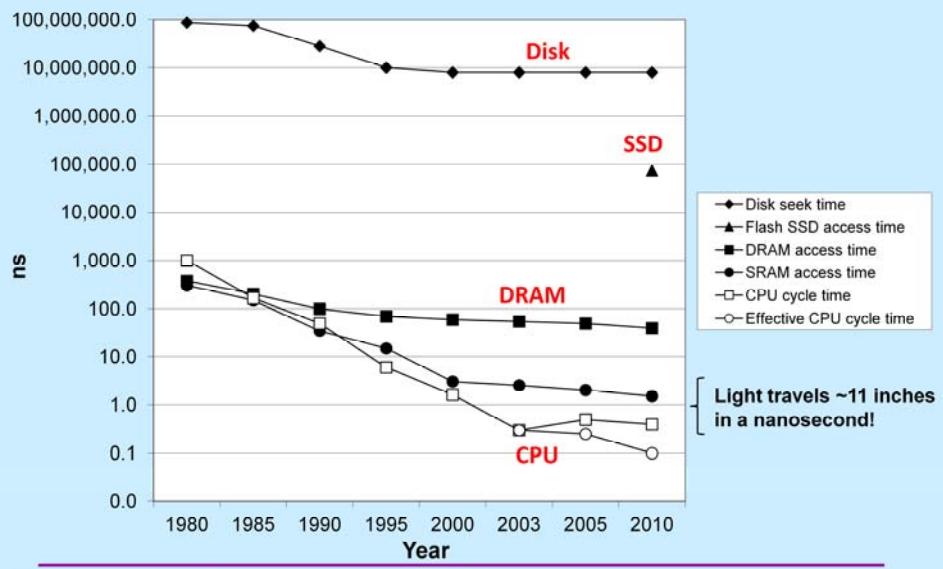
Inflection point in computer history
when designers hit the “Power Wall”

	1980	1990	1995	2000	2003	2005	2010	2010:1980
CPU	8080	386	Pentium	P-III	P-4	Core 2	Core i7	---
Clock rate (MHz)	1	20	150	600	3300	2000	2500	2500
Cycle time (ns)	1000	50	6	1.6	0.3	0.50	0.4	2500
Cores	1	1	1	1	1	2	4	4
Effective cycle time (ns)	1000	50	6	1.6	0.3	0.25	0.1	10,000

Supplied by CMU.

The CPU-Memory Gap

The gap widens between DRAM, disk, and CPU speeds



Supplied by CMU.

Locality to the Rescue!

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

Supplied by CMU.

Today

- Storage technologies and trends
- Locality of reference
- Caching in the memory hierarchy

Supplied by CMU.

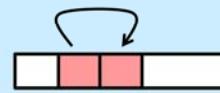
Locality

- **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

Supplied by CMU.

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**

Supplied by CMU.

Qualitative Estimates of Locality

- **Claim:** being able to look at code and get a qualitative sense of its locality is a key skill for a professional programmer
- **Question:** does this function have 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;
}
```

Supplied by CMU.

Locality Example

- **Question:** does this function have 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;
}
```

Supplied by CMU.

Locality Example

- **Question:** 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)?

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

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

Supplied by CMU.

Memory Hierarchies

- 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 as a **memory hierarchy**

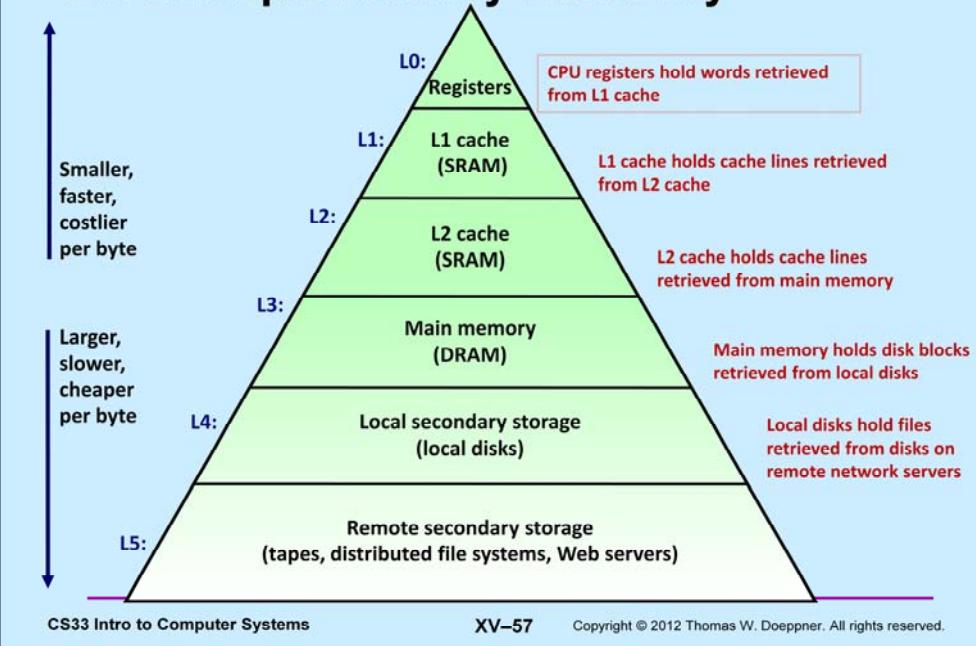
Supplied by CMU.

Today

- Storage technologies and trends
- Locality of reference
- **Caching in the memory hierarchy**

Supplied by CMU.

An Example Memory Hierarchy



Supplied by CMU.

Caches

- **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
- **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

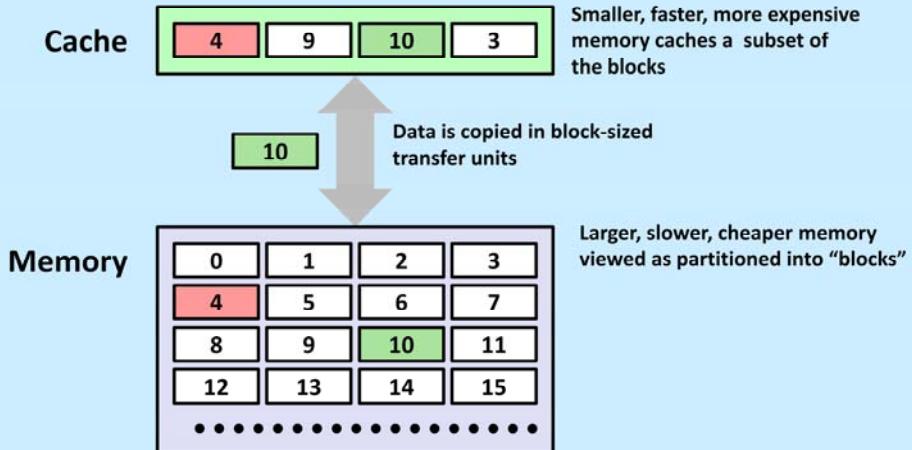
Supplied by CMU.

Putting Things Into Perspective ...

- **Reading from:**
 - ... the L1 cache is like grabbing a piece of paper from your desk (3 seconds)
 - ... the L2 cache is picking up a book from a nearby shelf (14 seconds)
 - ... main system memory is taking a 4-minute walk down the hall to talk to an officemate
 - ... a hard drive is like leaving the building to roam the earth for one year and three months

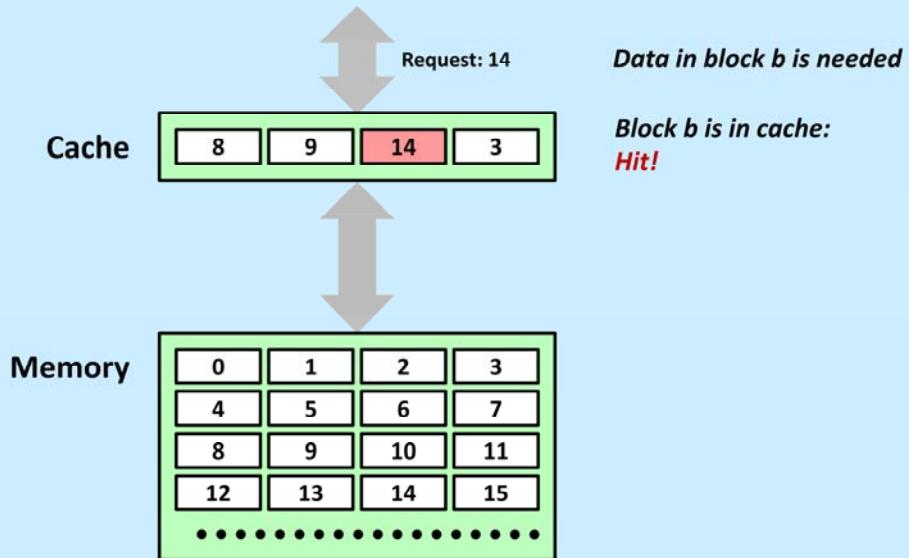
This analogy is from <http://duartes.org/gustavo/blog/post/what-your-computer-does-while-you-wait> (definitely worth reading!).

General Cache Concepts



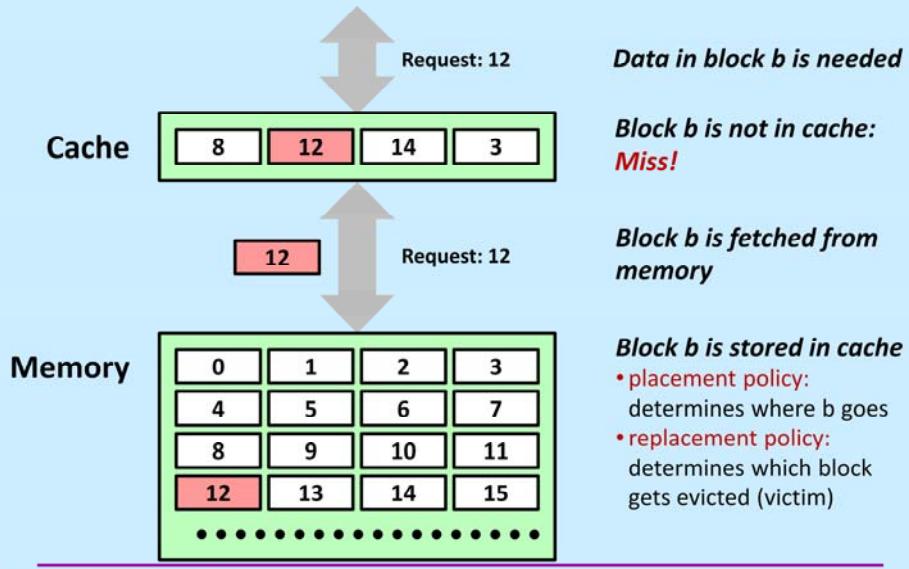
Supplied by CMU.

General Cache Concepts: Hit



Supplied by CMU.

General Cache Concepts: Miss



Supplied by CMU.

General Caching Concepts: 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 $(i \bmod 4)$ 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

Supplied by CMU.

Examples of Caching in the Hierarchy

Cache Type	What is Cached?	Where is it Cached?	Latency (cycles)	Managed By
Registers	4-8 bytes words	CPU core	0	Compiler
TLB	Address translations	On-Chip TLB	0	Hardware
L1 cache	64-bytes block	On-Chip L1	1	Hardware
L2 cache	64-bytes block	On/Off-Chip L2	10	Hardware
Virtual Memory	4-KB page	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	CIFS/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

Supplied by CMU.

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

Supplied by CMU.