

# The Memory Hierarchy

CPSC 235 - Computer Organization

# References

- Slides adapted from CMU

# Outline

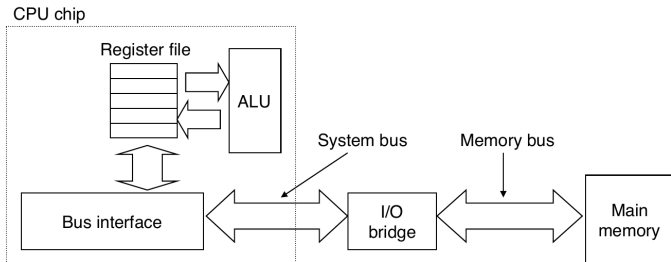
- The memory abstraction
- RAM: main memory
- Locality of reference
- The memory hierarchy
- Storage technologies and trends

# Writing and Reading Memory

- Write
  - Transfer data from CPU to memory
  - Example: `movq %rax, 8(%rsp)`
  - “Store” operation
- Read
  - Transfer data from memory to CPU
  - Example: `movq 8(%rsp), %rax`
  - “Load” operation

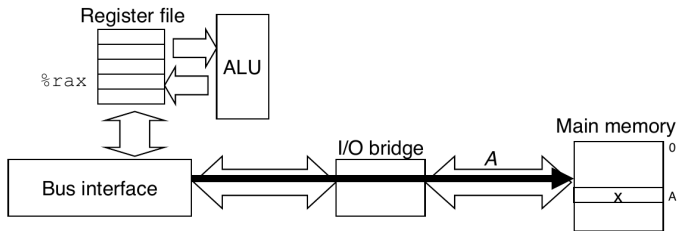
# Traditional Bus Structure Connecting CPU and Memory

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



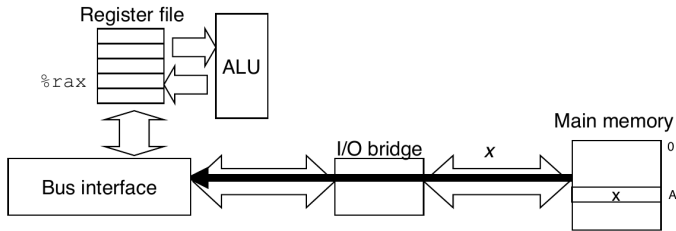
# Memory Read Transaction (1)

- Example: `movq A, %rax`
- CPU places address `A` on the memory bus



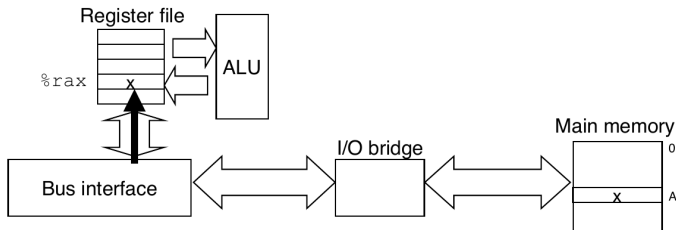
# Memory Read Transaction (2)

- Example: `movq A, %rax`
- Main memory reads A from the memory bus, retrieves word x, and places it on the bus



# Memory Read Transaction (3)

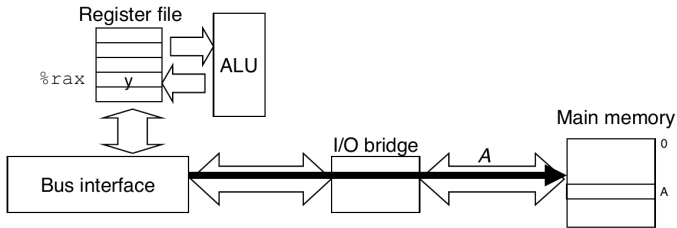
- Example: `movq A, %rax`
- CPU read word `x` from the bus and copies it into register `%rax`





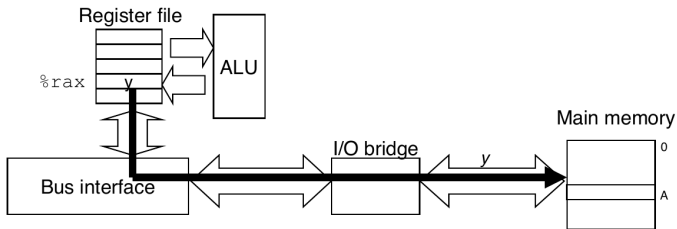
# Memory Write Transaction (1)

- Example: `movq %rax, A`
- CPU places address `A` on the memory bus; main memory reads it and waits for the corresponding data word to arrive



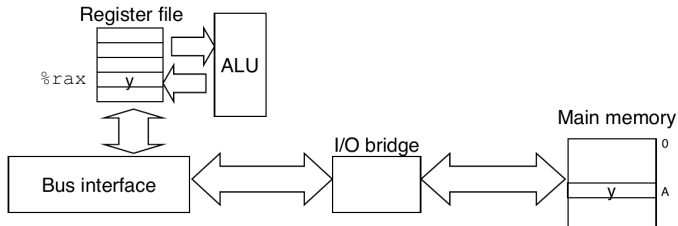
# Memory Write Transaction (2)

- Example: `movq %rax, A`
- CPU places data word  $y$  on the bus



# Memory Write Transaction (2)

- Example: `movq %rax, A`
- Main memory reads data word `y` from the bus and stores it at address `A`



# Random-Access Memory (RAM)

- Key features
  - RAM is traditionally packaged as a chip or embedded as part of processor chip
  - Basic storage unit is normally a cell (one bit per cell)
  - Multiple RAM chips form a memory
- RAM comes in two varieties
  - SRAM (static RAM)
  - DRAM (Dynamic RAM)

# RAM Technologies

- DRAM
  - 1 transistor + 1 capacitor per bit
  - Must refresh state periodically
- SRAM
  - 6 transistors per bit
  - Holds state indefinitely (but will still lose data on power loss)
- Trends
  - SRAM scales with semiconductor technology
  - DRAM scaling limited by need to minimum capacitance

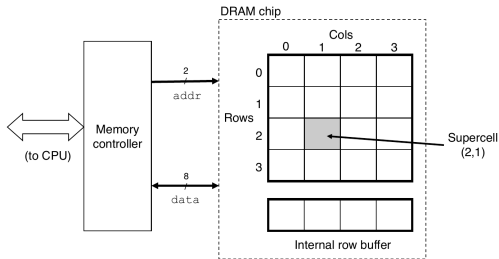
# Enhanced DRAMs

- Operation of DRAM cell has not changed since its invention
  - 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
  - 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), DDR4 (16 bits)

# Conventional DRAM Organization

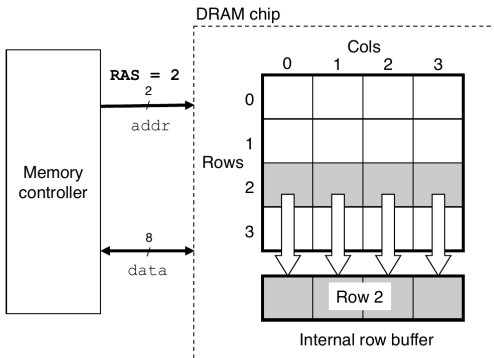
- $d \times w$  DRAM

- $d \cdot w$  total bits organized as  $d$  supercells of size  $w$  bits



# Reading DRAM Supercell (2,1)

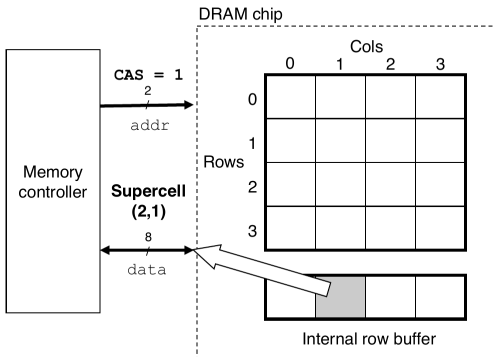
- Step 1(a): Row access strobe (RAS) selects row 2
- Step 1(b): Row 2 copied from DRAM array to row buffer



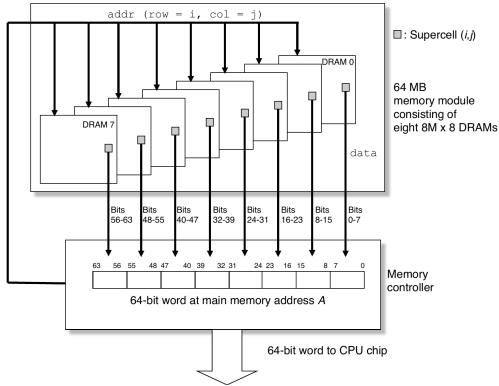


# Reading DRAM Supercell (2,1)

- 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
- Step 3: All data written back to row to provide refresh

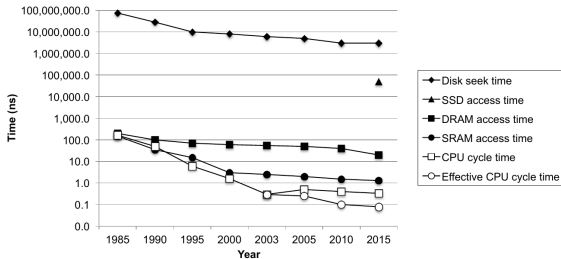


# Memory Modules



# The CPU-Memory Gap

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



# 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
  - References array elements in succession (spatial)
  - Reference variable `sum` each iteration (temporal)
- Instruction references
  - Reference instructions in sequence (spatial)
  - Cycle through loop repeatedly (temporal)

# Qualitative Estimates of Locality

- Claim: being able to look at code and get a qualitative sense of its locality is a good 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;
}
```

# Locality Example

- Question: can you permute the loops so that the function scans the 3D array with a stride-1 reference pattern (and thus have good spatial locality)?

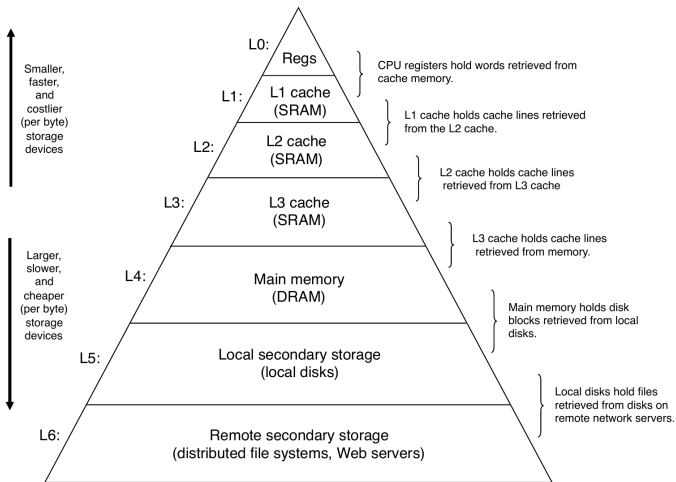
```
int sum_array_3d(int a[M][N][N]) {
    int i, j, k, sum = 0;
    for (i = 0; i < M; i++) {
        for (j = 0; j < N; j++) {
            for (k = 0; k < M; k++) {
                sum += a[k][i][j];
            }
        }
    }
    return sum;
}
```

# 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



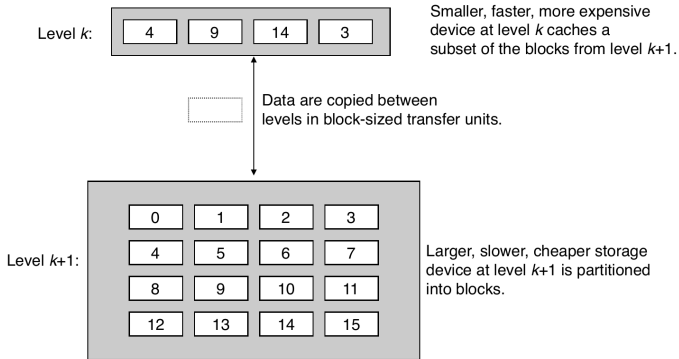
# Example Memory Hierarchy



# 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 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, larger, and cheaper per bit
- Big Idea (Ideal): The memory hierarchy creates a large pool of storage that costs as much as the cheap storage near the bottom, but serves data to programs at the rate of the fast storage near the top

# General Cache Concepts



# General Cache Concepts

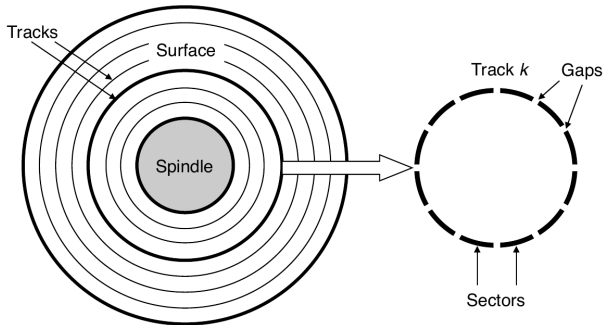
- A cache hit is when the data in block  $b$  is needed and is in the cache
- A cache miss is when the data in block  $b$  is needed and is in not the cache
- Types of cache misses:
  - Cold (compulsory) miss: occur because the cache starts empty and this is the first reference to the block
  - Capacity miss: occur when the set of active cache blocks (working set) is larger than the cache
  - Conflict miss: occur when the level  $k$  cache is large enough, but multiple data objects all map to the same level  $k$  block where a block is a small subset of the block positions at level  $k - 1$

# Storage Technologies

- Magnetic disks
  - Store on magnetic medium
  - Electromechanical access
- Nonvolatile (Flash) memory
  - Store as persistent charge
  - Implemented with 3D structure

# 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

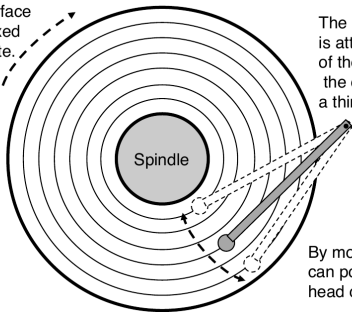


# Disk Capacity

- Capacity: maximum number of bits that can be stored
  - Vendors express capacity in units of gigabytes (GB) or terabytes (TB), where  $1 \text{ GB} = 10^9 \text{ Bytes}$  and  $1 \text{ TB} = 10^{12} \text{ Bytes}$
- Capacity is determined by these technology factors:
  - Recording density (bits/in): number of bits that can be squeezed into a 1 inch segment
  - Track density (tracks/in): number of tracks that can be squeezed into a 1 inch radial segment
  - Areal density (bits/in<sup>2</sup>): product of recording density and track density

# Disk Operation

The disk surface spins at a fixed rotational rate.

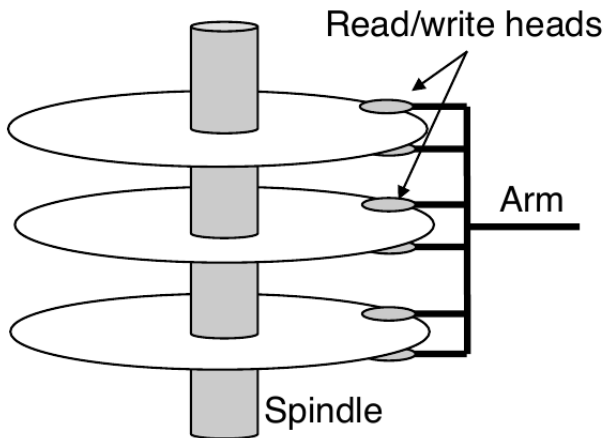


The read/write head is attached to the end of the arm and flies over the disk surface on a thin cushion of air.

By moving radially, the arm can position the read/write head over any track.



# Disk Operation



# Disk Access Time

- Average time to access some target sector approximated by:

- $T_{access} = T_{seek} + T_{rotation} + T_{transfer}$

- Seek time

- Time to position heads over cylinder containing target sector
  - Typical  $T_{seek}$  is 3 to 9 ms

- Rotational latency

- Time waiting for the first bit of target sector to pass under read/write head
  - $T_{rotation} = \frac{1}{2} \cdot \frac{1}{RPMs} \cdot \frac{60 s}{1 min}$
  - Typical rotational rate is 7,200 RPMs

- Transfer time

- Time to read the bits in the target sector
  - $T_{transfer} = \frac{1}{RPM} \cdot \frac{1}{avg\ sectors\ per\ track} \cdot \frac{60 s}{1 min}$

# Disk Access Time Example

## ■ Given

- Rotational rate = 7200 RPM
- Average seek time = 9 ms
- Average number of sectors per track = 400

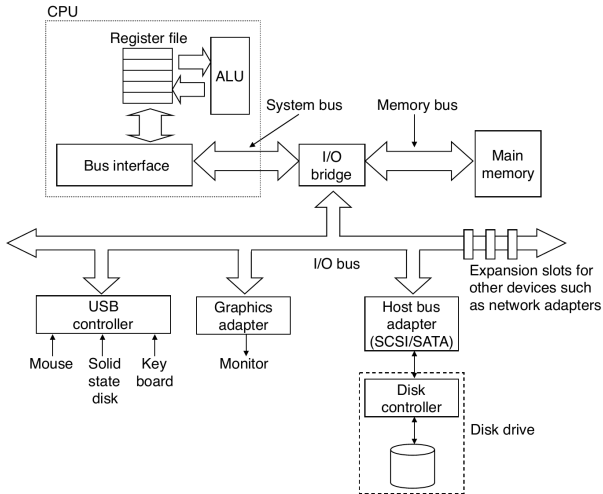
## ■ Derived:

- $T_{rotation} = 4 \text{ ms}$
- $T_{transfer} = 0.02 \text{ ms}$
- $T_{access} = 0.02 \text{ ms}$

## ■ Important points:

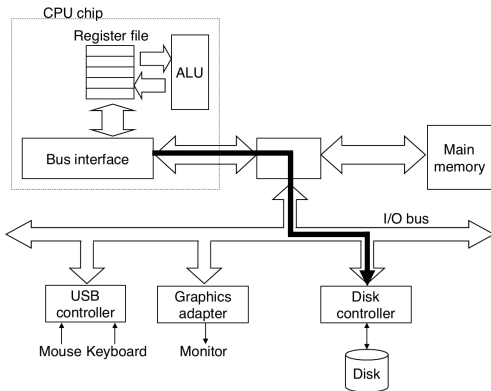
- Access time is dominated by seek time and rotational latency
- First bit in sector is the most expensive, the rest are free
- SRAM access time is about 4 ns per double word, DRAM about 60 ns
  - Disk is about 40,000 times slower than SRAM
  - Disk is about 2,500 times slower than DRAM

# I/O Bus



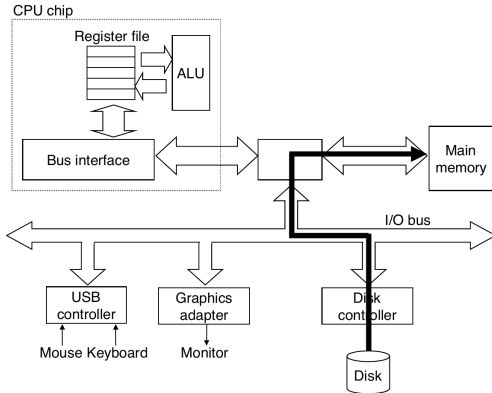
# Reading a Disk Sector (1)

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



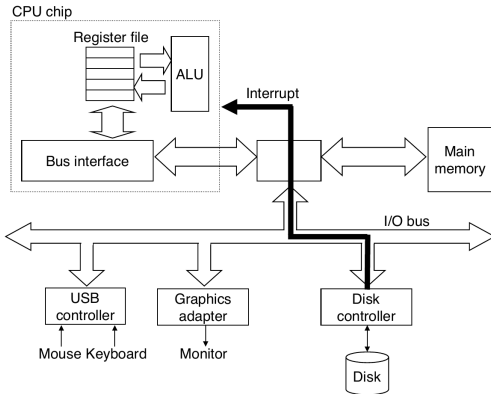
# Reading a Disk Sector (2)

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



# Reading a Disk Sector (3)

- When the DMA transfer completes, the disk controller notifies the CPU with an interrupt

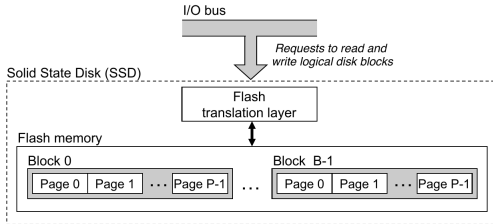


# Nonvolatile Memories

- 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
  - Electrically erasable PROM (EEPROM): electronic erase capability
  - Flash memory: EEPROMS with partial (block level) erase capability
- Uses for Nonvolatile Memories
  - Firmware programs stored in a ROM
  - Solid state disks
  - Disk caches



# 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 about 100,000 repeated writes

# SSD Tradeoffs versus Rotating Disks

- Advantages
  - No moving parts
- Disadvantages
  - Have the potential to wear out
  - More expensive per byte
- Applications
  - Smartphones, laptops
  - Increasingly common in desktops and servers

# 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
- Flash memory progress outpacing all other memory and storage technologies