Transition

• First half of course: hardware focus
  – How the hardware is constructed
  – How the hardware works
  – How to interact with hardware / ISA

• Up next: performance and software systems
  – Memory performance
  – Operating systems
  – Standard libraries (strings, threads, etc.)
• How to **Efficiently** Run Programs

• Good algorithm is critical...

• Many systems concerns to account for too!
  – The memory hierarchy and its effect on program performance
  – OS abstractions for running programs efficiently
  – Support for parallel programming
Efficiency

• How to Efficiently Run Programs

• Good algorithm is critical...

• Many systems concerns to account for too!
  – The memory hierarchy and its effect on program performance
  – OS abstractions for running programs efficiently
  – Support for parallel programming
Suppose you’re designing a new computer architecture. Which type of memory would you use? Why?

A. low-capacity (~1 MB), fast, expensive

B. medium-capacity (a few GB), medium-speed, moderate cost

C. high-capacity (100’s of GB), slow, cheap

D. something else (it must exist)

trade-off between capacity and speed
Classifying Memory

• Broadly, two types of memory:
  1. Primary storage: CPU instructions can access any location at any time (assuming OS permission)
  2. Secondary storage: CPU can’t access this directly
Random Access Memory (RAM)

• Any location can be accessed directly by CPU
  – Volatile Storage: lose power → lose contents

• Static RAM (SRAM)
  – Latch-Based Memory (e.g. RS latch), 1 bit per latch
  – Faster and more expensive than DRAM
    • “On chip”: Registers, Caches

• Dynamic RAM (DRAM)
  – Capacitor-Based Memory, 1 bit per capacitor
    • “Main memory”: Not part of CPU
Memory Technologies

• Static RAM (SRAM)
  – 0.5ns – 2.5ns, $2000 – $5000 per GB

• Dynamic RAM (DRAM)
  – 50ns – 100ns, $20 – $75 per GB
  (Main memory, “RAM”)

We’ve talked a lot about registers (SRAM) and we’ll cover caches (SRAM) soon. Let’s look at main memory (DRAM) now.
Dynamic Random Access Memory (DRAM)

Capacitor based:
- cheaper and slower than SRAM
- capacitors are leaky (lose charge over time)
- **Dynamic**: value needs to be refreshed (every 10-100ms)

Example: DIMM
(Dual In-line Memory Module):
Connecting CPU and Memory

• Components are connected by a **bus**:
  • A bus is a collection of parallel wires that carry address, data, and control signals.
  • Buses are typically shared by multiple devices.
How A Memory Read Works

(1) CPU places address A on the memory bus.

Load operation: \texttt{mov (Address A), \%rax}
(2) Main Memory reads address A from memory, fetches value at that address and puts it on the bus.
How A Memory Read Works

(3) CPU reads value from the bus, and copies it into register rax. A copy also goes into the on-chip cache memory.
How a Memory Write Works

1. CPU writes A to bus, memory reads it
2. CPU writes value to bus, memory reads it
3. Memory stores value at address A
Secondary Storage

- Disk, Tape Drives, Flash Solid State Drives, ...

- Non-volatile: retains data without a charge

- Instructions **CANNOT** directly access data on secondary storage
  - No way to specify a disk location in an instruction
  - Operating System moves data to/from memory
Secondary Storage

CPU
- Register
- Register
- Register
- ALU
- CPU Cache

Memory Module Slots

Memory Bus

I/O Bus (e.g., PCI)
- SATA Controller
- USB Controller
- IDE Controller
- ... (Secondary Storage Devices)

I/O Controller

(path is much longer)
What’s Inside A Disk Drive?

- Spindle
- Arm
- Actuator
- Platters
- Controller Electronics (includes processor & memory)
- bus connector
- R/W head

Data Encoded as points of magnetism on Platter surfaces

Device Driver (part of OS code) interacts with Controller to R/W to disk

Image from Seagate Technology
Reading and Writing to Disk

Data blocks located in some **Sector** of some **Track** on some **Surface**

1. Disk Arm moves to correct **track** (seek time)
2. Wait for **sector** spins under R/W head (rotational latency)
3. As sector spins under head, data are Read or Written (transfer time)
Memory Technology

• Static RAM (SRAM)
  – 0.5ns – 2.5ns, $2000 – $5000 per GB

• Dynamic RAM (DRAM)
  – 50ns – 100ns, $20 – $75 per GB

  Solid-state disks (flash): 100 us – 1 ms, $2 - $10 per GB

• Magnetic disk
  – 5ms – 15ms, $0.20 – $2 per GB

Like walking:
  Down the hall

Across campus
  (to Cleveland / Indianapolis)

To Seattle

1 ms == 1,000,000 ns
The Memory Hierarchy

- **Local secondary storage (disk)**:
  - Larger
  - Slower
  - Cheaper per byte
  - ~100 M cycles to access

- **Main memory (DRAM)**:
  - Smaller
  - Faster
  - Costlier per byte
  - ~100 cycles to access

- **Cache(s) (SRAM)**:
  - On Chip Storage
  - ~10’s of cycles to access

- **Registers**:
  - 1 cycle to access

CPU instrs can directly access

Smaller Faster Costlier per byte

Larger Slower Cheaper per byte
Where does accessing the network belong?

- Larger
- Slower
- Cheaper per byte

- Smaller
- Faster
- Costlier per byte

Local secondary storage (disk)

~100 M cycles to access

Cache(s) (SRAM)

~10’s of cycles to access

Main memory (DRAM)

~100 cycles to access

Registers

1 cycle to access

A: Here

B: Here

C: Somewhere else
The Memory Hierarchy

- **Local secondary storage (disk)**
  - Larger
  - Slower
  - Cheaper per byte
  - ~100 M cycles to access

- **Remote secondary storage**
  - (tapes, Web servers / Internet)
  - Slower than local disk to access

- **Main memory (DRAM)**
  - ~100 cycles to access

- **Cache(s) (SRAM)**
  - ~10’s of cycles to access

- **Registers**
  - 1 cycle to access

- **CPU instructions can directly access**

- **Smaller**
- **Faster**
- **Costlier per byte**