Computer Organization and Assembly Language (COAL)

Lecture 3

Dr. Naveed Anwar Bhatti

Webpage: naveedanwarbhatti.github.io

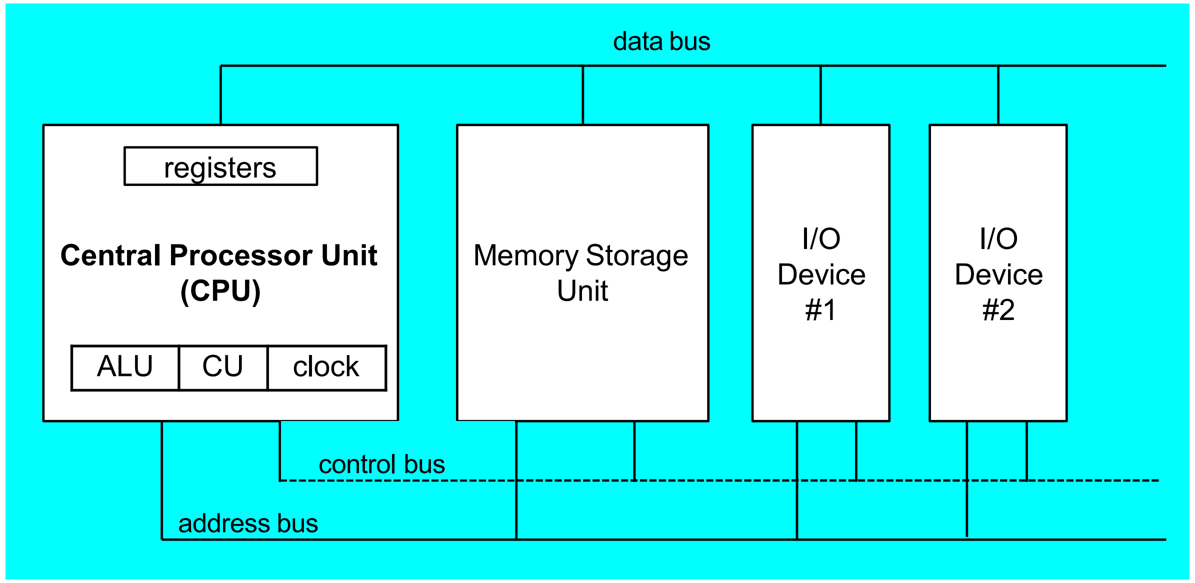
x86 Processor Architecture

- General Concepts
- IA-32 Processor Architecture
- IA-32 Memory Management
- 64-Bit Processors
- Components of an IA-32 Microcomputer
- Input-Output System

General Concepts

- Basic microcomputer design
- Instruction execution cycle
- Reading from memory
- How programs run

Basic Microcomputer Design



CPU

- Calculations and logical operations take place
- Contains:
  - a limited number of storage locations named **registers**
  - a high-frequency **clock**
  - a **control unit**
  - an **arithmetic logic unit**
- The *clock* synchronizes the internal operations of the CPU with other system components.
- The *control unit* (CU) coordinates the sequencing of steps involved in executing machine instructions.
- The *arithmetic logic unit* (ALU) performs arithmetic operations such as addition and multiplication and logical operations such as AND, OR, and NOT
- CPU is attached to the rest of the computer via pins
- Most pins connect to the data bus, the control bus, and the address bus.

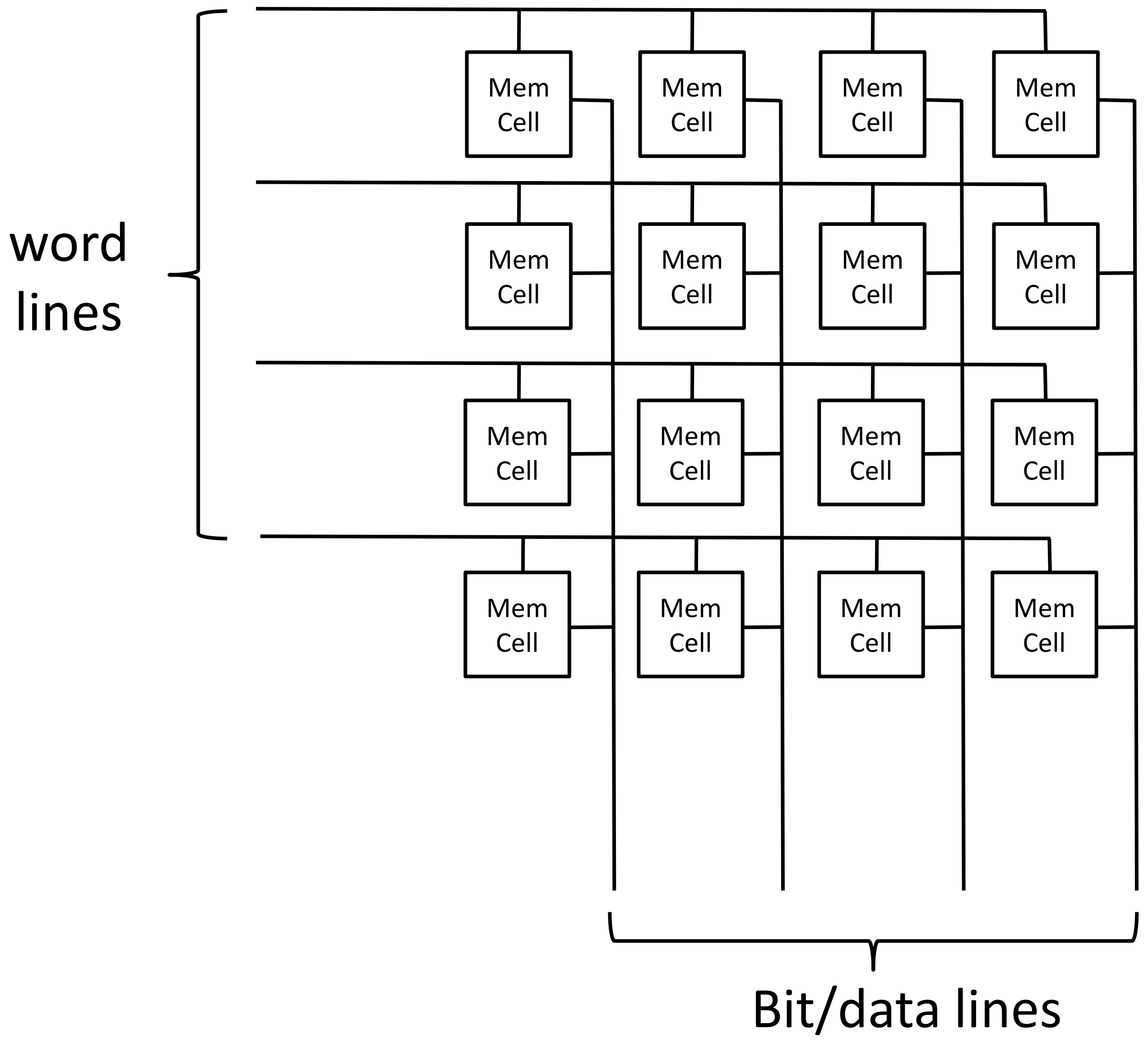
Memory Storage Unit

- Also known as main/internal/primary storage
- Instructions and data are held while a computer program is running
- Receives requests for data from the CPU
- Transfers data from RAM to the CPU and vice versa
- Consists of 2 types of memory:
  - ROM
  - RAM

Memory technologies landscape

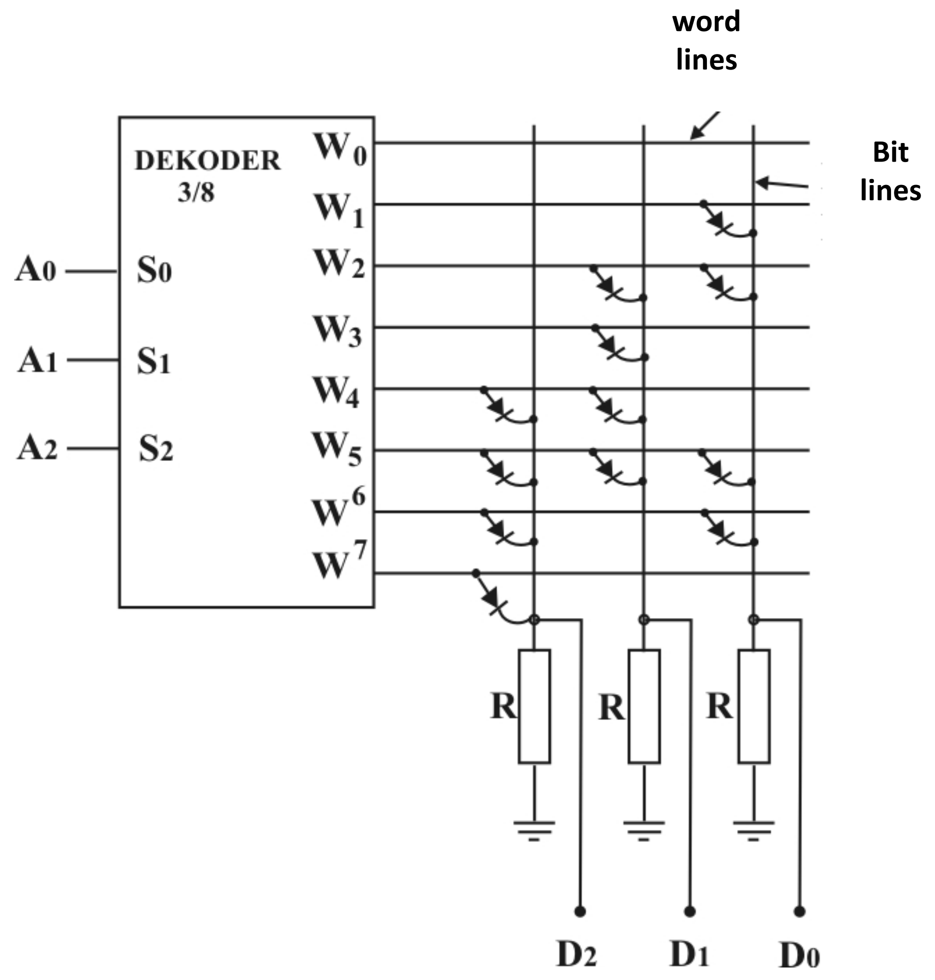
	Volatile	Non-Volatile
RAM	Static RAM (SRAM) Dynamic RAM (DRAM)	EEPROM Flash Memory FRAM MRAM BBSRAM
ROM	n/a	Mask ROM OTPROM EPROM

Internal organization of memory is usually an array



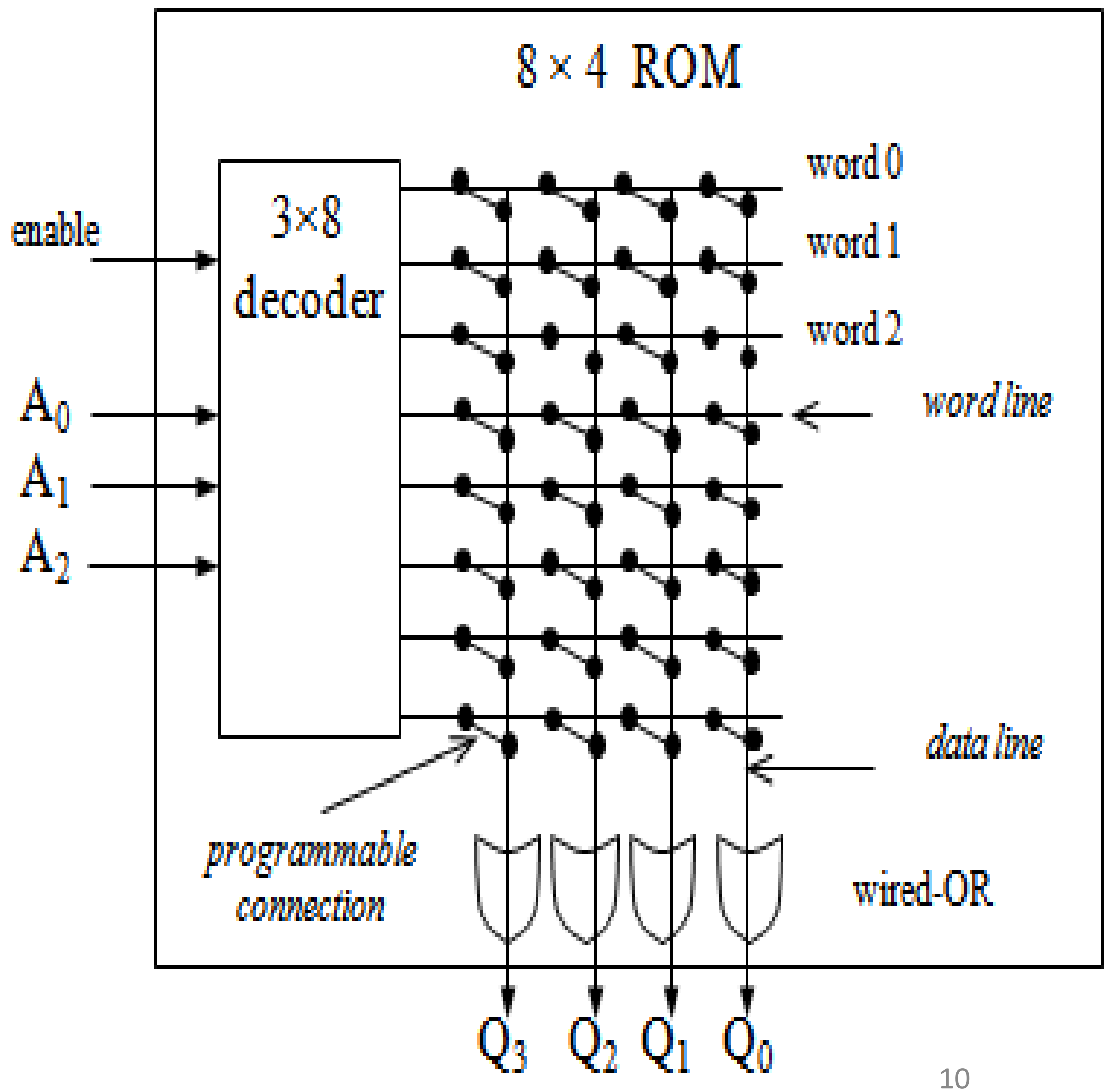
Non-Volatile Memory: Mask ROM

- The “simplest” memory technology
- **Presence/absence of diode** at each cell denote value
- Contents are **fixed** when chip is made; cannot be changed
- High upfront setup costs (mask costs)
- Small recurring marginal costs
- Good for which applications?



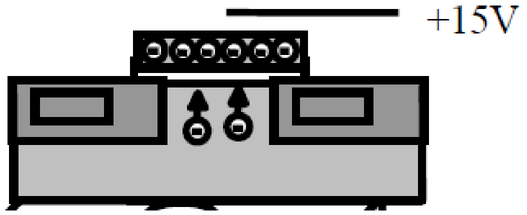
OTP ROM with Example use

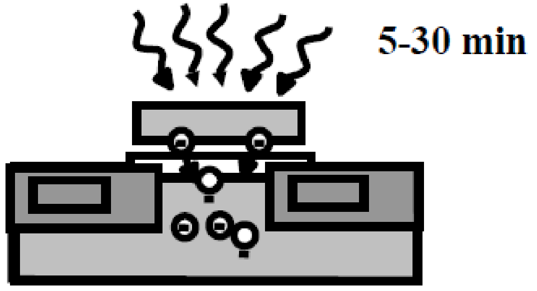
- Fuses that can be burned once
- Example:
  - Lines connected only at circles
  - Address input is 010
  - Output is ______?

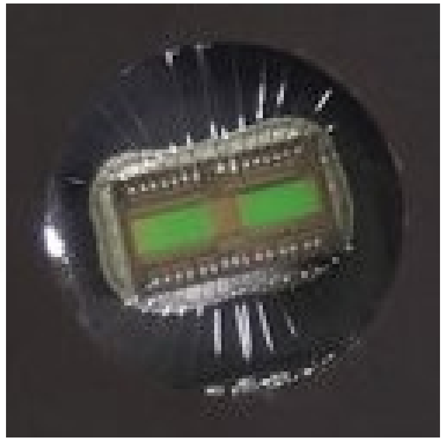


Non-Volatile Memory: EPROM

- Erasable Programmable Read-Only Memory
- Constructed from floating gate FET.
- Charge trapped on the FG disconnects src and drain
  - By applying high voltage (15V +) to the control gate
    - "Writes" the cell with a 0
    - Writing means changing from 1 → 0
- Erasing means changing form 0 → 1
  - Uses UV light (not electrically!)
  - Electrons are trapped on a floating gate
- Erase unit is the whole device
- Retains data for 10-20 years
  - But susceptible to radiation!
- **PROM (or OTP) is same, just w/o window**







Non-Volatile Memory: EEPROM

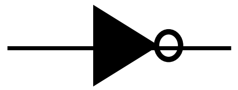
- Electrically Erasable and Programmable ROM
- Can erase bit with higher than normal voltage (instead of UV)
- Unlike the EPROMs can program and erase individual words
  - Why?
- Slow Write due to erasing and programming
- Similar storage permanence to EPROM (about 10 years)
  - but more expensive

Non-Volatile Memory: Flash Memory

- Electrically erasable (like EEPROM, unlike EPROM)
- Floating gate technology
  - **Erase/write cycles are limited** (10K to 100K, typically)
- Used in many reprogrammable systems these days
- **Erase size is block** (not word)
  - Blocks are a few KB
- Reads are like standard RAM
- Write can be slow
  - **Entire block erased and rewritten for change to a single word**.
  - The erase is time consuming, writing is fast!

Volatile Memory: Static RAM and Dynamic RAM

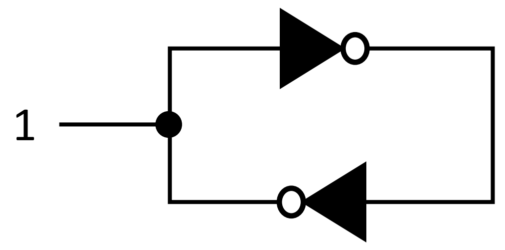
Static RAM



Dynamic RAM

Volatile Memory: Static RAM and Dynamic RAM

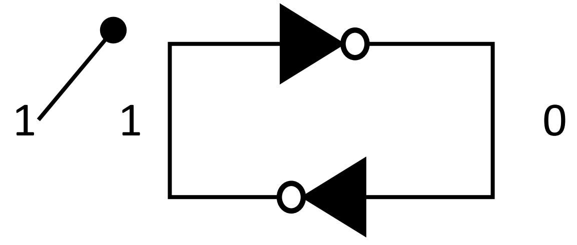
Static RAM



Dynamic RAM

Volatile Memory: Static RAM and Dynamic RAM

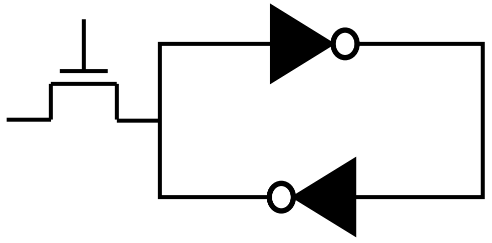
Static RAM



Dynamic RAM

Volatile Memory: Static RAM and Dynamic RAM

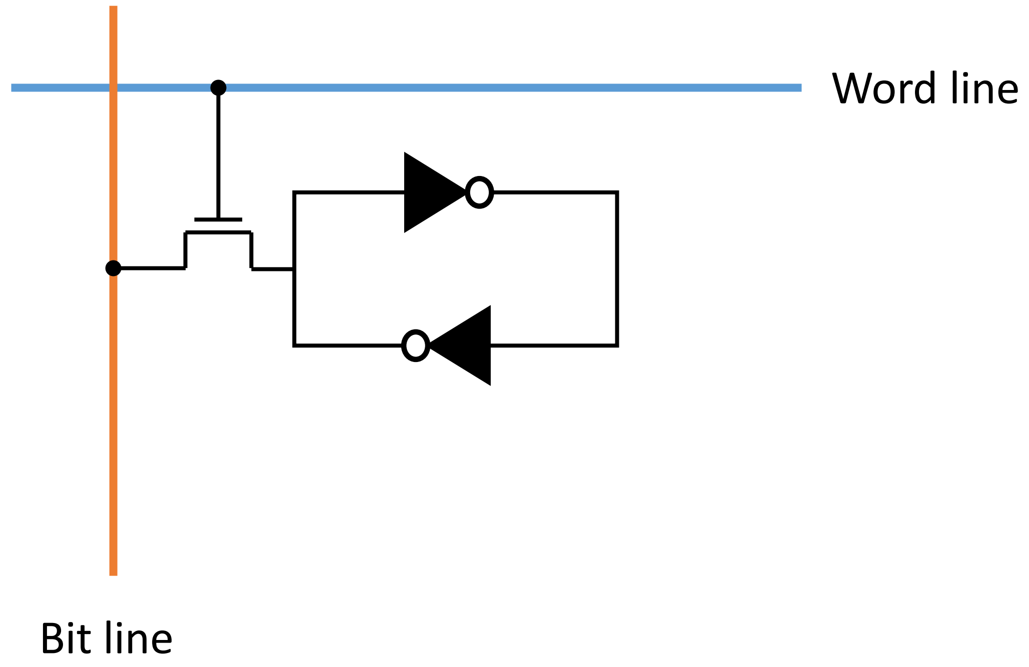
Static RAM



Dynamic RAM

Volatile Memory: Static RAM and Dynamic RAM

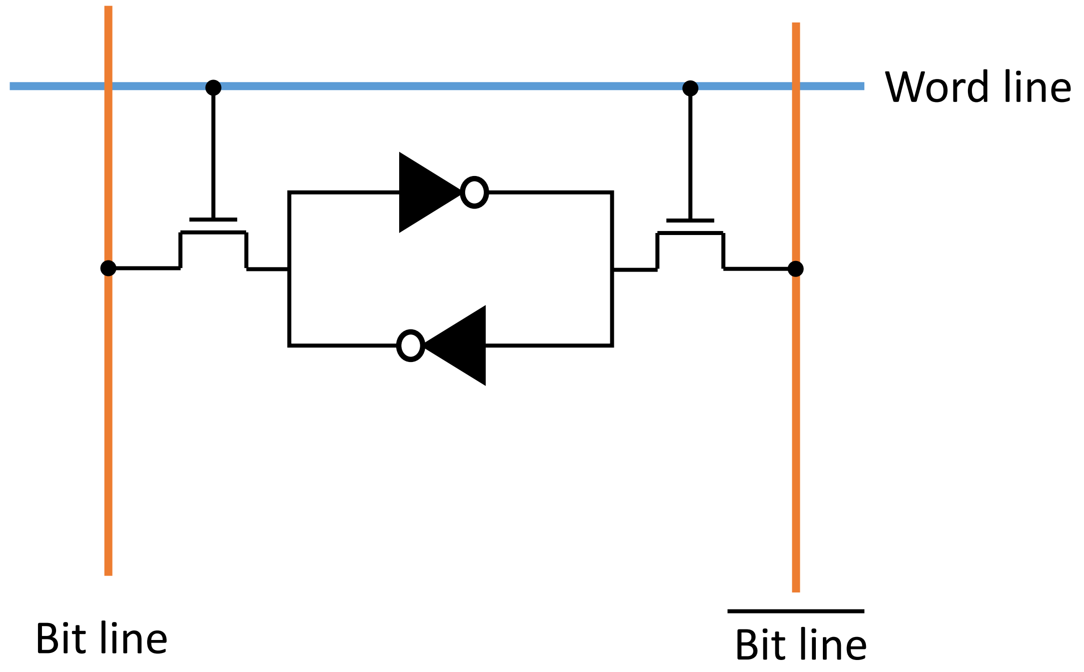
Static RAM



Dynamic RAM

Volatile Memory: Static RAM and Dynamic RAM

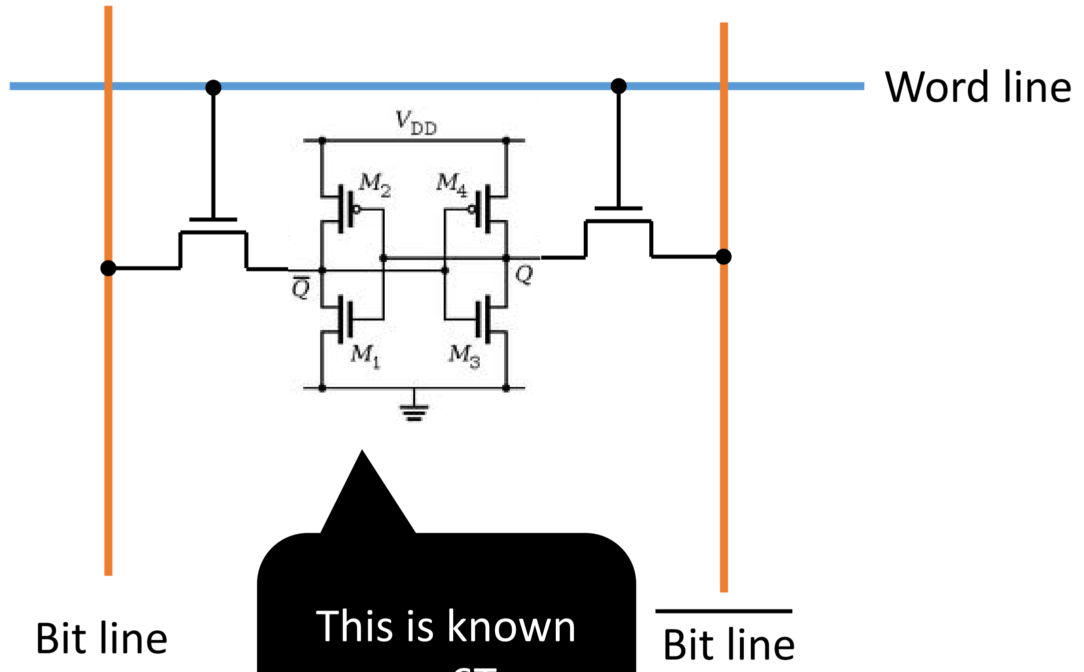
Static RAM



Dynamic RAM

Volatile Memory: Static RAM and Dynamic RAM

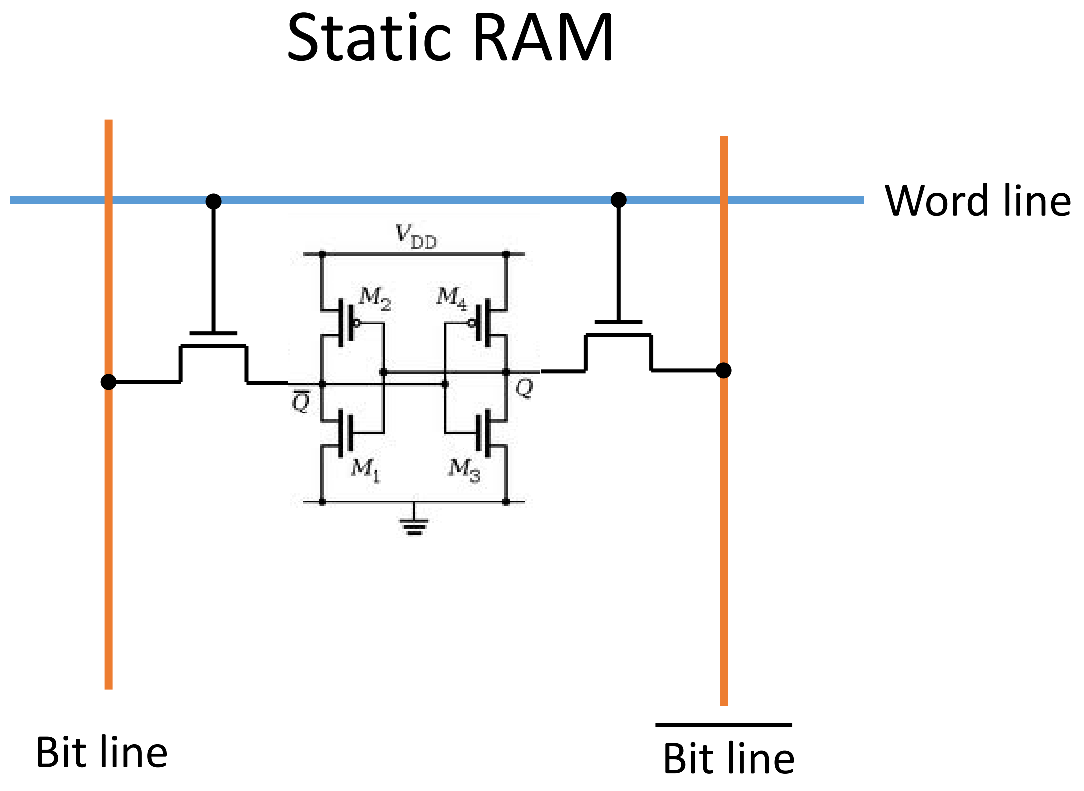
Static RAM

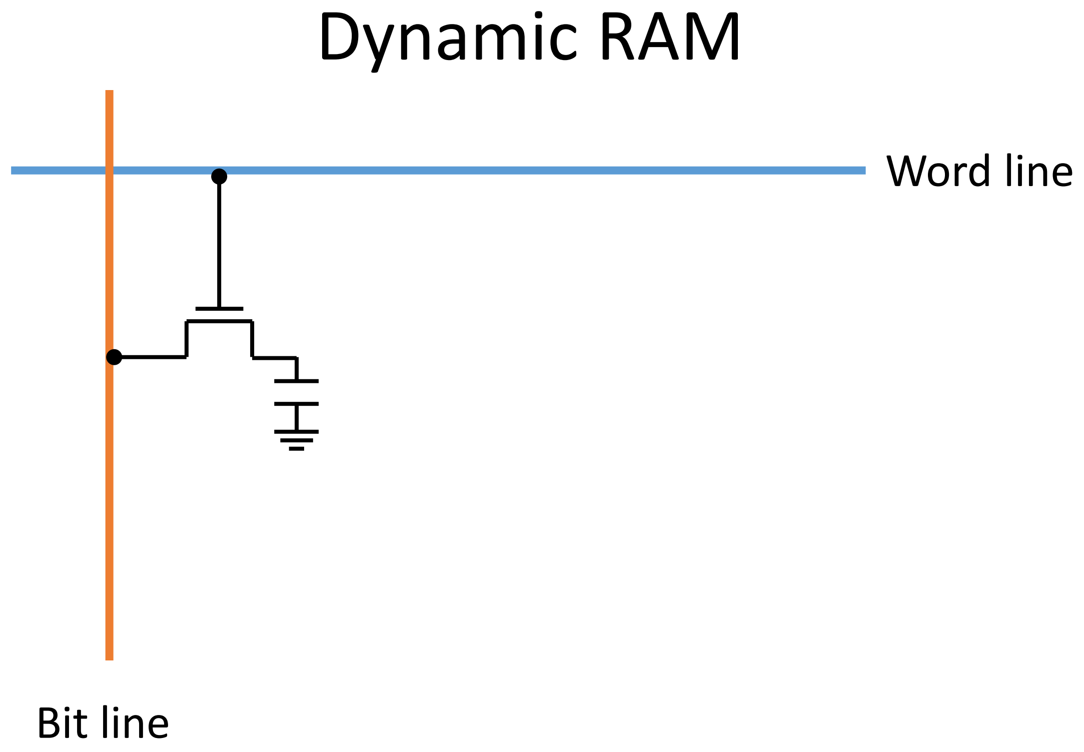


This is known as 6T configuration

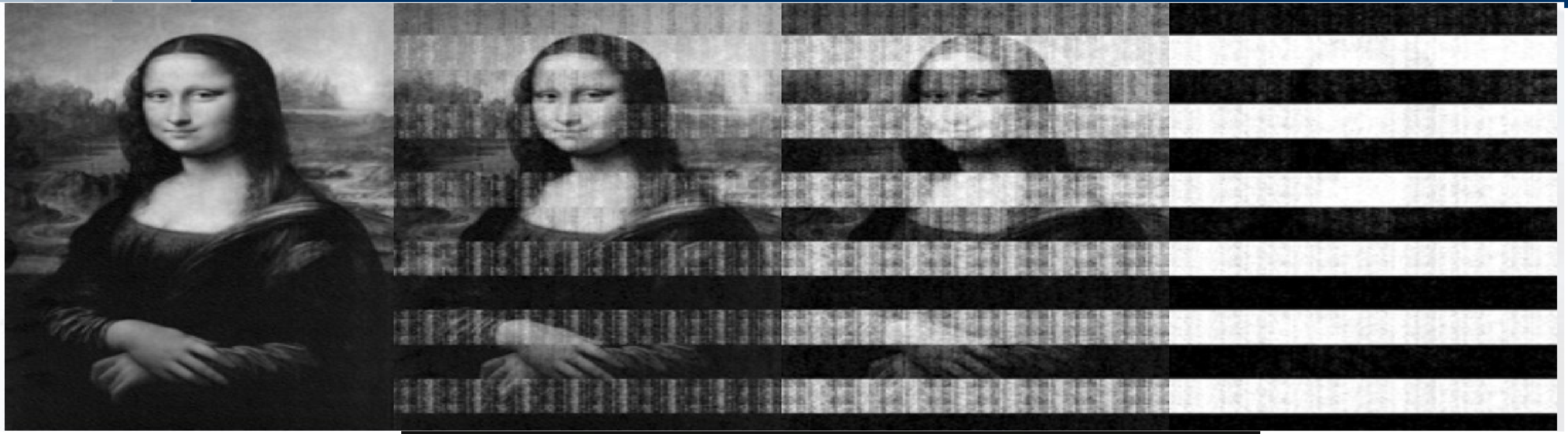
Dynamic RAM

Volatile Memory: Static RAM and Dynamic RAM

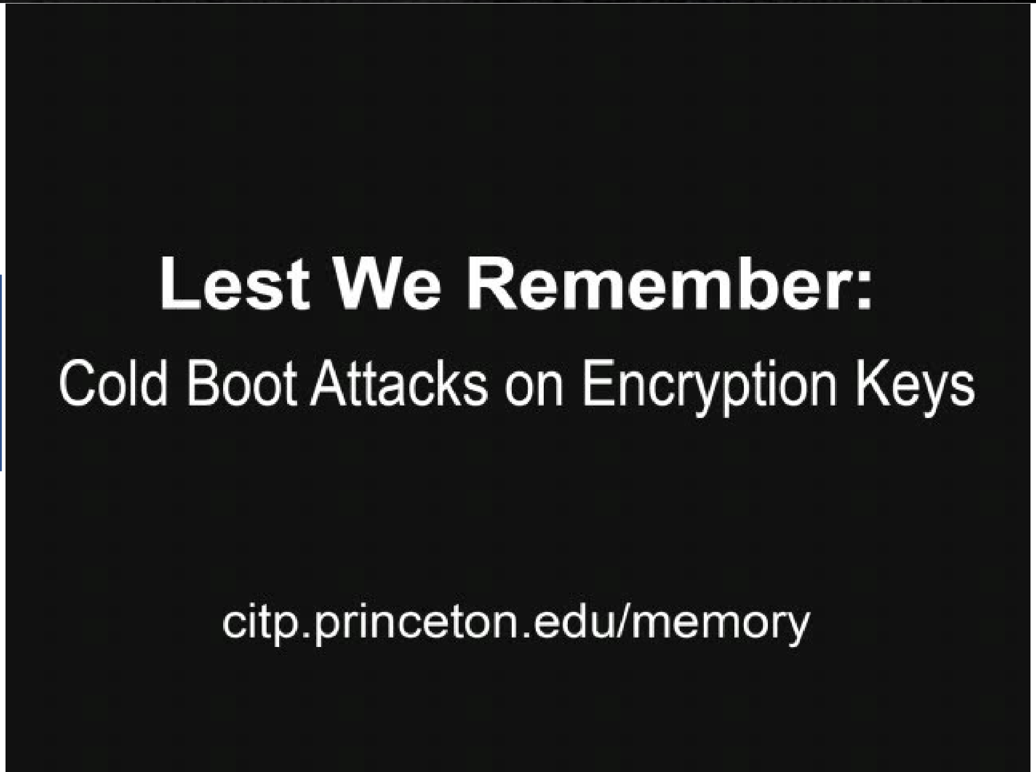




Is RAM really that volatile?



Not really!
Here is an example of bad usage

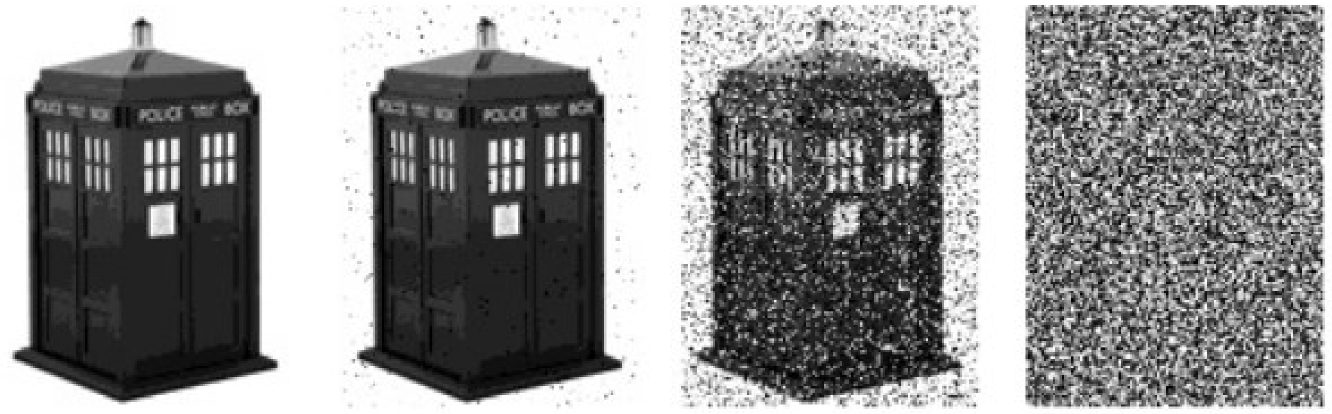


Lest We Remember:

Cold Boot Attacks on Encryption Keys

citp.princeton.edu/memory

A good use!



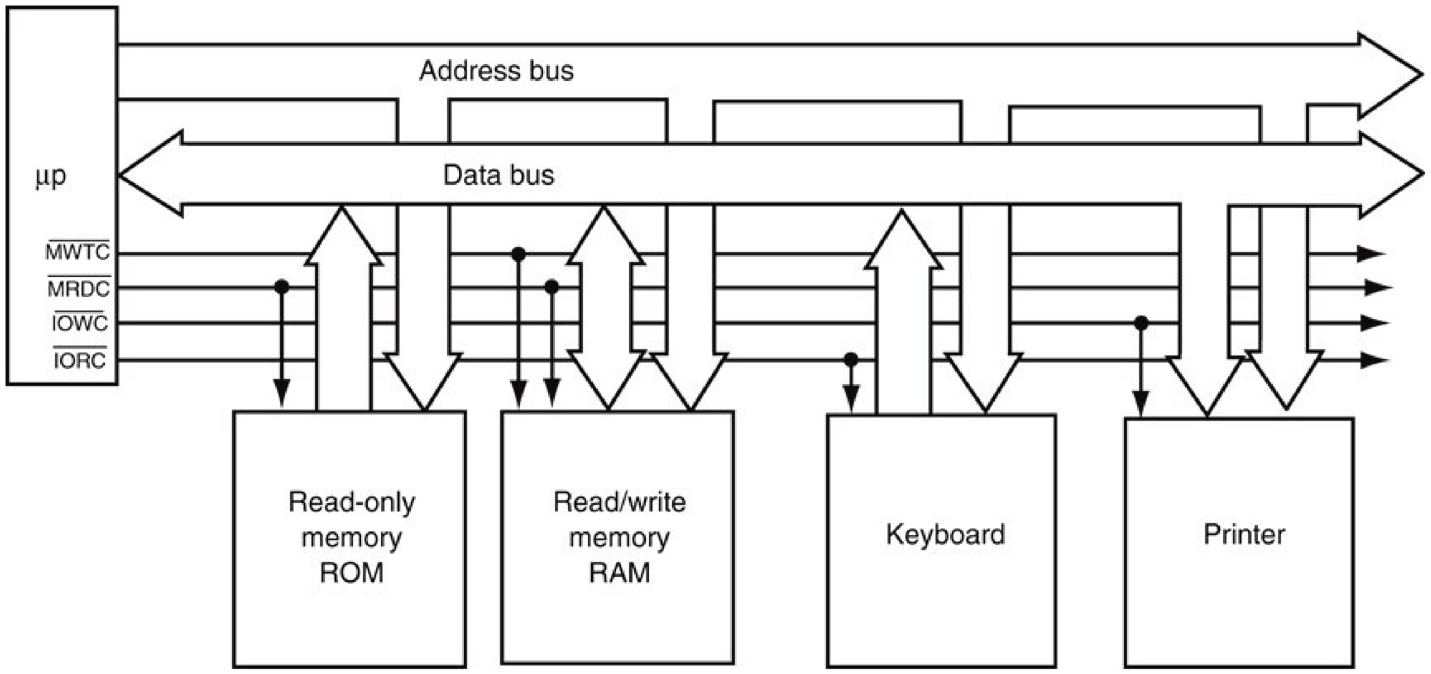
Images: Kevin Fu

DATA DEMATERIALIZES: Data (like this picture of the Tardis) stored in an RFID chip's SRAM decays. The TARDIS technology uses that decay as a clock that tells when the chip last received power.

http://spectrum.ieee.org/semiconductors/memory/could-an-sram-hourglass-save-rfid-chips-just-in-time

Buses

- A common group of wires that interconnect components in a computer system.
- Transfer **address**, **data**, & **control** information between microprocessor, memory and I/O.
- Three buses exist for this transfer of information: address, data, and control.



Buses

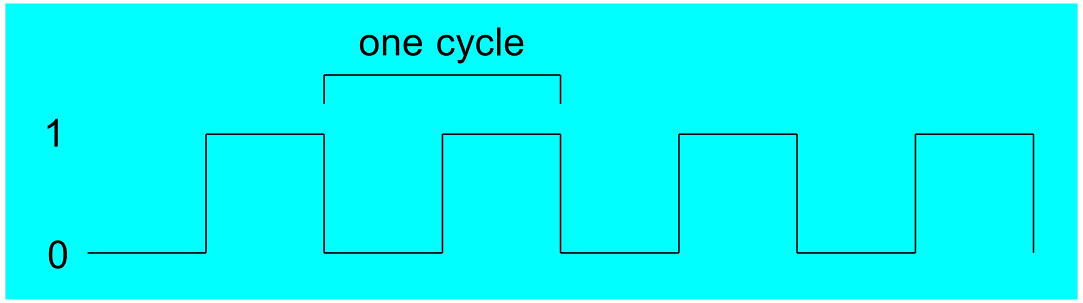
- The **data bus** transfers information between the microprocessor and its memory and I/O address space.
- Data transfers vary in size, from **8 bits** wide to **64 bits** wide in various Intel microprocessors.
  - 8088 has an 8-bit data bus that transfers 8 bits of data at a time
  - 8086, 80286, 80386SL, 80386SX, and 80386EX transfer 16 bits of data
  - 80386DX, 80486SX, and 80486DX, 32 bits
  - Pentium through Core2 microprocessors transfer **64 bits of data**
- Advantage of a wider data bus is speed in applications using wide data.

Buses

- **Control bus** lines select and cause memory or I/O to perform a read or write operation.
- In most computer systems, there are four control bus connections:
  - *MRDC* (**memory read control**)
  - *MWTC* (**memory write control**)
  - *IORC* (**I/O read control**)
  - *IOWC* (**I/O write control**).
- overbar indicates the control signal is active-low; (active when logic zero appears on control line)

Clock

- Synchronizes all CPU and BUS operations
- Machine (clock) cycle measures time of a single operation
- Clock is used to trigger events

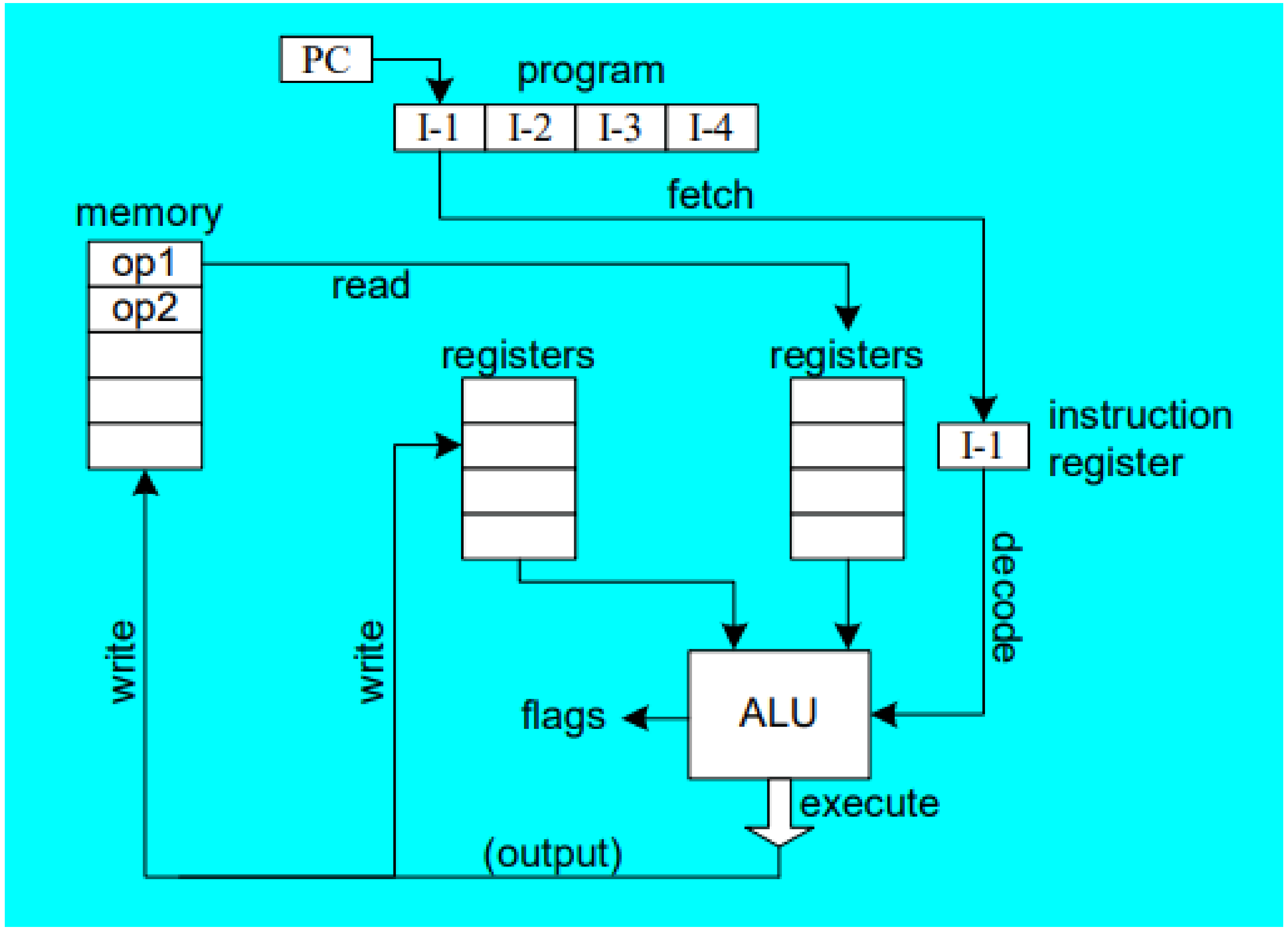


Next Comes

- Basic microcomputer design
- **Instruction execution cycle**
- **Reading from memory**
- **How programs run**

Instruction Execution Cycle

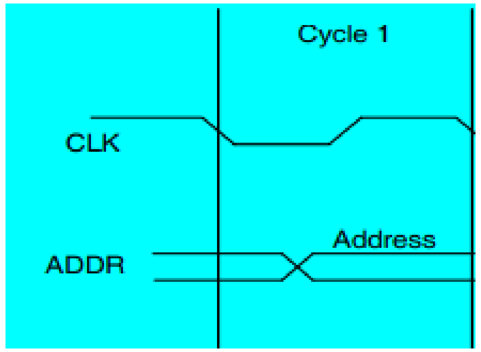
Fetch Instruction
- Decode
- Fetch operands
- Execute
- Store output



Reading from Memory

Multiple machine cycles are required when reading from memory, because it responds much more slowly than the CPU. The steps are:

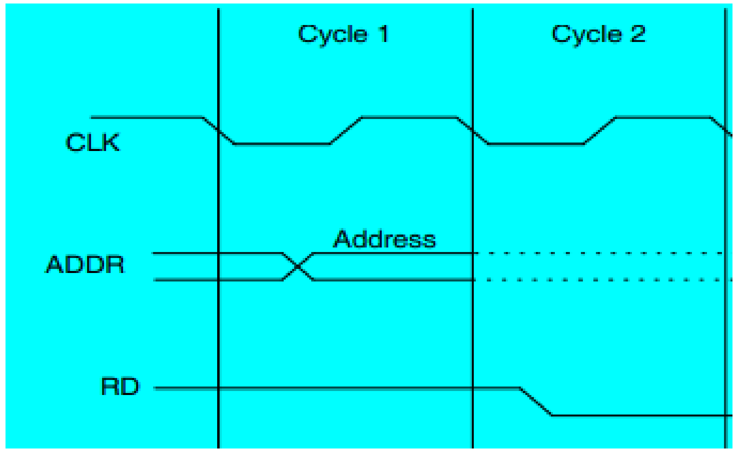
1. Place the address of the value you want to read on the **address bus**



Reading from Memory

Multiple machine cycles are required when reading from memory, because it responds much more slowly than the CPU. The steps are:

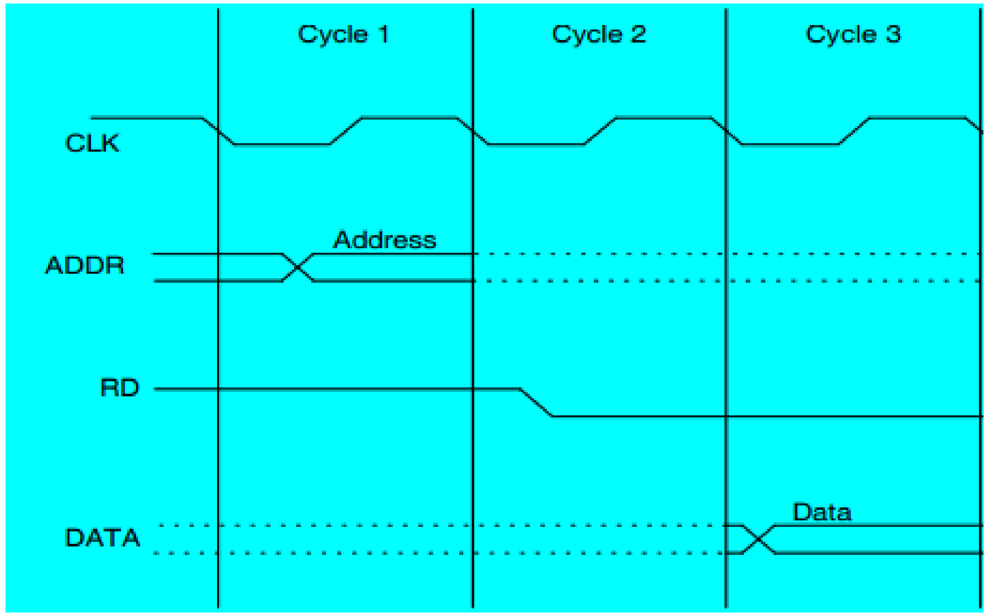
1. Place the address of the value you want to read on the **address bus**.
2. Assert (changing the value of) the processor's RD (read) pin.



Reading from Memory

Multiple machine cycles are required when reading from memory, because it responds much more slowly than the CPU. The steps are:

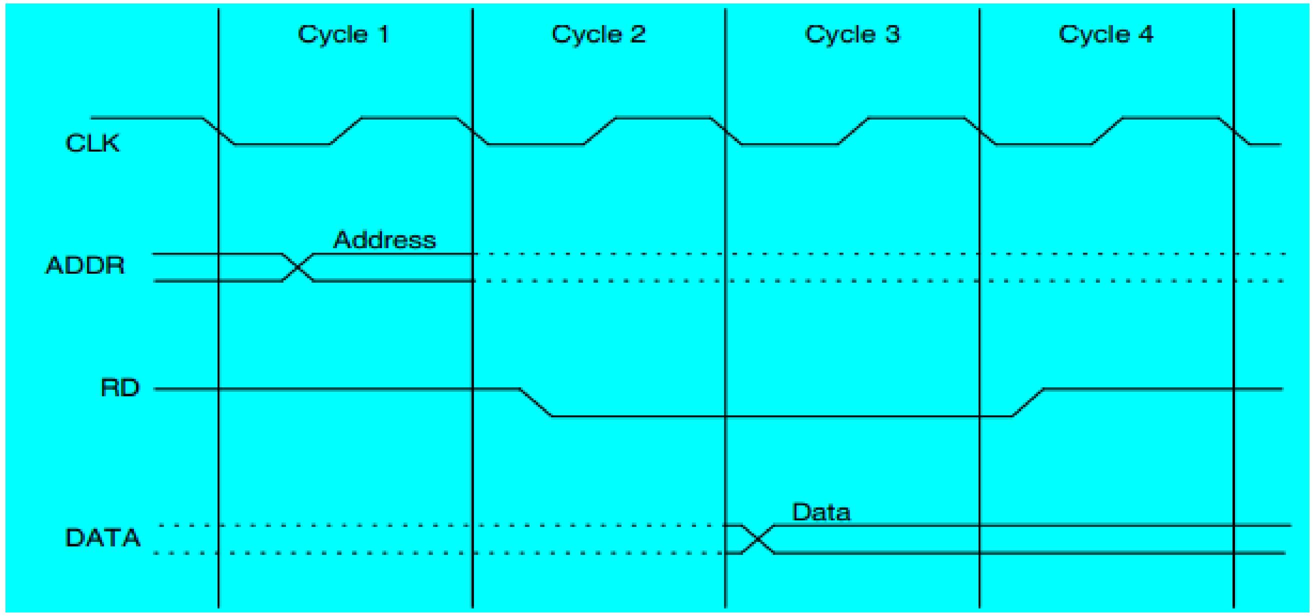
1. Place the address of the value you want to read on the **address bus**.
2. Assert (changing the value of) the processor's RD (read) pin.
3. Wait one clock cycle for the memory chips to respond.



Reading from Memory

Multiple machine cycles are required when reading from memory, because it responds much more slowly than the CPU. The steps are:

1. Place the address of the value you want to read on the **address bus**.
2. Assert (changing the value of) the processor's RD (read) pin.
3. Wait one clock cycle for the memory chips to respond.
4. Copy the data from the data bus into the destination operand



Cache Memory

- High-speed expensive static RAM both inside and outside the CPU.
  - Level-1 cache: inside the CPU
  - Level-2 cache: outside the CPU
- Cache hit: when data to be read is already in cache memory
- Cache miss: when data to be read is not in cache memory.

How a Program Runs

- OS searches for the program's filename in the current disk directory/paths, if not found then issues an error message

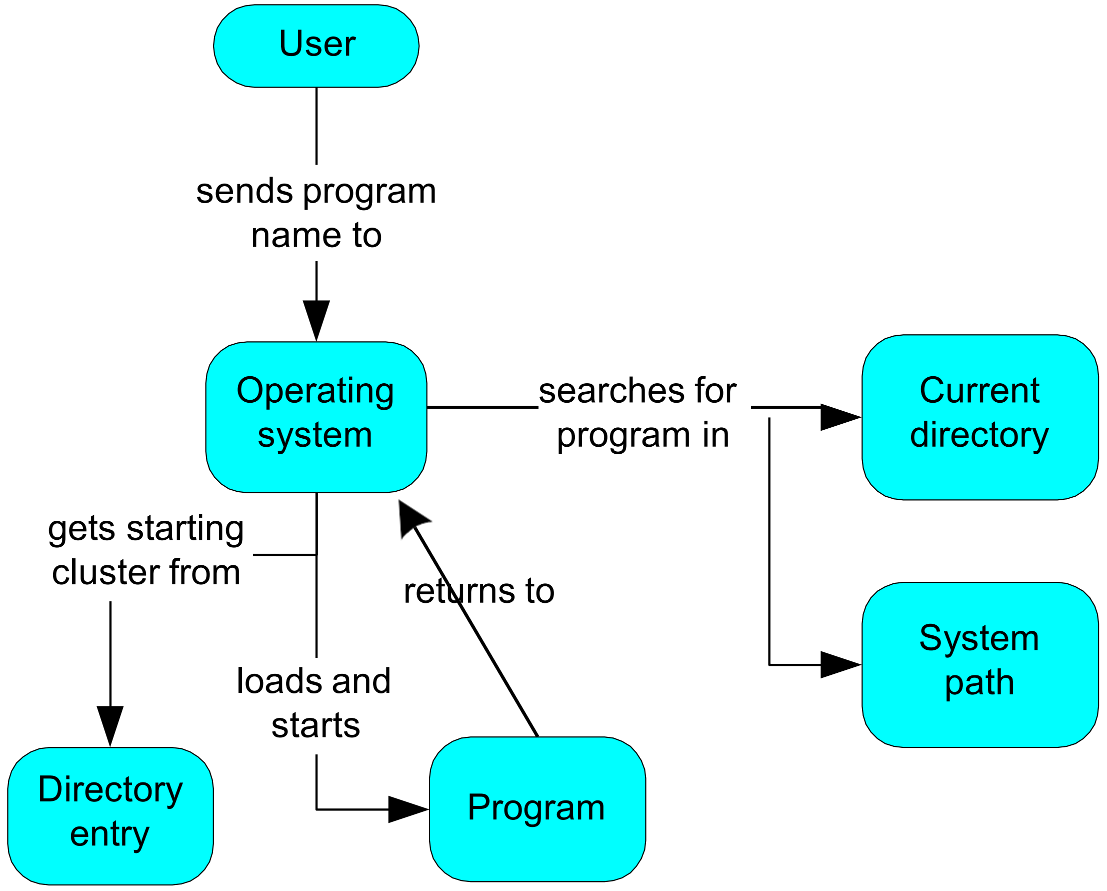
- OS retrieves basic information about the program's file from the disk directory, including the file size and its physical location on the disk drive

- OS determines the next available location in memory and loads the program file into memory. It allocates a block of memory to the program and enters information about the program's size and location into a table (sometimes called a ***descriptor table***)

How a Program Runs

- The OS begins execution of the program's first machine instruction (its entry point)
- As soon as the program begins running, it is called a ***process***. The OS assigns the process an identification number (***process ID***), which is used to keep track of it while running.
- The *process* runs by itself. It is the OS's job to track the execution of the process and to respond to requests for system resources
- When the process ends, it is removed from memory

How a Program Runs



What's Next

- General Concepts
- **IA-32 Processor Architecture**
- IA-32 Memory Management
- 64-Bit Processors
- Components of an IA-32 Microcomputer
- Input-Output System

IA-32 Processor Architecture

- Modes of operation
- Basic execution environment
- Floating-point unit
- Intel Microprocessor history

Modes of Operation

- The operating mode controls how the processor sees and **manages the system memory** and the tasks that use it.
- **Protected mode**
  - native mode (Windows, Linux)
- **Real-address mode**
  - native MS-DOS
- **System management mode**
  - power management, system security, diagnostics

- Virtual-8086 mode
  - hybrid of Protected
  - each program has its own 8086 computer

Modes of Operation

- One of the big differences between the operating modes is in the way memory addressing works.

- Both the amount of memory that can be addressed and the translation process between logical addresses and to physical addresses may vary depending on the operating mode

Basic Execution Environment

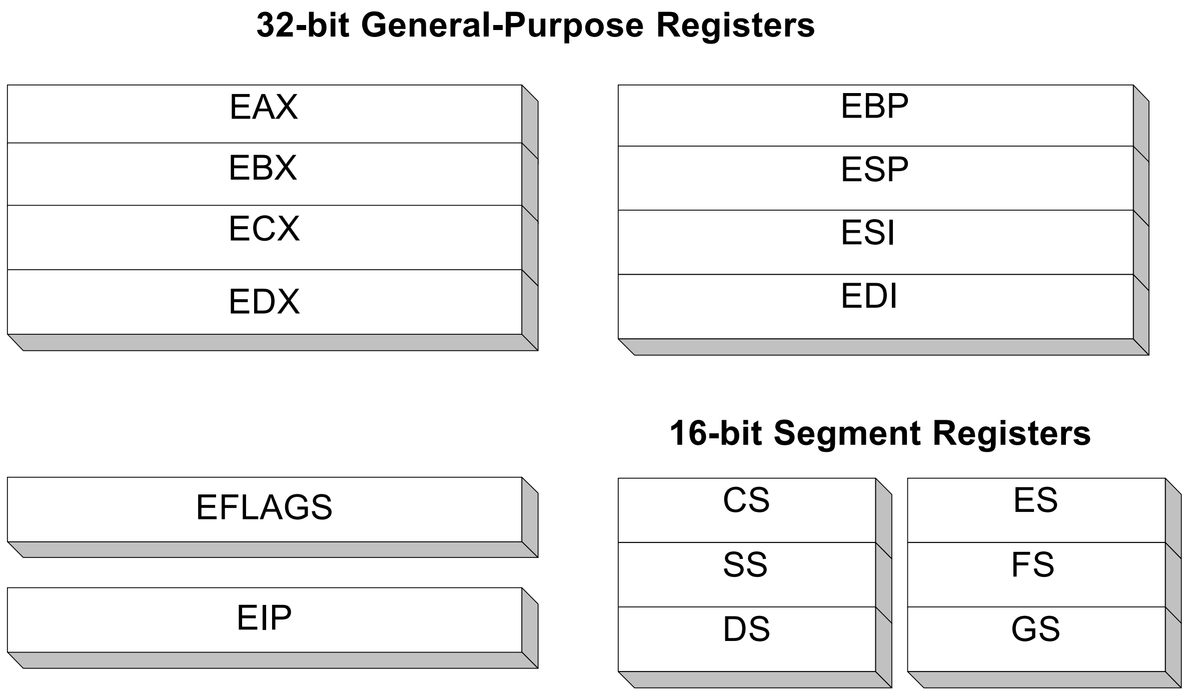
- Addressable memory
- General-purpose registers
- Index and base registers
- Specialized register uses
- Status flags
- Floating-point, MMX, XMM registers

Addressable Memory

- Protected mode
  - 4 GB
  - 32-bit address
- Real-address and Virtual-8086 modes
  - 1 MB space
  - 20-bit address

General-Purpose Registers

Named storage locations inside the CPU, optimized for speed.

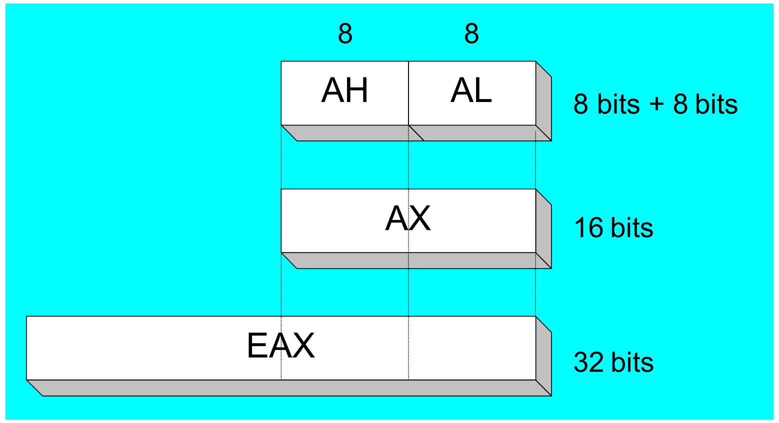


Some Specialized Register Uses (1 of 2)

- General-Purpose
  - EAX/EDX – accumulator
  - ECX – loop counter
  - ESP – stack pointer
  - ESI, EDI – index registers
  - EBP – extended frame pointer (stack)
- Segment
  - CS – code segment
  - DS – data segment
  - SS – stack segment
  - ES, FS, GS - additional segments

Accessing Parts of Registers

- Use 8-bit name, 16-bit name, or 32-bit name
- Applies to EAX, EBX, ECX, and EDX



32-bit	16-bit	8-bit (high)	8-bit (low)
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL

Index and Base Registers

- Some registers have only a 16-bit name for their lower half:

32-bit	16-bit
ESI	SI
EDI	DI
EBP	BP
ESP	SP

Some Specialized Register Uses (2 of 2)

- EIP – instruction pointer
- EFLAGS
  - status and control flags
  - each flag is a single binary bit

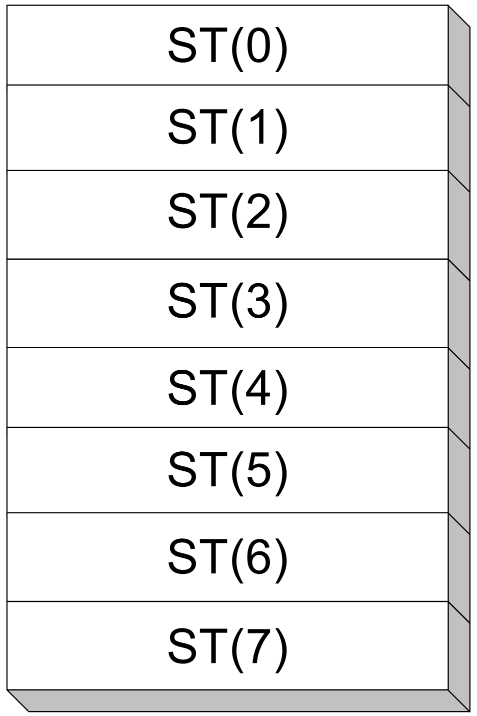
Status Flags

- Carry
  - unsigned arithmetic out of range
- Overflow
  - signed arithmetic out of range
- Sign
  - result is negative
- Zero
  - result is zero
- Auxiliary Carry
  - carry from bit 3 to bit 4
- Parity
  - sum of 1 bits is an even number

Floating-Point, MMX, XMM Registers

- Eight 80-bit floating-point data registers
  - ST(0), ST(1), . . . , ST(7)
  - arranged in a stack
  - used for all floating-point arithmetic

- Eight 64-bit MMX registers
- Eight 128-bit XMM registers for single-instruction multiple-data (SIMD) operations



What's Next

- General Concepts
- IA-32 Processor Architecture

  IA-32 Memory Management
- 64-Bit Processors
- Components of an IA-32 Microcomputer
- Input-Output System

X86 Memory Management

- x86 processors manage memory according to the basic modes of operation
- Protected mode is the most **robust and powerful**, but it does restrict application programs from directly accessing system hardware

Real Address Mode

- Only 1 MByte of memory can be addressed
  - from hexadecimal 00000 to FFFFF
- The processor can run only one program at a time
  - it can momentarily interrupt that program to process requests (called *interrupts*) from peripherals
- Application programs are permitted to access any memory location, including addresses that are linked directly to system hardware
- The MS-DOS operating system runs in real-address mode, and Windows 95 and 98 can be booted into this mode

Protected Mode

- Processor can run multiple programs at the same time
- 4 GB addressable RAM
  - (00000000 to FFFFFFFFh)
- Each program assigned a memory partition which is protected from other programs
- Designed for multitasking
- Supported by Linux & MS-Windows

Virtual-8086 Mode

- Computer runs in protected mode and creates a virtual-8086 machine with its own 1-MByte address space that simulates an 80x86 computer running in real address mode

What's Next

- General Concepts
- IA-32 Processor Architecture
- IA-32 Memory Management

64-Bit Processors

- Components of an IA-32 Microcomputer
- Input-Output System

64-Bit Processors (1/2)

- The instruction set is a 64-bit extension of the x86 instruction set
- Some of the essential features:
  - It is backward-compatible with the x86 instruction set
  - Addresses are 64 bits long, allowing for a virtual address space of size **16 exabytes**. In current chip implementations, only the lowest 48 bits are used
  - It can use 64-bit general-purpose registers, allowing instructions to have 64-bit integer operands.
  - It uses eight more general-purpose registers than the x86.
  - It uses a 48-bit physical address space, which supports up to **256 terabytes** of RAM.

- There is a *legacy mode* that still supports 16-bit programming, but it is not available in 64-bit versions of Microsoft Windows.

64-Bit Processors (2/2)

- 64-Bit Operation Modes
  - Compatibility mode – can run existing 16-bit and 32-bit applications (Windows supports only 32-bit apps in this mode)
  - 64-bit mode – Windows 64 uses this, the processor runs applications that use the 64-bit linear address space
- Basic Execution Environment
  - addresses can be 64 bits (48 bits, in practice)
  - 16 64-bit general purpose registers
  - 64-bit instruction pointer named RIP
  - 8 80-bit floating-point registers
  - A 64-bit status flags register named RFLAGS

64-Bit General Purpose Registers

- 64-bit general purpose registers:
  - RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP, R8, R9, R10, R11, R12, R13, R14, R15

- 32-bit general purpose registers:
  - EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP, R8D, R9D, R10D, R11D, R12D, R13D, R14D, R15D

Specialized Registers

- There are some specialized registers for multimedia processing
  - Eight 64-bit MMX registers
  - Sixteen 128-bit XMM registers (in 32-bit mode, you have only 8 of these)

Operand Size	Available Registers
8 bits	AL, BL, CL, DL, DIL, SIL, BPL, SPL, R8L, R9L, R10L, R11L, R12L, R13L, R14L, R15L
16 bits	AX, BX, CX, DX, DI, SI, BP, SP, R8W, R9W, R10W, R11W, R12W, R13W, R14W, R15W
32 bits	EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP, R8D, R9D, R10D, R11D, R12D, R13D, R14D, R15D
64 bits	RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP, R8, R9, R10, R11, R12, R13, R14, R15

What's Next

- General Concepts
- IA-32 Processor Architecture
- IA-32 Memory Management
- 64-Bit Processors

Components of an IA-32 Microcomputer

- Input-Output System

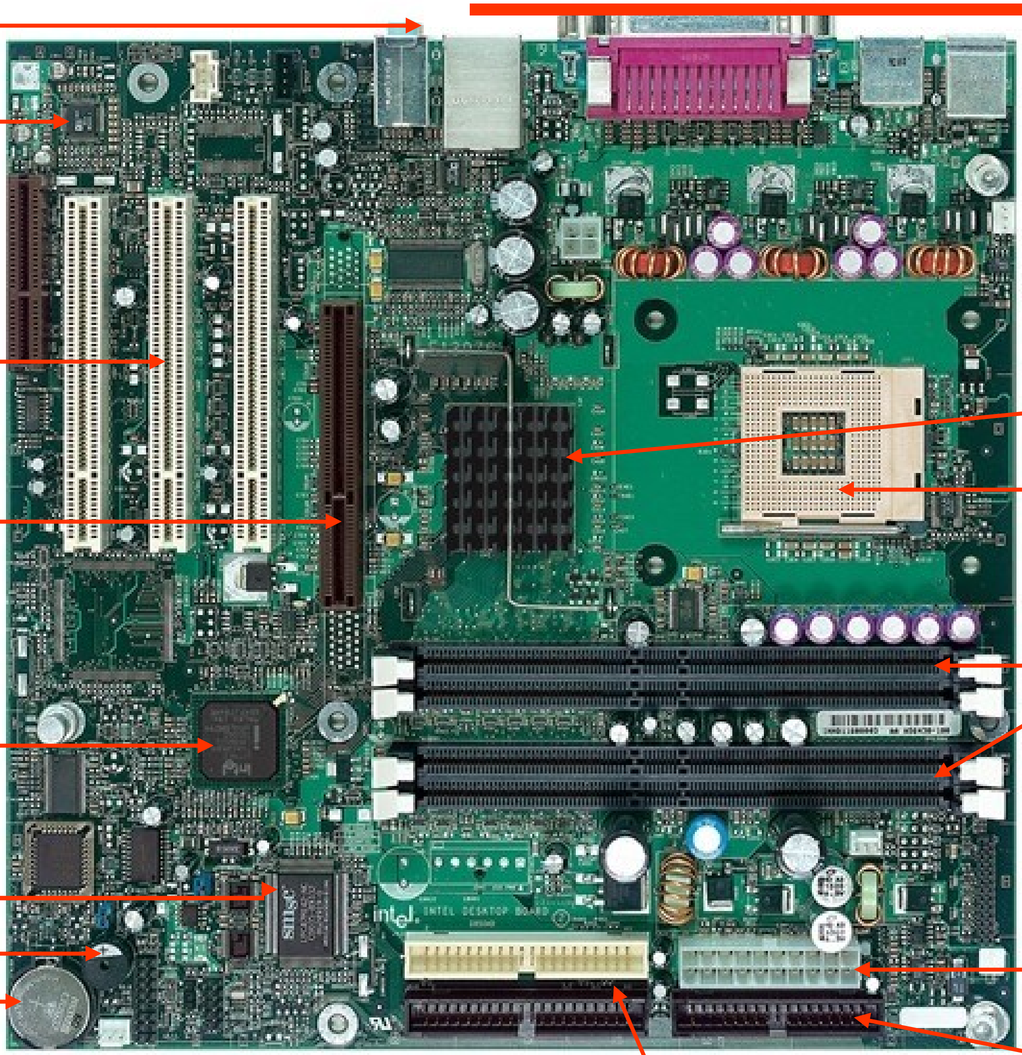
Components of an IA-32 Microcomputer

- Motherboard
- Memory
- Input-output ports

Motherboard

- CPU socket
- External cache memory slots
- Main memory slots
- BIOS chips
- Sound synthesizer chip (optional)
- Video controller chip (optional)
- IDE, parallel, serial, USB, video, keyboard, joystick, network, and mouse connectors
- PCI bus connectors (expansion cards)

Intel D850MD Motherboard



Source: Intel® Desktop Board D850MD/D850MV Technical Product Specification

Input-Output Ports

- USB (universal serial bus)
  - intelligent high-speed connection to devices
  - up to 12 megabits/second
  - USB hub connects multiple devices

- Parallel
  - short cable, high speed
  - common for printers
  - bidirectional, parallel data transfer

- Serial
  - RS-232 serial port
  - UART Protocol

Wired Technology: UART

UART: Universal Asynchronous Receiver Transmitter

A simple half-duplex, asynchronous, serial protocol.

What makes it ‘universal’ ?

Its parameters (format,speed ..) are configurable.

Why ‘asynchronous’ ?

It doesn’t have a clock

UART Functions

Outbound data

- Convert from parallel to serial
- Add start and stop delineators (bits)
- Add parity bit

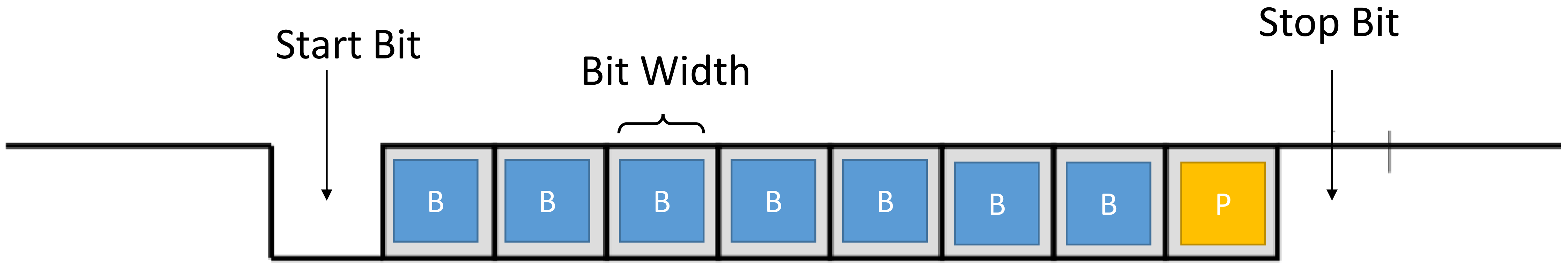
Inbound data

- Convert from serial to parallel
- Remove start and stop delineators (bits)
- Check and remove parity bit

UART Basics

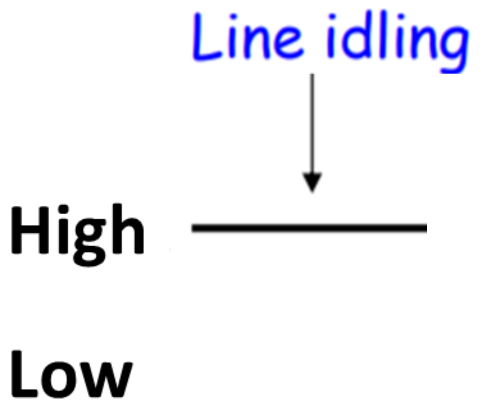
- **Baud Rate**
  - Number of bits per sec

- **Format of Communication**
  - The start bit marks the beginning of a new word
  - Next follows the data bits (7 or 8)
  - The parity bit is added to make the number of 1's even or odd
  - The stop bit marks the end of transmission



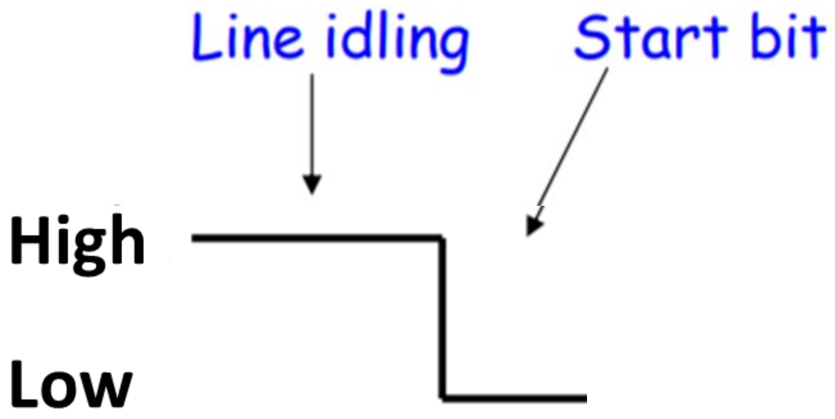
UART Transmission Example

Send the ASCII letter ‘W’ (1010111)



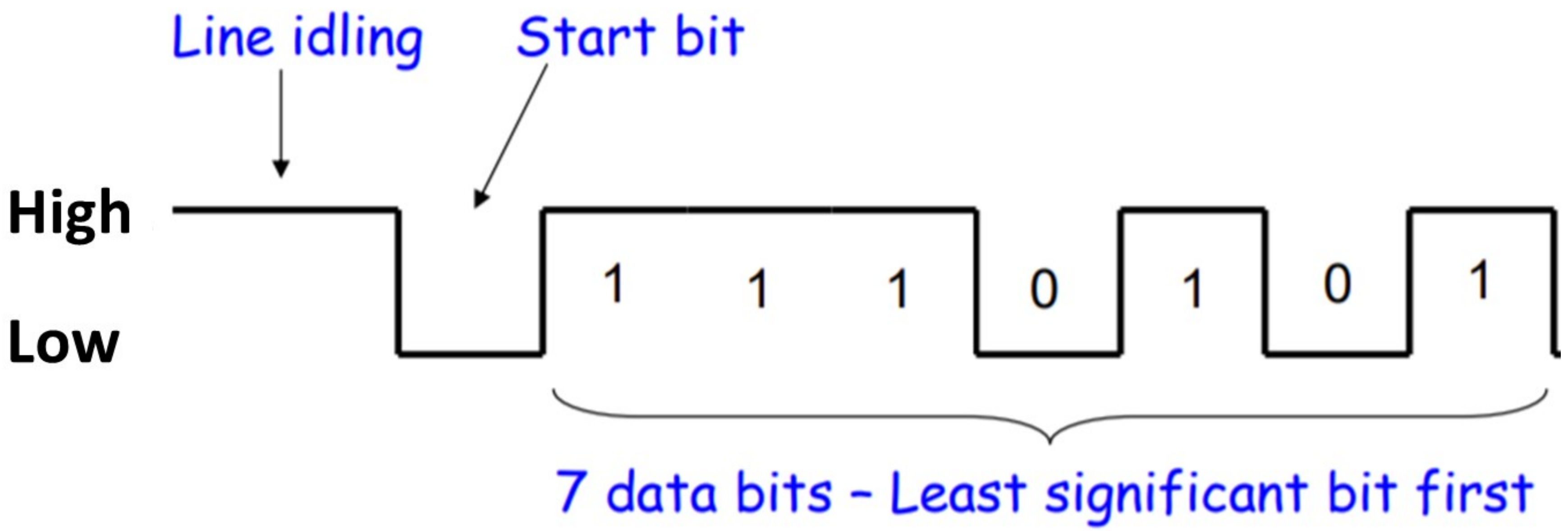
UART Transmission Example

Send the ASCII letter ‘W’ (1010111)



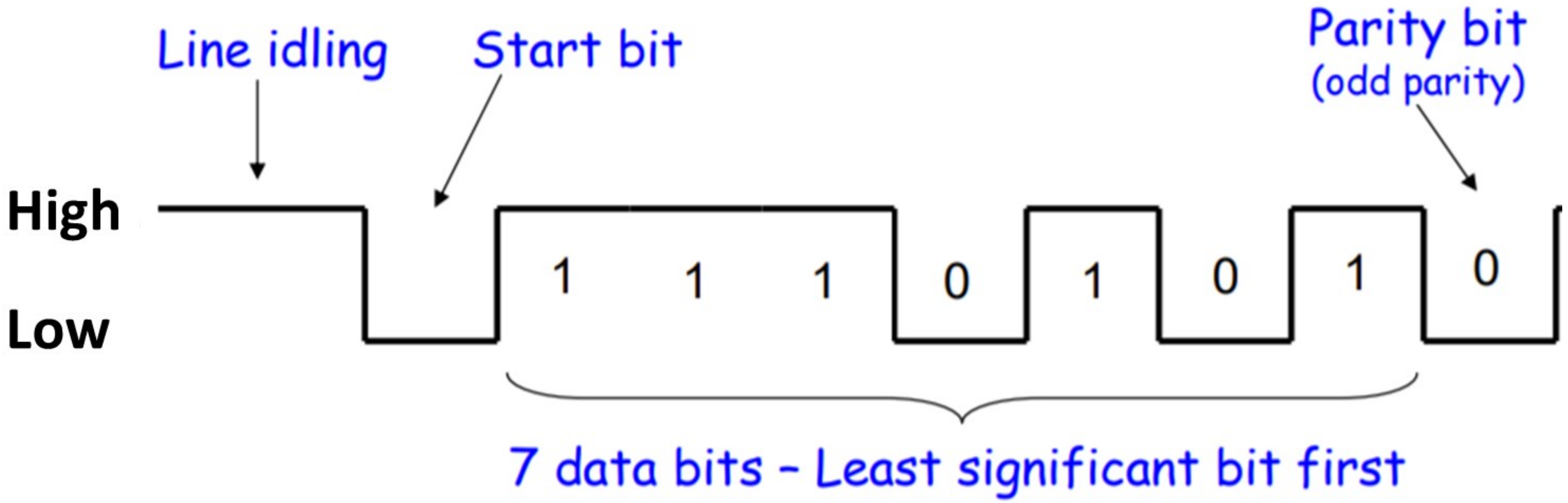
UART Transmission Example

Send the ASCII letter ‘W’ (1010111)



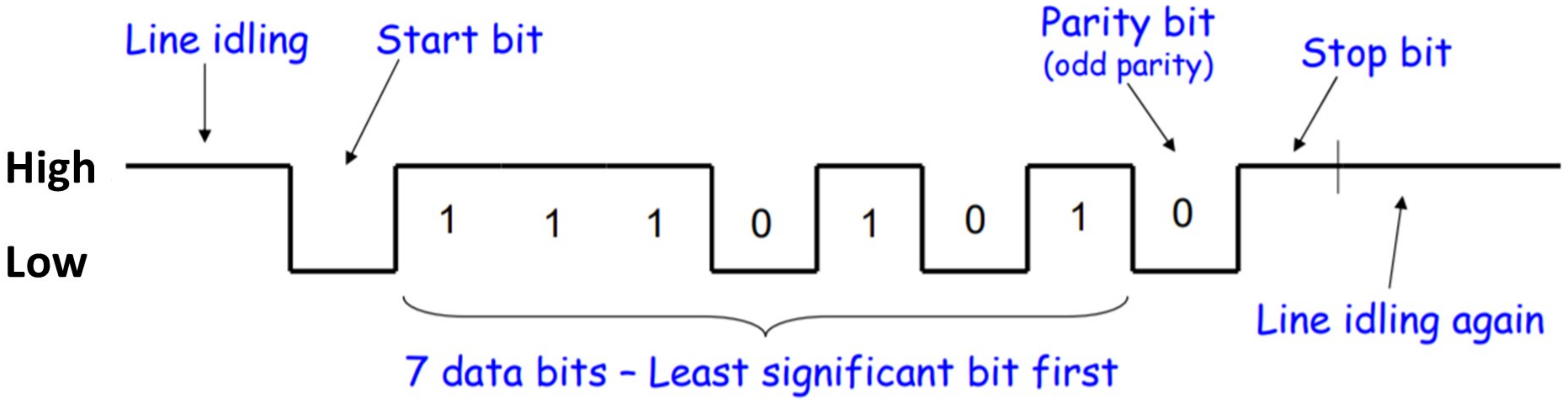
UART Transmission Example

Send the ASCII letter ‘W’ (1010111)



UART Transmission Example

Send the ASCII letter ‘W’ (1010111)



Device Interfaces

- ATA host adapters
  - intelligent drive electronics (hard drive, CDROM)
- SATA (Serial ATA)
  - inexpensive, fast, bidirectional
- FireWire
  - high speed (800 MB/sec), many devices at once
- Bluetooth
  - small amounts of data, short distances, low power usage
- Wi-Fi (wireless Ethernet)
  - IEEE 802.11 standard, faster than Bluetooth

What's Next

- General Concepts
- IA-32 Processor Architecture
- IA-32 Memory Management
- Components of an IA-32 Microcomputer

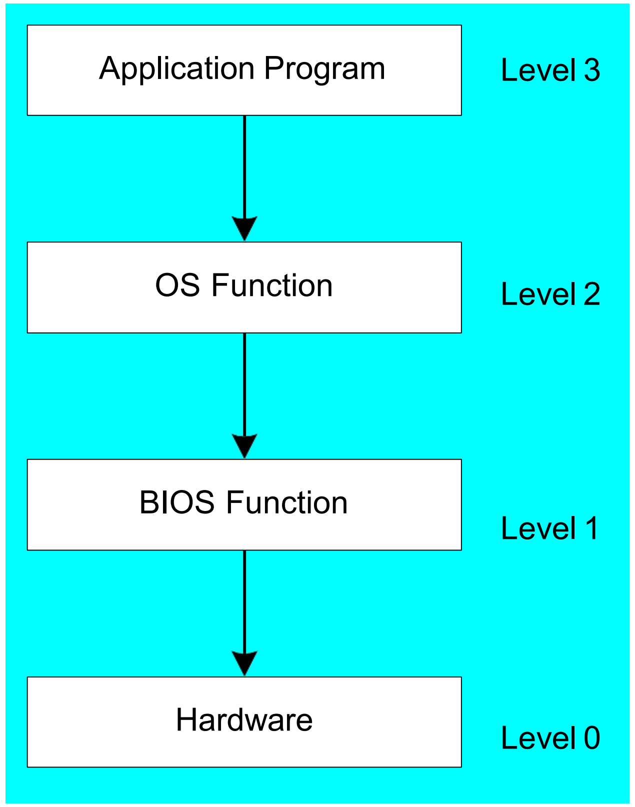
Input-Output System

Levels of Input-Output

- Level 3: High-level language function
  - examples: C++, Java
  - portable, convenient, not always the fastest
- Level 2: Operating system
  - Application Programming Interface (API)
  - extended capabilities, lots of details to master
- Level 1: BIOS
  - drivers that communicate directly with devices
  - OS security may prevent application-level code from working at this level

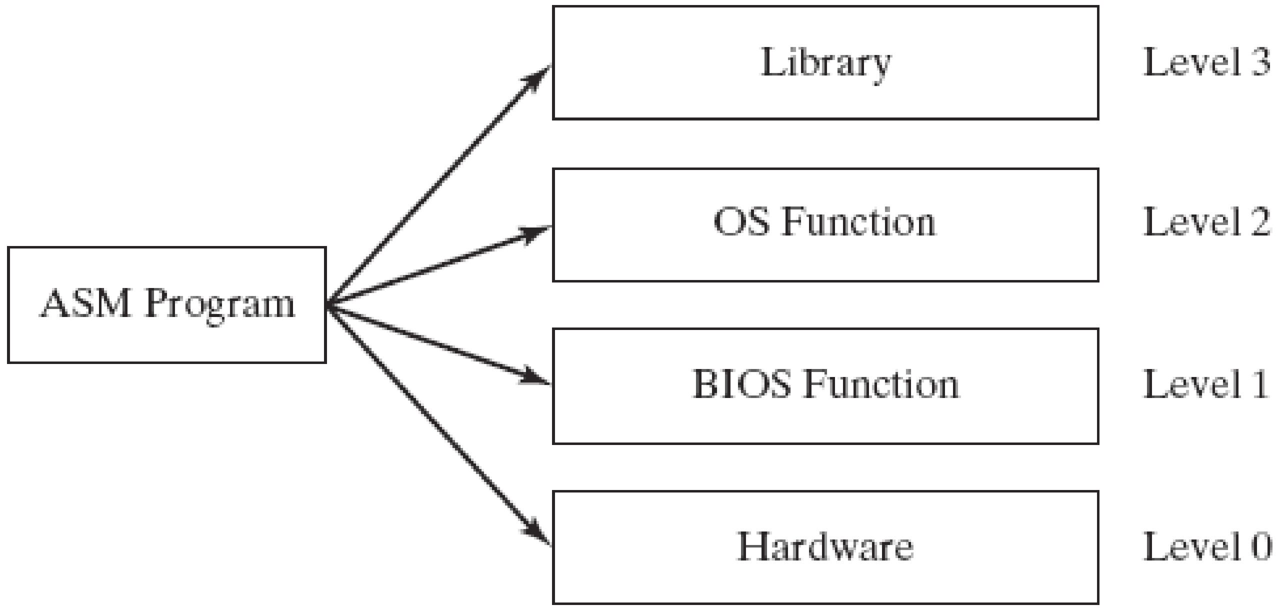
Displaying a String of Characters

When a HLL program displays a string of characters, the following steps take place:



Programming levels

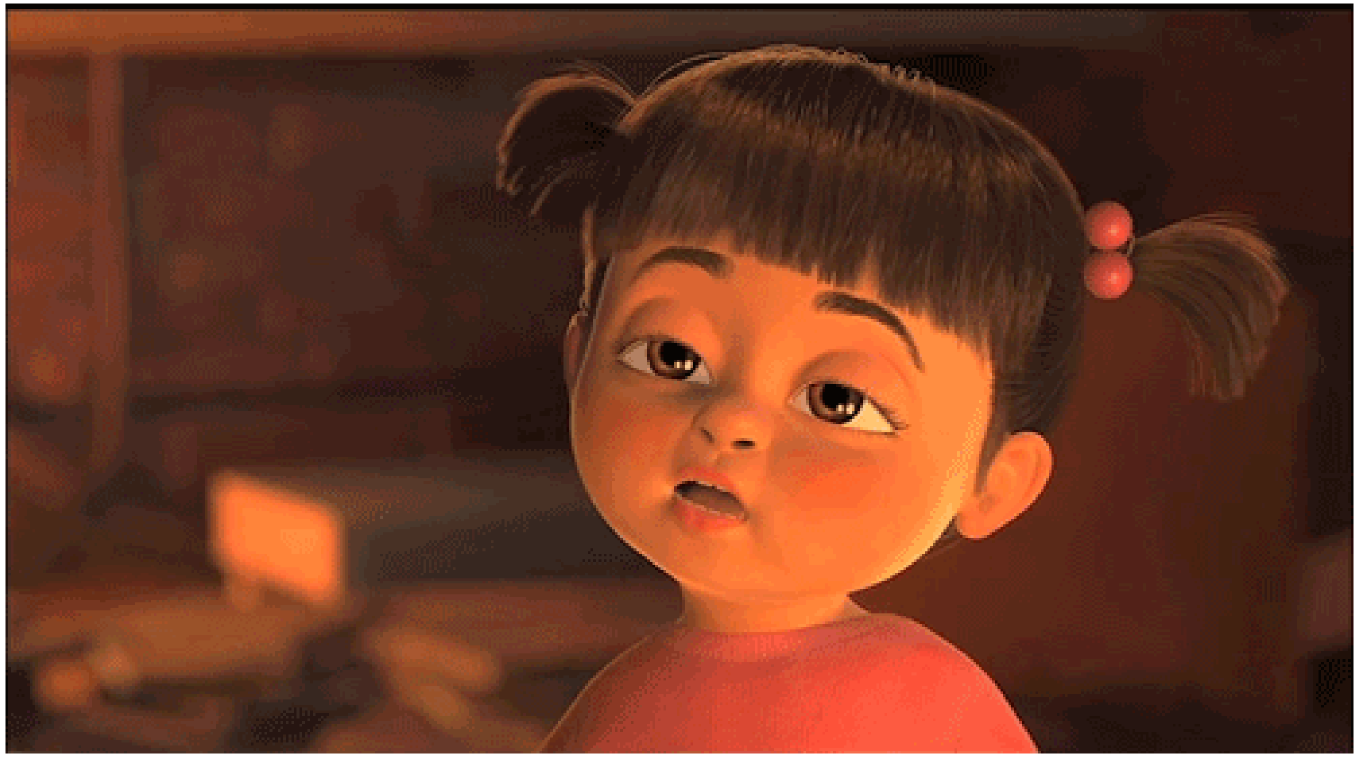
Assembly language programs can perform input-output at each of the following levels:



Summary

- Central Processing Unit (CPU)
- Arithmetic Logic Unit (ALU)
- Instruction execution cycle
- Multitasking
- Floating Point Unit (FPU)
- Complex Instruction Set
- Real mode and Protected mode
- Motherboard components
- Memory types
- Input/Output and access levels

Thanks a lot



If you are taking a Nap, **wake up**........Lecture Over