Chapter 1

Week 01 – Lecture

Chapter 1.1-1.4 & 1.8, C

Team

- Instructor
  - Giancarlo Succi
- Lab Instructors
  - Shokhista Ergasheva
  - Artem Kruglov
  - Nikita Lozhnikov (also Tutorial instructor)
  - Xavier Vasquez

Sources

- These slides have been adapted from the original slides of the adopted book:
  - Tanenbaum & Bos, ModernOperating Systems: 4th edition, 2013, Prentice-Hall, Inc
- and customized for the needs of this course
- Additional input for the slides are detailed later

Outline

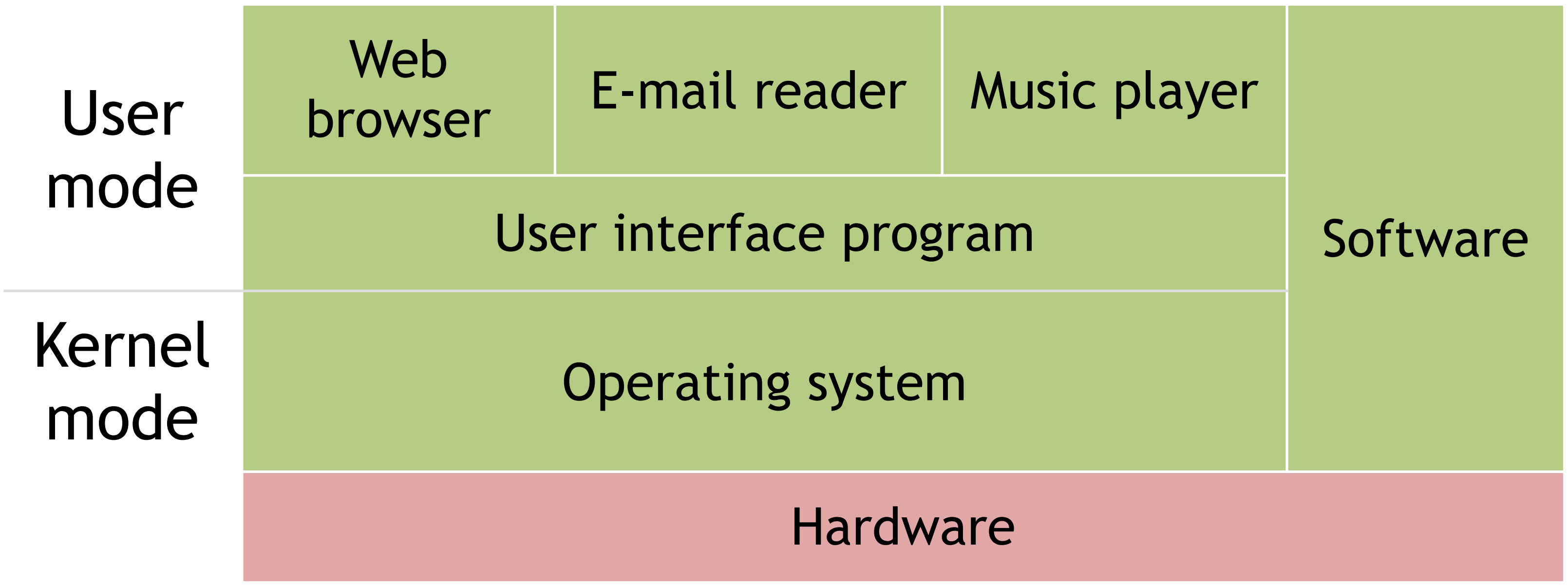
- What is OS
- Kernel mode & User mode
- OS as Extended Machine and Resource Manager
- History of OS (1st - 5th Generations)
- Computer Hardware Review (Processors, Memory, Disks, I/O, Buses)
- Booting the Computer
- OS Zoo

What is an Operating System?

- The OS is a layer of software that:
  - Provides user with a better, simpler, cleaner model
  - Handles managing all the resources

Kernel mode & User mode

- Most computers have two modes of operation: kernel mode and user mode



Kernel mode

- The operating system runs in kernel mode (also called supervisor mode):
  - has complete access to all the hardware
  - can execute any instruction the machine is capable of executing

User mode

- The rest of the software runs in user mode:
  - only a subset of the machine instructions is available
  - those instructions that affect control of the machine or do
- I/O (Input/Output) are forbidden to user-mode programs. To obtain services from the OS, a user program must make a system call (TRAP), which traps into the kernel and invokes the OS

The OS as an Extended Machine

- **Top-down view**
  - **Problem:**
    - hardware is very complicated and present difficult and inconsistent interfaces to the software developers
  - **Solution:**
    - use abstractions, for instance, a disk driver, that deals with the hardware and provides an interface to read and write disk blocks

The OS as a Resource Manager (1/2)

- **Bottom-up view**
  - Manage all the pieces of a complex system
  - The job of the operating system is to provide for an orderly and controlled allocation of the processors, memories, and I/O devices among the various programs wanting them

The OS as a Resource Manager (2/2)

- Multiplexing (sharing) resources can be done in two ways:
  - **Time multiplexing**
    - different programs or users take turns using it (example: CPU, printers)
  - **Space multiplexing**
    - instead of the customers taking turns, each one gets part of the resource (example: memory, disks)

1945-1955 - The 1st Generation. Vacuum tubes, plug boards.

- Colossus by Alan Turing
- ENIAC by William Mauchley
  - No OS
  - Slow & Huge
  - Large air conditioner
  - High electricity consumption
  - Limited storage capacity
  - Programmed in mechanical language
- Punch cards & paper tapes to feed programs. Data to get results
- Magnetic tapes & Magnetic drums for secondary memory

1955-1965 - The 2nd Generation

- Transistors and Batch Systems / Mainframes
  - Transistors
  - Core Memory - developed
  - Faster than the 1st generation
  - First Operating System
  - Programming was in Machine Language & Assembly Language
  - Magnetic tapes & discs were used
  - Computers became smaller than the 1st generation
  - Computers consumed less heat & electricity

1965-1980 - The 3rd Generation

- Transistors and Batch Systems / Mainframes
  - Integrated Circuits and Multiprogramming
  - Multiprogramming
  - Spooling
  - Timesharing

- Examples:
  - IBM System/360

1965-1971 - The 3rd Generation. 1 Part.

- Part 1. Integrated circuits
  - Integrated circuits developed
  - Power consumption was low
  - SSI & MSI Technology was used
  - High level languages were used
- Example:
  - Honeywell-6000 series
  - PDP(Personal Data Processor)
  - IBM-360 and 370/168 series
  - TDC-316

1971-1980 - The 3rd Generation. 2 Part.

- Part 2. Microprocessors
  - LSI & VLSI Technology used
  - Development of Portable Computers
  - RAID Technology of data storage
  - Used in virtual reality, multimedia, simulation
  - Computers started in use for Data Communication
  - Different types of memories with very high accessing speed & storage capacity
- Used in: parallel processing, superconductors, speech recognition, intelligent robots, artificial intelligence

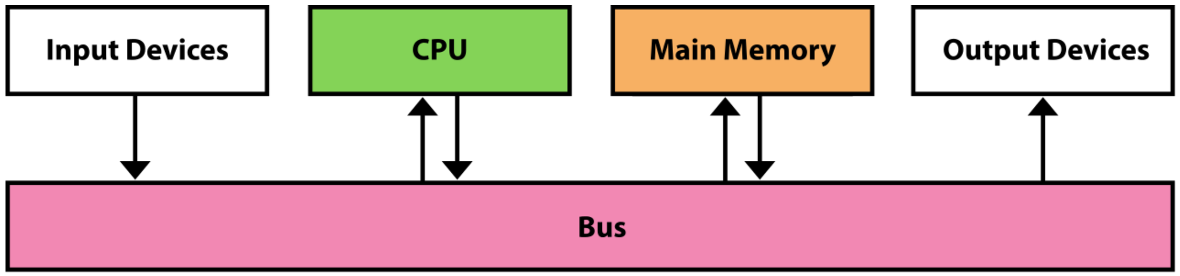
1980-1990 - The 4th Generation. Personal Computers

- Personal computers were developed after LSI (Large Scale Integration) circuits were invented.
- First Microcomputer:
  - Intel 8080 CPU + attached 8-inch floppy disk
  - First disk based OS CP/M (Control Program for Microcomputers)
- In 1980s IBM designed the IBM PC and contacted Bill Gates for an operating System

1990- Present - The 5th Generation. Mobile Computers

Computer Hardware

- An OS is tied to the hardware of the computer it runs on. It extends the computer's instruction set and manages its resources
  - Processors
  - Main memory
  - Disks
  - I/O devices
  - Buses



Processors (1/7)

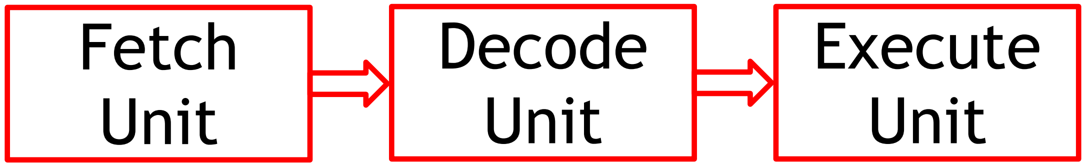
- The basic cycle of every CPU:
  - **Fetch Instruction** - read next expected instruction into buffer
  - **Decode Instruction** - determine opcode & operand specifiers
  - **Calculate Operands** - calculate the effective address of each source operand
  - **Fetch Operands** - fetch each operand from memory. Operands in registers need not be fetched
  - **Execute Instruction** - perform the indicated operation and store the result
  - **Write Operand** - store the result in memory

Processors (2/7)

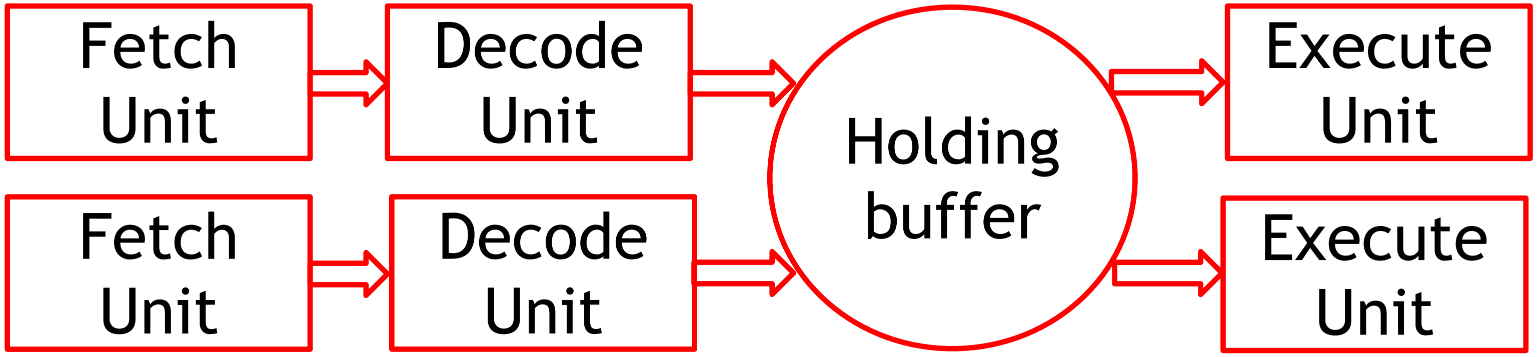
- CPU contains some registers inside to hold key variables and temporary results:
  - **General registers** - hold variables and temporary results
  - **Program counter** - contains the address of next instruction to be fetched
  - **Stack pointer** - points to the current stack in memory
  - **PSW (Program Status Word)** - bits of results of comparison instructions, the CPU priority, the mode (kernel or user) and various other control bits

Processors (3/7)

a) A three-stage pipeline (Figure 1.7)



b) A superscalar CPU (Figure 1.7)

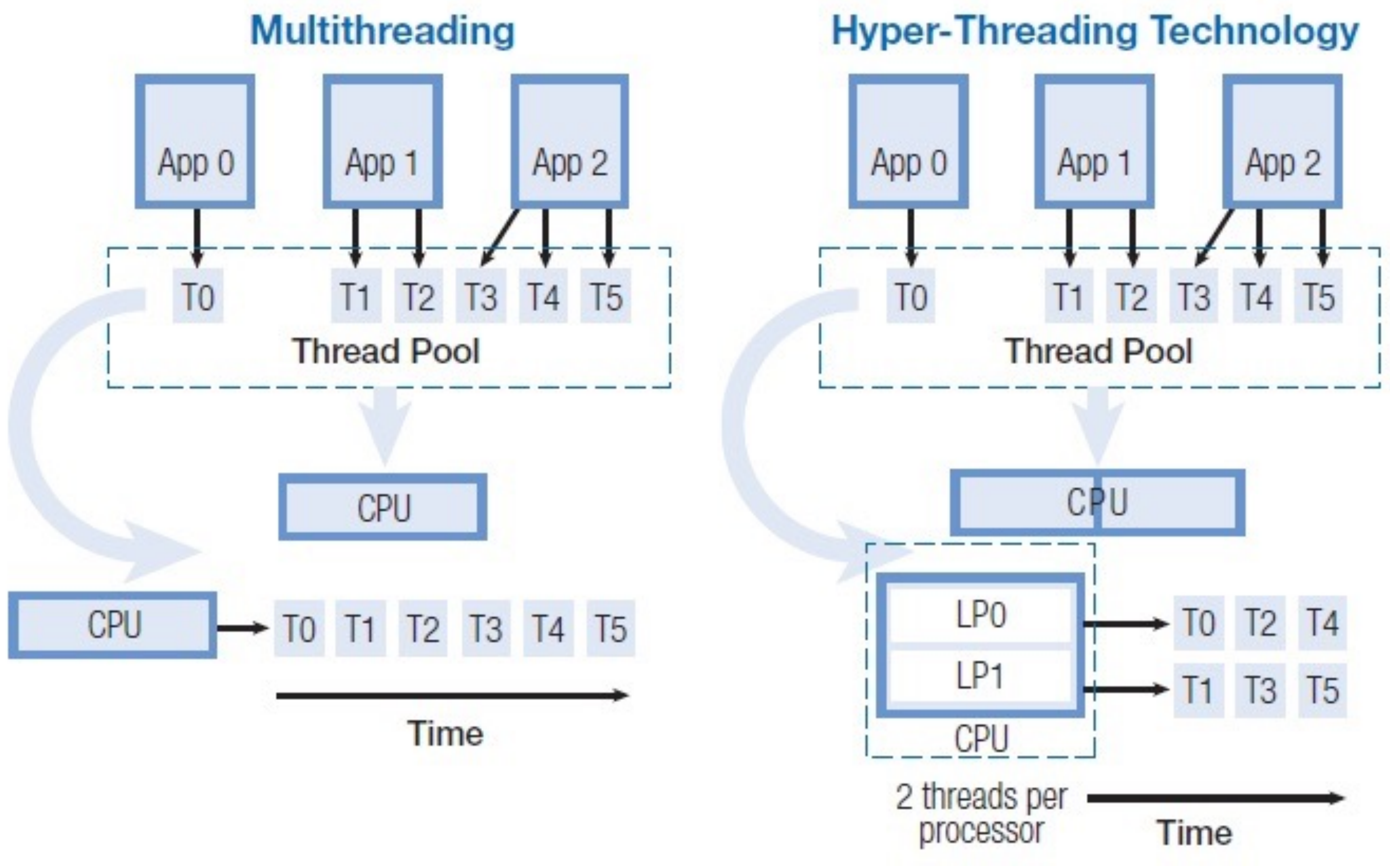


Processors (4/7) - Multithreading

- Multithreading or hyper-threading
  - It allows the CPU to hold the state of two different threads and then switch between them in nanoseconds
  - If one of the processes needs to read a word from memory (long operation), a multithreaded CPU can just switch to another thread thus saving time
  - Multithreading does not offer true parallelism

Processors (5/7) - Multithreading

Multithreading or hyper-threading



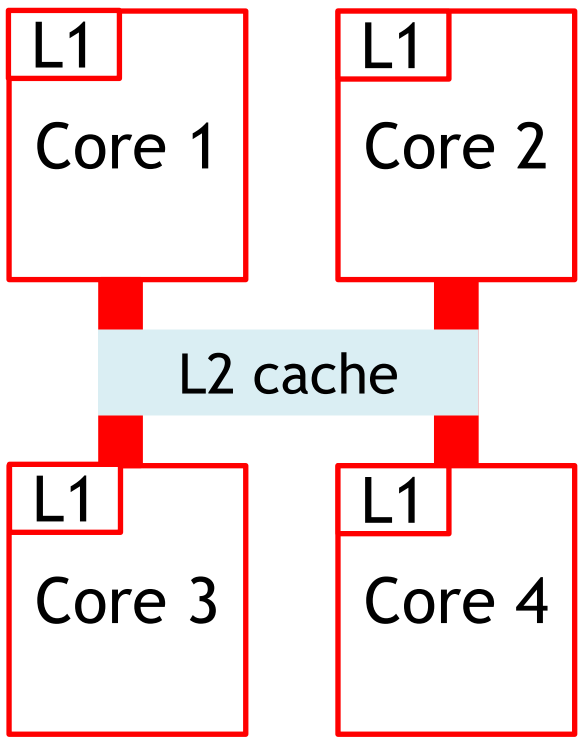
Processors (6/7) - Multithreading

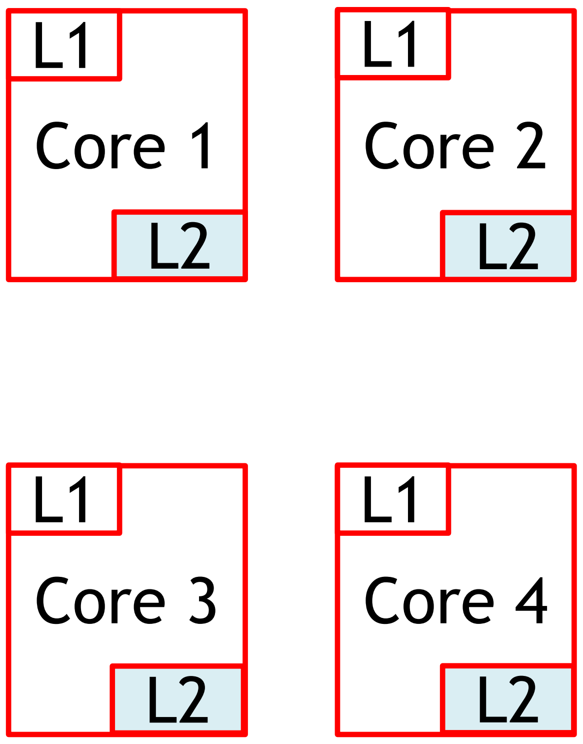
- Many CPU chips now have some complete processors or cores. Making use of such a multicore chip will require a multiprocessor OS
  - A modern GPU (Graphics Processing Unit) is a processor with thousands of tiny cores which are very good for many small computations done in parallel, like rendering polygons in graphics applications
  - Multicore chips do true parallelism

Processors (7/7) - Multicore Chips

a) A quad-core chip with a shared L2 cache

b) A quad-core chip with separate L2 caches

a)


b)


Memory (1/7)

- The memory system is constructed as a hierarchy of layers. The top layers have higher speed, smaller capacity, and greater cost per bit than the lower ones

Typical access time		Typical capacity
1 nsec	Registers	<1 KB
2 nsec	Cache	4 MB
10 nsec	Main memory	1-8 GB
10 msec	Magnetic disk	1-4 TB

Memory (2/7)

- Main memory is divided up into cache lines, typically 64 bytes. The most heavily used cache lines are kept in a high-speed cache located inside the CPU

Memory (3/7)

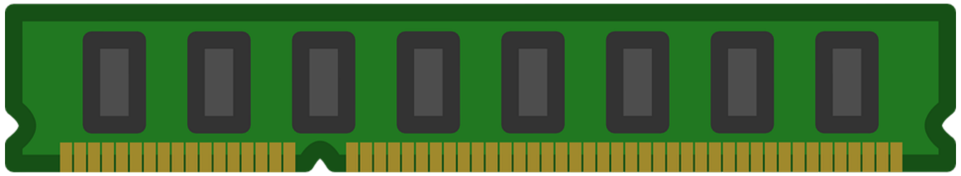
- When the program needs to read a memory word, the cache hardware checks to see if the line needed is in the cache. If it is (a cache hit), no memory request is sent over the bus to the main memory. It normally takes about 2 clock cycles
- Some machines have two or even three levels of cache, each one slower and bigger than the one before it

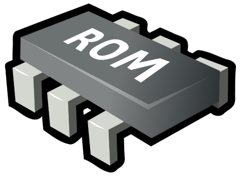
Memory (4/7)

- Caching system issues:
  - When to put a new item into the cache
  - Which cache line to put the new item in
  - Which item to remove from the cache when a slot is needed
  - Where to put a newly evicted item in the larger memory

Memory (5/7) - Caching

- RAM (Random-Access Memory) serves all CPU requests that cannot be satisfied out of the cache





- ROM (Read-Only Memory) does not lose its contents when the power is switched off, is programmed at the factory and cannot be changed afterward. On some computers, the bootstrap loader used to start the computer is contained in ROM

Memory (6/7) - Main Memory

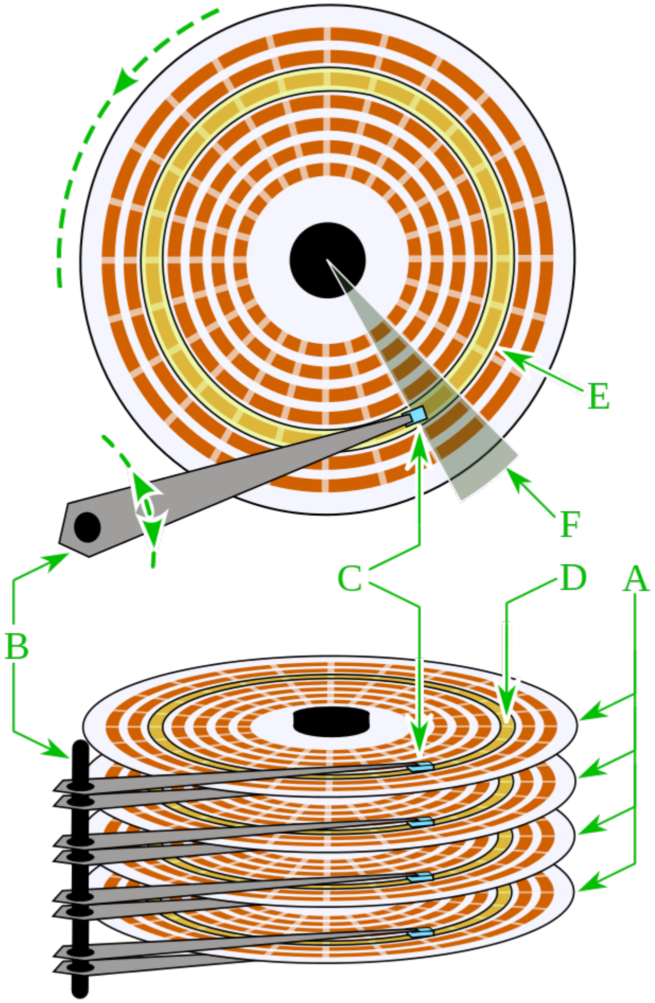
- EEPROM (Electrically Erasable Programmable ROM) and flash memory are also nonvolatile, but in contrast to ROM can be erased and rewritten
- Flash memory is also commonly used as the storage medium in portable electronic devices. It is intermediate in speed between RAM and disk and it wears out

Memory (7/7) - Main Memory

- Many computers use complementary metal-oxide-semiconductor (CMOS) memory to hold the current time and date and the configuration parameters, such as which disk to boot from

Disks (1/2)

- Multi-platter HDD
  A. Platters
  B. Actuator arm
  C. Read/write head
  D. Segment –
     an arc on the track
  E. Track
  F. Sector



Disks (2/2)

- Time to read/write from disk consists of
  - **Seek time** is a physical positioning of read/ write head
  - **Rotation delay (latency)** is the amount of time it takes for the disc to rotate to the required position for the read/write head
  - **Transfer time (data rate)** is the amount of time it takes for data to be read or written

I/O Devices (1/7)

- Generally consist of two parts:
  - a controller that accepts commands from the operating system and carries them out
  - a device itself that has fairly simple interfaces, both because they cannot do much and to make them standard

I/O Devices (2/7)

- Device driver is a software that talks to a controller, giving it commands and accepting responses. It has to be put into the OS so it can run in kernel mode
- Every controller has a small number of registers that are used to communicate with it. The collection of all the device registers forms the **I/O port space**
- On some computers, the device registers are mapped into the operating system's address space (the addresses it can use), so they can be read and written like ordinary memory words

I/O Devices (3/7)

- On other computers, the device registers are put in a special I/O port space, with each register having a port address. On these machines, special IN and OUT instructions are available in kernel mode to allow drivers to read and write the registers
- Input and output can be done in three different ways:
  - **Busy waiting:**
    - the driver starts the I/O and sits in a tight loop continuously polling the device to see if it is done. It has the disadvantage of tying up the CPU polling the device until it is finished

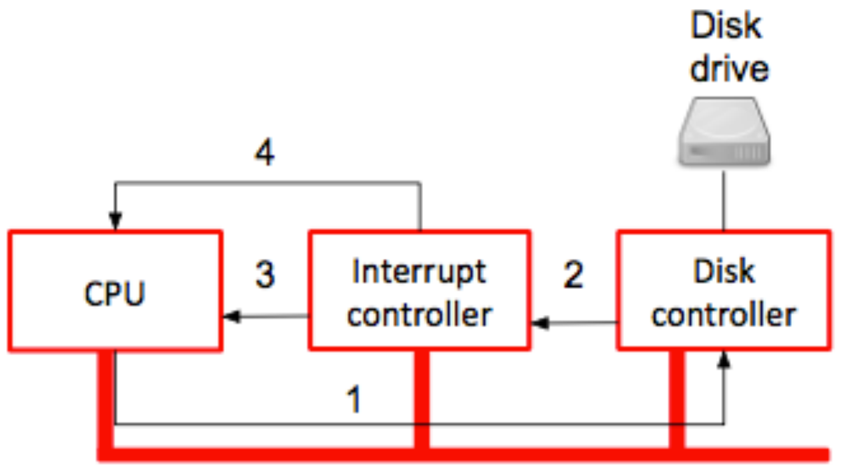
I/O Devices (4/7) - Busy Waiting

- **Interrupt:**
  - the driver starts the device and asks it to give an interrupt when it is finished. The operating system then blocks the caller if need be and looks for other work to do
  - When the controller detects the end of the transfer, it generates an interrupt to signal completion. The device number may be used as an index into part of memory to find the address of the interrupt handler for this device. This part of memory is called **the interrupt vector**

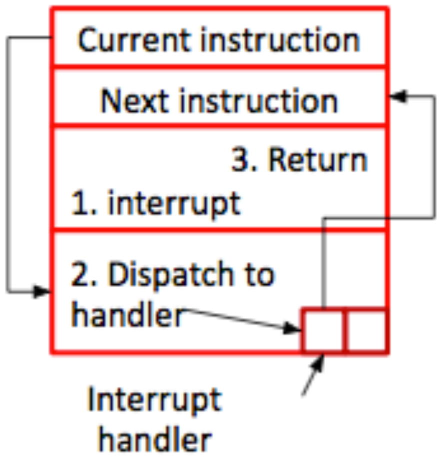
I/O Devices (5/7) - Interrupts

a) The steps in starting an I/O device and getting an interrupt
b) Interrupt processing involves taking the interrupt, running the interrupt handler, and returning to the user program

a)



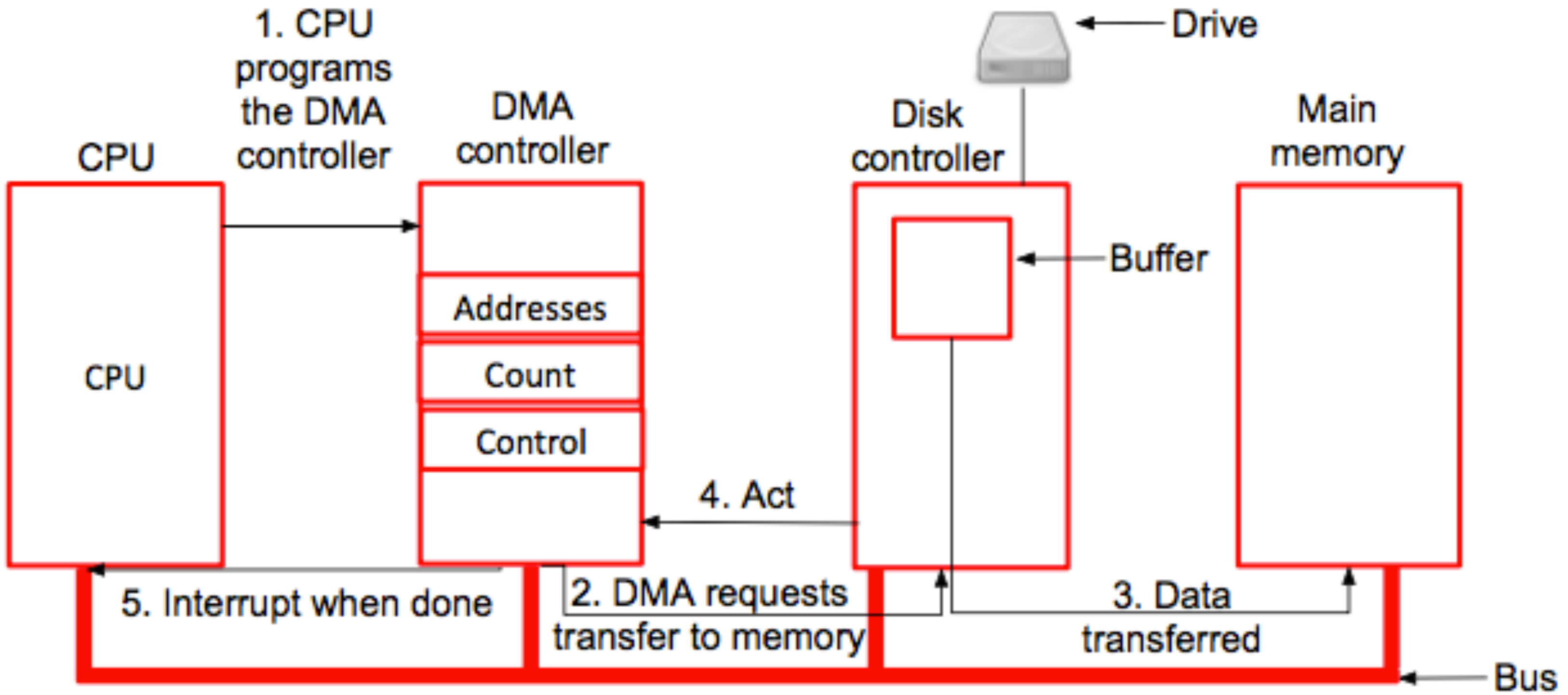
b)



I/O Devices (6/7) - DMA

- **Direct Memory Access (DMA):**
  - DMA is a chip that can control the flow of bits between memory and some controller without constant CPU intervention
  - The CPU programs the DMA chip, telling it what and where to transfer and lets it go. When the DMA chip is done, it causes an interrupt, which is handled as described above

I/O Devices (7/7) - DMA

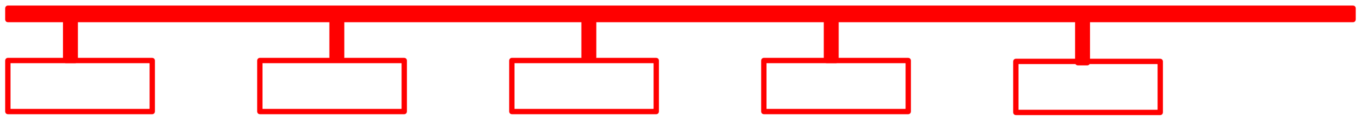


Buses (1/5)

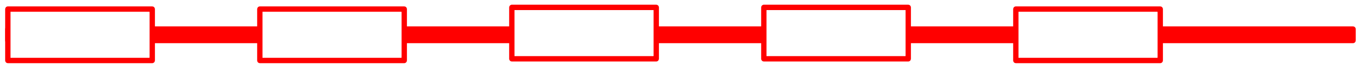
- The system has many buses, each with a different transfer rate and function. The OS must be aware of all of them for configuration and management.
- A shared bus architecture means that multiple devices use the same wires to transfer data which needs an arbiter to determine who can use the bus.

Buses (2/5)

- A parallel bus architecture means that you send each word of data over multiple wires. For instance, in regular PCI buses, a single 32-bit number is sent over 32 parallel wires.



- A serial bus architecture sends all bits in a message through a single connection, known as a lane. Parallelism is still used, because you can have multiple lanes in parallel (send 32 messages via 32 lanes).

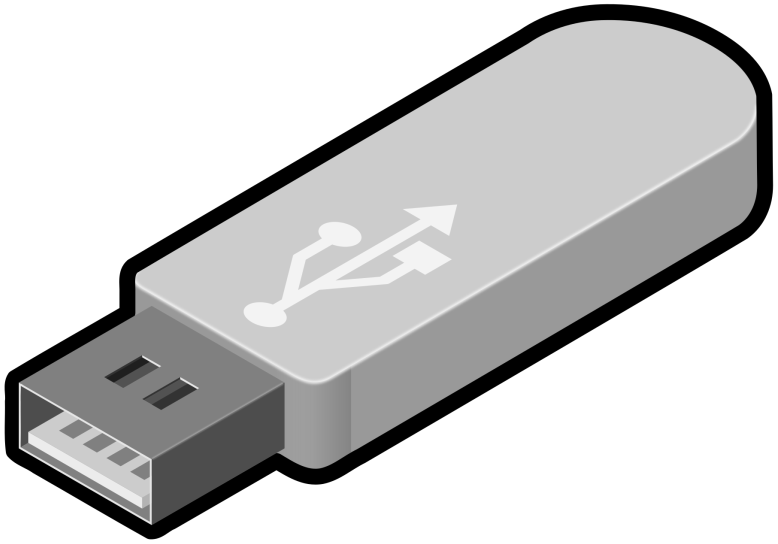


Buses (3/5)

- **Double Data Rate (DDR3, DDR4)** connects CPU and RAM
- **Peripheral Component Interconnect (PCIe)** is a bus to an external graphical device
- **Direct Media Interface (DMI)** is a link between north bridge and south bridge - a hub for all the other devices
- **SCSI, SATA** - connects hard disks

Buses (4/5)

- **Universal Serial Bus (USB)** is a centralized bus in which a root device polls all the I/O devices to see if they have any traffic.



Buses (5/5) - Plug & Play

- Before **plug and play**, each I/O card had a fixed interrupt request level and fixed addresses for its I/O registers. And two different pieces of hardware might use the same interrupt, so they will conflict
- **Plug and play** makes the system automatically collect information about the I/O devices, centrally assign interrupt levels and I/O addresses, and then tell each card what its numbers are

Booting the Computer (1/5)

- **Basic Input Output System (BIOS)** is a program on the parent board that contains low-level I/O software.
- After the BIOS is started it performs **Power-On Self-Test (POST)** to test integrity and see how much RAM is installed and other basic devices are installed and responding correctly.

Booting the Computer (2/5)

- It starts out by scanning the buses to detect all the devices attached to them. Then it determines the boot device by trying a list of devices stored in the CMOS memory
- CMOS (Complementary metal–oxide–semiconductor) is a technology for constructing integrated circuits

Booting the Computer (3/5)

- The boot sector (first sector from the boot device) is read into memory and executed. This sector contains a program that normally examines the partition table at the end of the boot sector to determine which partition is active

Booting the Computer (4/5)

- **A secondary boot loader** is read in from that partition. This loader reads in the OS from the active partition and starts it. The OS then queries the BIOS to get the configuration information. For each device, it checks to see if it has the device driver.

Booting the Computer (5/5)

- Once it has all the device drivers, the OS loads them into the kernel. Then it initializes its tables, creates whatever background processes are needed, and starts up a login program or GUI

The OS Zoo (1/2)

- Mainframe Operating Systems
  - Example: OS/390, OS/360
- Server Operating Systems
  - Example: UNIX, Windows 2000, Linux
- Multiprocessor Operating Systems
- Personal Computer Operating Systems
  - Examples: Windows 98, XP, Mac OS, Linux

The OS Zoo (2/2)

- Handheld Computer Operating Systems
- Embedded Operating Systems
- Sensor Node Operating Systems
- Real-Time Operating Systems
- Smart Card Operating Systems

Mainframe Operating Systems

- **Mainframes** are room-sized computers in data centers. The operating systems for mainframes are heavily oriented toward processing many jobs at once, most of which need prodigious amounts of I/O

Mainframe Operating Systems

- Typically kinds of service:
  - A **batch** system processes routine jobs without any interactive user present
  - **Transaction-processing** systems handle large numbers of small requests
  - **Timesharing** systems allow multiple remote users to run jobs on the computer at once, such as querying a big database

Server Operating System

- They run on servers, which are either very large personal computers, workstations, or even mainframes. They serve multiple users at once over a network and allow the users to share hardware and software resources.
- Examples: Solaris, FreeBSD, Linux, Windows Server.

Multiprocessor Operating Systems

- Connect multiple CPUs in a single system:
  - parallel computers
  - multicomputers
  - multiprocessors
- Nowadays all PC and notebooks OS deal with small-scale multiprocessors
  - Examples: Linux, Windows

Personal Computer Operating Systems

- You know what is it.
  - Examples: FreeBSD, Linux, Mac OS X, Windows

Handheld Computer Operating Systems

- OS for PDAs earlier and Mobile Devices nowadays
  - Example: Google Android, Apple iOS

Embedded Operating Systems

- Run on the computers that control devices that are not generally thought of as computers and which do not accept user-installed software: microwave ovens, MP3 players, TV sets, cars, etc.
  - Examples: Embedded Linux, QNX, VxWorks

Sensor-Node Operating Systems

- Each sensor node is a real computer, with a CPU, RAM, ROM, and one or more environmental sensors
- These nodes communicate with each other and with a base station using wireless communication
  - Example: TinyOS

Real-time Operating Systems

- **A hard real-time system** must provide absolute guarantees that a certain action will occur by a certain time.
- **A soft real-time system** is one where missing an occasional deadline, while not desirable, is acceptable and does not cause any permanent damage.
- Example: QNX

Smart Card Operating Systems

- The smallest operating systems run on smart cards, which are credit-card-sized devices containing a CPU chip
- **Java oriented smart cards** - means that the ROM on the smart card holds an interpreter for the Java Virtual Machine (JVM). Java applets (small programs) are downloaded to the card interpreted by the JVM interpreter

End

Week 01 – Lecture

References (1/3)

- Tanenbaum & Bos, ModernOperating Systems: 4th edition, 2013, Prentice-Hall, Inc.

References (2/3)

- http://www.slideshare.net/chukidadiz/evolution-of-programming-languages-24938001
- http://www.digibarn.com/collections/posters/tongues/
- http://www.quicklycode.com/infographics_posters/the-history-of-programming-languages-poster
- https://www.youtube.com/watch?v=bGk9W65vXNA
- https://prezi.com/0ifwtcl48iy_/first-generation-1940-1956-vacuum-tubes/
- http://www.slideshare.net/akiladj/2-history-the-generations-of-computer-history
- http://thapaprabin.blogspot.ru/2012/08/generation-of-computer.html
- https://zuber01.wordpress.com/2012/06/30/computer-generation/
- http://slideplayer.com/slide/6671114/
- https://en.wikipedia.org/wiki/Instruction_pipelining
- https://en.wikibooks.org/wiki/IB/Group_4/Computer_Science/Computer_Organisation

References (3/3)

- http://www.tutorialspoint.com/operating_system/os_io_hardware.htm
- http://www.karbosguide.com/books/pcarchitecture/chapter01.htm
- http://pcunstructuredinfo.blogspot.ru/2015/05/evolution-of-intel-hyper-threading_37.html
- http://www.clker.com/clipart-14807.html
- https://en.wikipedia.org/wiki/Random-access_memory
- https://www.youtube.com/watch?v=C_NbeOaUtog
- http://jules.dourlens.com/using-eeprom-library-for-long-term-storage-on-arduino/
- https://commons.wikimedia.org/wiki/File:Basic_disk_displaying_CHS.svg