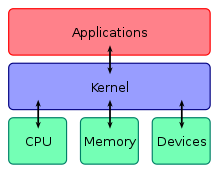
Programming Languages Lesson 1

IS-2053-007, Spring 2022

**What computers were made for:**

* Mathematical problem solving
* ENIAC
* Processors
* Microprocessors

**Software:**

* Programs/Applications
* Scripts
* Processes
* Threads
* Device Drivers
* OS’s
* Kernel
  + A blue circle with black text

    Description automatically generated with low confidence

**Hardware:**

* Input Devices
  + Keyboard
  + Mouse
  + Microphone
  + Touch screen
  + Joystick
  + Camera
  + Keypad (ATM)
  + Controller
  + Button
  + Drawing tablet (WACOM)
  + Card Reader
* Output devices
  + Monitor
  + Printer
  + Speakers
  + TV
  + Headphones
  + Smart Fridge / IoT
  + GPS
* Dual devices
  + USB drives (can upload and download, so to speak)
  + CD/DVD’s
  + Camera
  + Modem
* CPU
* RAM cards
* GPU
* Motherboard
  + Buses
  + Flip-flops
  + Gates
  + NIC
  + Wireless NIC
  + Ethernet Port

**Memory**

* Main
  + RAM
    - Volatile
    - Copies/loads the program into RAM
* Cache
  + L2
  + L3
* Secondary Storage
  + HDD
  + USB
  + CD/DVD
  + Floppy disk (almost obsolete)
* Servers

**Numbering Systems:**

* Base 10 system
  + 10,356 = (10 \* 103 ) + (3 \* 102 ) + (5 \* 101 ) + (6 \* 100 )
  + Remember, all integers can be expressed as fractions
    - 1 = 1/1
    - 5 = 5/1
  + Rational and irrational numbers are essentially fractions, a portion of a whole amount.
* Binary system (Base 2)
  + Bits = 1 or 0. Nothing else. That’s it.
    - Smallest writeable value in a computer
  + 4 bits = a nybble/nibble
  + 8 bits = 1 byte
    - 27 = 128
    - 28  = 256
    - 512
    - 1024
    - 2048
    - 4096
    - Etc…..
  + 1024 bytes = 1KB
  + 1024 kilobytes = 1 MB
  + 1024 MB = 1GB
* A sector = 512 bytes. This is the smallest storage section of a HDD.
  + How many bits is this?
    - Answer = 4096 bits
* Let’s do a calculation and convert the following byte:
  + 1011 0111 Big Endian
  + (1 \* 27 )+ (0 \* 26 )+ (1 \* 25)+ (1 \* 24 )+ (0 \* 23 )+ (1 \* 22 )+ (1 \* 21 ) + (1 \* 20)
  + We can also do this as: 128, 64, 32, 16, 8, 4, 2, 1 in terms of their place values (easiest way to do by hand)f. So, 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255 (not 256). 1111 1111 = 255. 0000 000**1** 0000 0000 = 256
    - Answer?
  + TWOs-Complement is used to store negative numbers and uses something called signed and unsigned values
    - We will learn about this later while programming.

**Instruction Set Architectures**

* X86\_64/32
* RISC, RISC-V
* ARM v7, v8, and earlier
* MIPS
* Sparc

**How Instructions are used:**

* All programming constructs are eventually changed into instructions that a specific CPU can execute
* ASCII capital “A” = 65 OR 01000001 in machine code (64 + 1)
* A
  + This means we need a byte to represent a single CHAR (character)
  + A WORD = 2 bytes
  + A DWORD = 4 bytes
  + A QWORD = 8 bytes
* Integers = 4 bytes, or 32 bits
  + If you need to go beyond +- 2,147,483,647, then you use the Long Int
    - This allocates the size dynamically
* More complicated instructions can occur.

**Next Lecture on Thursday:**

* How programs are fetched, decoded, and executed
* High Level Languages
* Difference between compiled and interpreted
* High vs low level languages
* Procedural and functional languages
* A quick intro into object-oriented language structures
* Python keywords
* How to use Python and Spyder