Computers

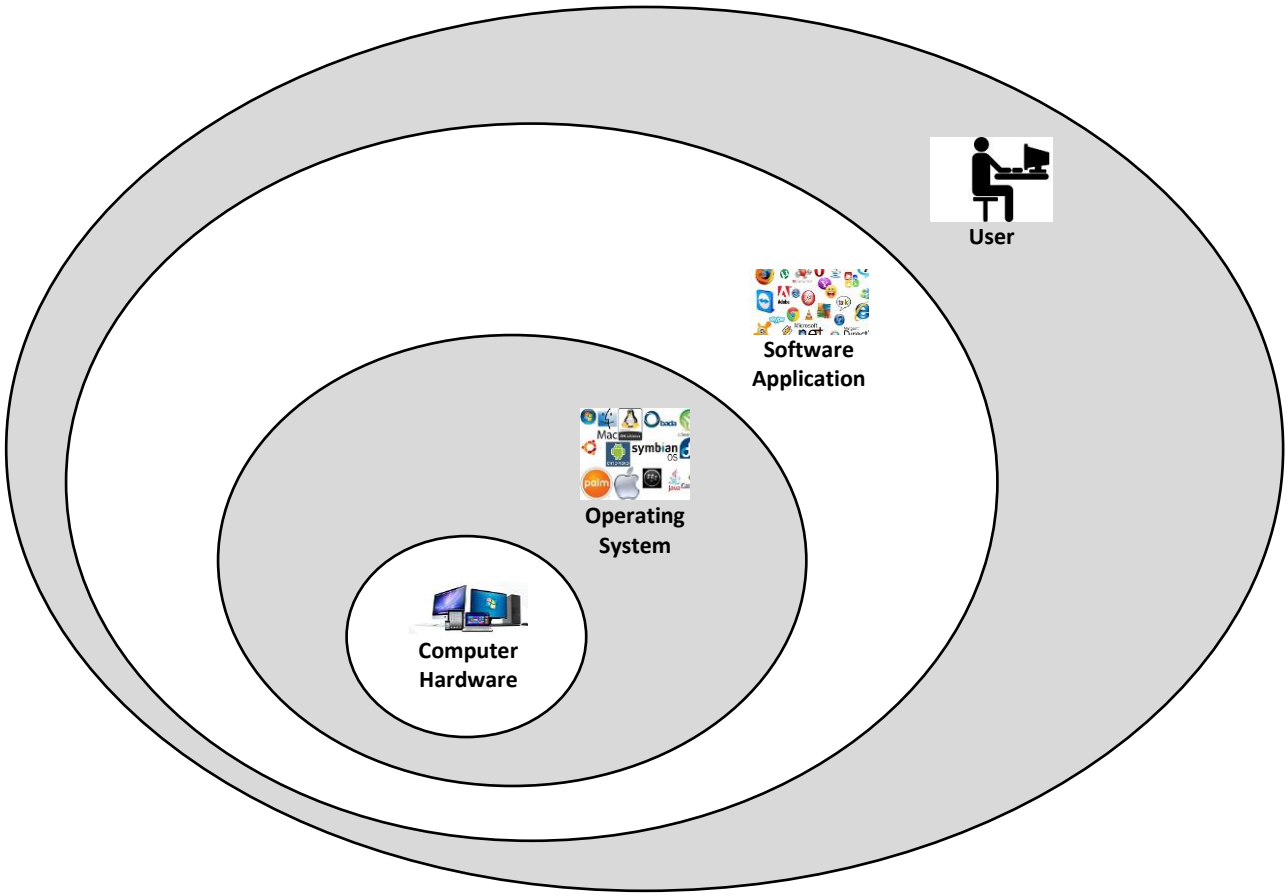
Hardware

Computer are everywhere!

- Many forms:
  - Phone
  - Laptop
  - Tablet
  - Personal computer (PC/Mac)
  - Even watches!
  - Etc…

- Require SOFTWARE to operate
  - Clearly defined set of instructions
  - “Coded” by a programmer in a computer language (many available)

Main Computer System Layers



Primary Storage (Memory)

- This is memory that is directly accessed by the central processing unit (CPU)
- Very fast

ROM

- Read Only Memory
- Manufactured with predefined instructions
    - Can be updated/changed such as firmware architecture (start-up instructions for a computer/device)
    - Some can't be updated or changed though (appliances)
- Data is "persistent" and is not lost when power is turned off

RAM

- Random Access Memory
- Holds program (application) instructions and program data
- Considered "volatile" (temporary) as this information is lost when the power is turned off

Central Processing Unit (CPU)

- The heart of a computer – executes all instructions one at a time as they are queued

Registers

- CPU internal memory

Arithmetic and Logic Unit (ALU)

- Integral (integer) comparisons and calculations
- Creates and/or changes data as instructed (CU)

Floating-Point Accelerator (FPA)

- Floating-point data (decimal) calculations
- Creates and/or changes data as instructed (CU)

Decode Unit (DU)

- Extracts and decodes instructions for the control unit (CU)

Control Unit (CU)

- Manages the data:
- Moves data to and from:
    - Registers
    - RAM
    - Other Devices
    - Passes and Decodes from/to ALU and FPA

Secondary Storage (Memory)

- Slower than primary storage/memory but more affordable
  (see "Memory Comparison" section: https://scs.senecac.on.ca/~ipc144/pages/content/probl.html)
    - Hard Disk Drives (HDD)
    - CD/DVD Disks
    - USB "Keys"/Flash Drives

Other Devices

- There are many other devices that can be attached to a computer also known as “Peripherals”

Input

- Keyboard
- Mouse
- Touch Screen
- Barcode Scanner
- Voice/Microphone
- Digital Scanner

Output

- Monitor
- Printer
- Sound/Speaker

Software

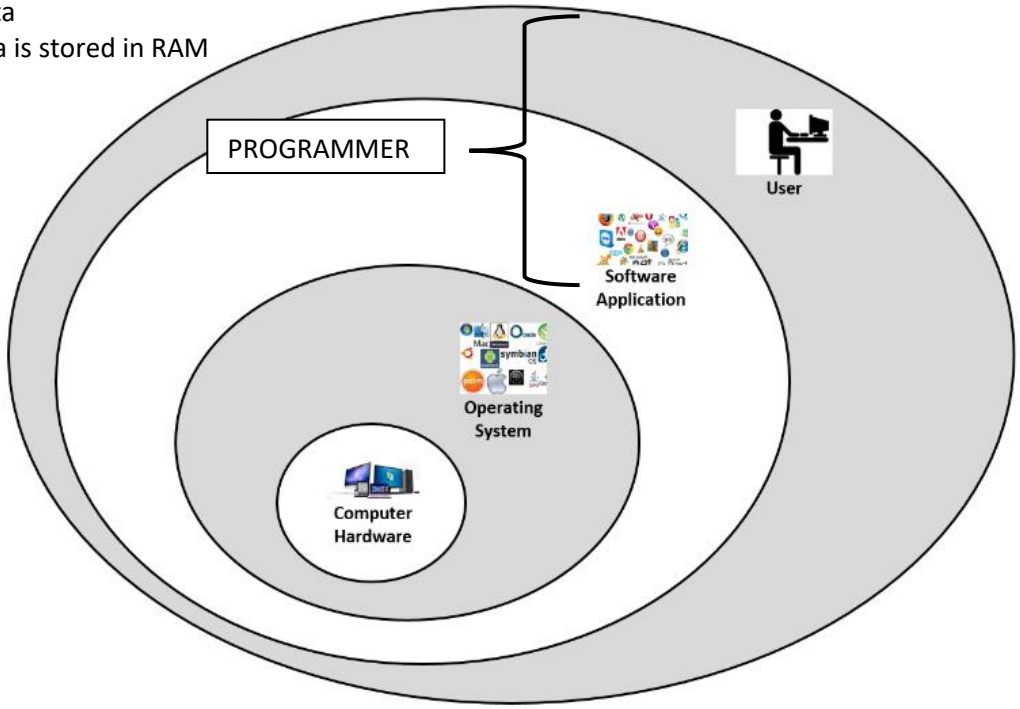
- Controls a computer as long as the power is on

Operating System

- Primary software application responsible for managing the interaction between devices, other software applications and the user
- The O/S is loaded into RAM

Application/Program

- Is given control from the operating system for execution
- The application is loaded into RAM
- Applications process data input, transforms the data in some manor and then generates some form of output (monitor, storage device, printer etc…)
- Application is developed (“Coded”) using a programming language:
    - Processes input
    - Transforms input data and/or stored data
    - Generates output data
    - Also defines how data is stored in RAM



Information

- A technical overview of how program instructions and program data is stored in RAM

Fundamental Units

Bit

- Smallest unit of data represented at the binary level ("binary digit" or "bit")
- Is either a value of 0 or 1 (on/off)

Bit

Nibble

- A larger unit conceptually referred to which represents 4 bits

Nibble			
Bit	Bit	Bit	Bit

Byte

- The fundamental addressable unit of RAM (denominations of Bytes sizes)
- Made up of 2 Nibbles or 8 bits

Byte							
Nibble				Nibble			
Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit

1 Byte = 2 Nibbles = 8 Bits

- Equivalent maximum value of one byte in decimal (base 10) is **256**
  - Equivalent to 2^8 (binary bit base 2 with 8 bits in a byte)

Byte								Decimal
Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Value
0	0	0	0	0	0	0	0	= 0
0	0	0	0	0	0	0	1	= 1
0	0	0	0	0	0	1	0	= 2
0	0	0	0	0	0	1	1	= 3
↓								
1	1	1	1	1	1	1	1	= 255

** Begins at zero not one ("Zero-based")*

HIGHEST order Bit ← **LOWEST order Bit**

Hexadecimal Numbering System

- Base-16 (each digit has a possible decimal base-10 value range from 0-15)
- 1 Byte holds 2 hexadecimal digits (each hexadecimal digit is 4 bits)
- "0x" denotes a value in hexadecimal format (the value will be prefixed with this)
- Each digit is comprised by the following: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  - "A" through "F" have a decimal (base 10) equivalent of 10 through 15

1 Byte (8 bits)								Hexadecimal
Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Value
0	0	0	0	0	0	0	0	= 0x00
0	0	0	0	0	0	0	1	= 0x01
0	0	0	0	0	0	1	0	= 0x02
0	0	0	0	0	0	1	1	= 0x03
⬇								
1	1	1	1	1	1	1	1	= 0xFF

- There are times when you need to work with Hexadecimal numbering systems and need to translate to and from binary format (more advanced topics you will cover in your 3rd semester and professional options)
  See: https://scs.senecac.on.ca/~ipc144/pages/content/datar.html

Binary to Hexadecimal Conversion

To convert a binary number to its hexadecimal equivalent, we:

1. Group the bits into nibbles,
2. Assign powers of 2 to the different bits in each nibble,
3. Multiply each bit value by the corresponding power of 2,
4. Add the products together for each nibble, and
5. Concatenate the nibble results

Consider the 8-bit number 01011100_2:

Nibble#	1				1			
Bit #	7	6	5	4	3	2	1	0
Multiplier	$(2^3)=8$	$(2^2)=4$	$(2^1)=2$	$(2^0)=1$	$(2^3)=8$	$(2^2)=4$	$(2^1)=2$	$(2^0)=1$
Binary Contents	0	1	0	1	1	1	0	0
Nibble Values	0*8 +	1*4 +	0*2 +	1*1	1*8 +	1*4 +	0*2 +	0*1
Nibble Val. Con't	0 +	4 +	0 +	1 = 5	8 +	4 +	0 +	0 = 12
Hex Nibble Value	0x5				0xC			
Hex Byte Value	0x5C							

* A=10, B=11, C=12

***Try converting 8-bit binary value:* 0 1 1 0 0 1 0 1**

Hexadecimal to Binary Conversion

To convert a hexadecimal number into its binary equivalent, we work from the lowest order bit to the highest.

General Rule

We identify the lowest order bit as the first target bit, then:

- divide by 2,
- put the remainder into the target bit,
- change the target to the next higher order bit
- Repeat the above for each bit

Consider the hexadecimal number 0x5C:

1. Identify the first lowest order target bit as bit 0
2. Divide the number (0x5C) into left and right hexadecimal digits (2 nibbles or 4 bits each)

Right Nibble

3. Take the right digit (0xC = 12_{10}), divide it by 2 and put the remainder (0) in bit 0
4. Take the result (0x6), divide it by 2 and put the remainder (0) in bit 1
5. Take the result (0x3), divide it by 2 and put the remainder (1) in bit 2
6. Take the result (0x1), divide it by 2 and put the remainder (1) in bit 3

Left Nibble

7. Take the left hexadecimal digit (0x5), divide it by 2 and put the remainder (1) in bit 4
8. Take the result (0x2), divide it by 2 and put the remainder (0) in bit 5
9. Take the result (0x1), divide it by 2 and put the remainder (1) in bit 6
10. Take the result (0x0), divide it by 2 and put the remainder (0) in bit 7

Bit #	7	6	5	4	3	2	1	0	
Hex Byte Value	**0x5C**								
Hex Nibble Value	**0x5**				**0xC**				* A=10, B=11, C=12
Divide by 2	0	0	1	2	0	1	3	6	12/2=6 (0 rem.)
Binary Bit Value	0	1	0	1	1	1	0	0	

***Try converting 1-Byte hexadecimal value:* 0x3A**

Decimal to Binary Conversion

To convert a non-negative integer into its binary equivalent, we work again from the lowest order bit to the highest.

General Rule

Create an empty bit container (start with 8) and put the integer value to convert into the lowest order bit (target) cell then:

- Divide the value by 2,
- Store the remainder in the target bit,
- Take the result as the new integer value,
- Identify the next higher-order bit our new target bit, and
- Repeat the above instructions until no value is left (may have to add more bits)

Consider the decimal value 92_{10}:

1. Identify the lowest bit order (target) bit as bit numbered 0
2. Take 92, divide it by 2 and put the remainder (0) in bit 0
3. Take the result (46), divide it by 2 and store the remainder (0) in bit 1
4. Take the result (23), divide it by 2 and store the remainder (1) in bit 2
5. Take the result (11), divide it by 2 and store the remainder (1) in bit 3
6. Take the result (5), divide it by 2 and store the remainder (1) in bit 4
7. Take the result (2), divide it by 2 and store the remainder (0) in bit 5
8. Take the result (1), divide it by 2 and store the remainder (1) in bit 6
9. Take the result (0), divide it by 2 and store the remainder (0) in bit 7

Bit #	7	6	5	4	3	2	1	0
Value Divide By 2	0	1	2	5	11	23	46	92
Bit Remainder Values	0	1	0	1	1	1	0	0

***Try converting decimal value:* 86**

Binary to Decimal Conversion

To convert a binary number into its decimal equivalent work from the lowest bit order to the highest.

General Rule

- Create an empty bit container that holds the number of bits you need to convert
- Calculate the multiplier for each bit by multiplying by the bit's corresponding power of 2
- Multiply each bit value by the multiplier and add them together

Consider the 8-bit binary number 01011100_2:

Bit #	7	6	5	4	3	2	1	0
Binary Value	0	1	0	1	1	1	0	0
Multiplier ($2^{Bit\#}$)	128	64	32	16	8	4	2	1
Formula/Bit	0*128 +	1*64 +	0*32 +	1*16 +	1*8 +	1*4 +	0*2 +	0*1
Decimal Value/Bit	0 +	64 +	0 +	16 +	8 +	4 +	0 +	0
Decimal Byte Value	92							

***Try converting binary value:* 1 1 0 0 1 0 1 1**