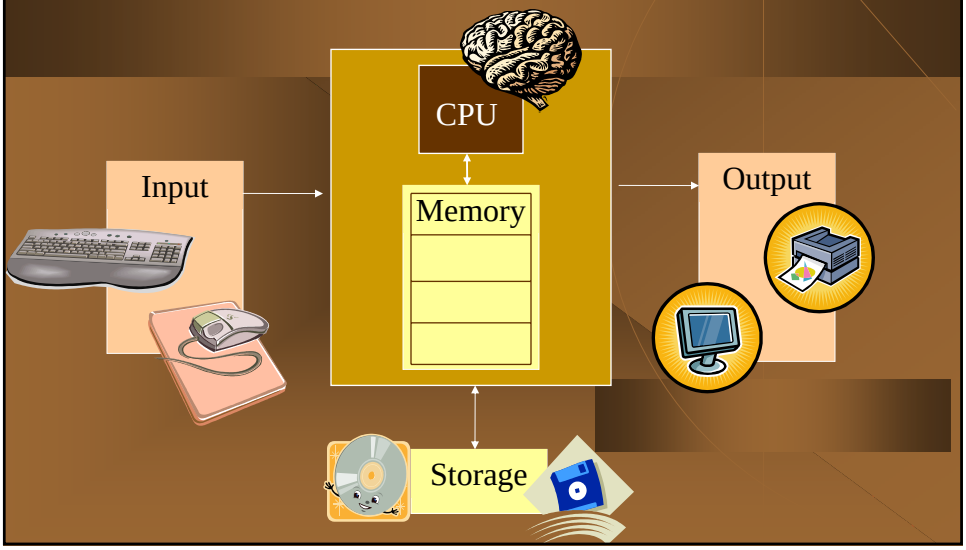
Lecture 2:

After completing this class, students will be able to:

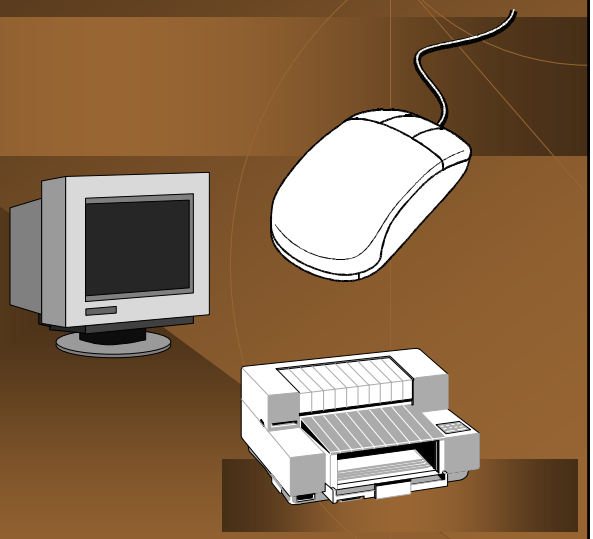
- Understand basic Computer Architecture
- Identify the major components of a computer.
- Understand how memory and memory addressing work
- Understand why programmers use binary and hexadecimal numbers.
- Recognize various types of computer errors.
- Write a pseudocode description of an algorithm.
- Identify the various phases of a software development project.

Computer Architecture



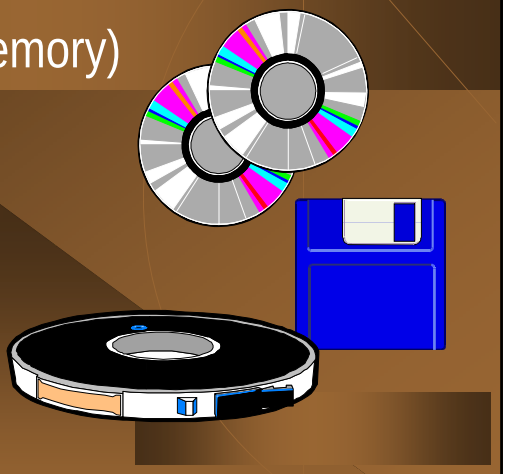
Input/Output

- keyboard
- mouse
- printer
- joystick
- monitor



Storage Devices (called secondary memory)

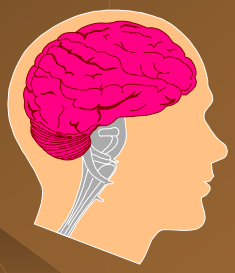
- Floppies
- CD_ROM
- Hard drives
- Magnetic tape



Central Processing Unit

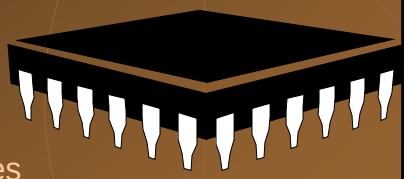
The **C**entral **P**rocessing **U**nit, also called the microprocessor or chip, is the brain of the computer. It interprets and carries out the instructions that operate a computer, like:

- running the operating system
- coordinating the hardware devices
- running application programs
- allocating resources, such as memory



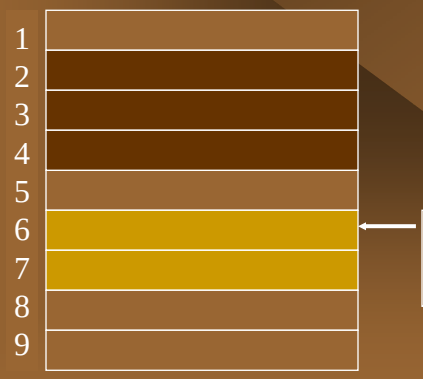
The CPU has two parts:

- The **Control Unit** coordinates and controls all parts of the computer
- The **A**rithmetic **L**ogic **U**nit does all the processing and calculations (including word processing)



01100111011101110110011101101110011010111011101110111011101

Memory Addresses (RAM)



2 byte memory location with address of 6

Binary Digits (bits)

Binary numbers use the base 2 number system.

Computers use binary.

Hexadecimal numbers use the base 16 number system.

Programmers use Hex because Hex converts easily to binary.

Example:
The decimal number 27 is
11011 in binary
2B in Hex

Example of ASCII:

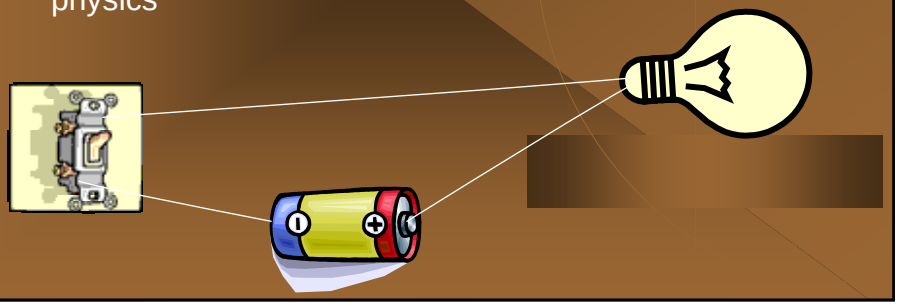
Char.	Dec.	Octal	Hex	Binary
“A”	65	101	41	0010 0001
“B”	66	102	42	0010 0010
“Z”	90	132	5A	0101 1010
“a”	97	141	61	0110 0001
“b”	98	142	62	0110 0010
“!”	33	041	21	0010 0001
“?”	63	077	3F	0011 1111

“Hello” stored in memory:

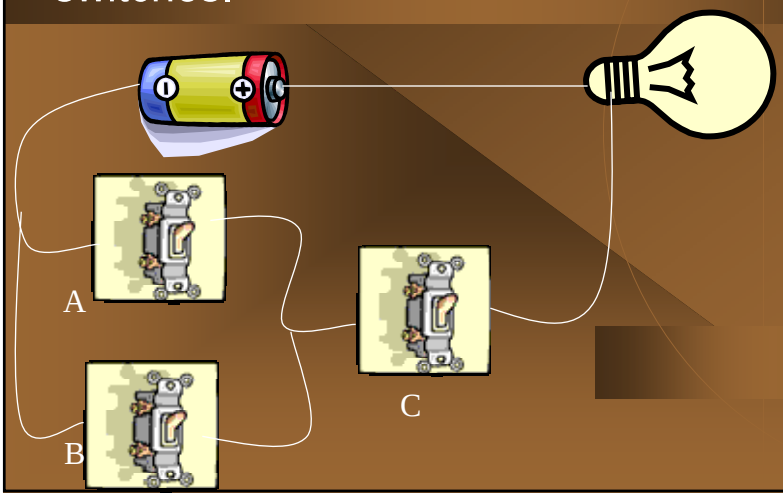
1	0100 1000
2	0110 0110
3	0110 1100
4	0110 1100
5	0110 1111
6	
7	
8	
9	

Programming a Computer

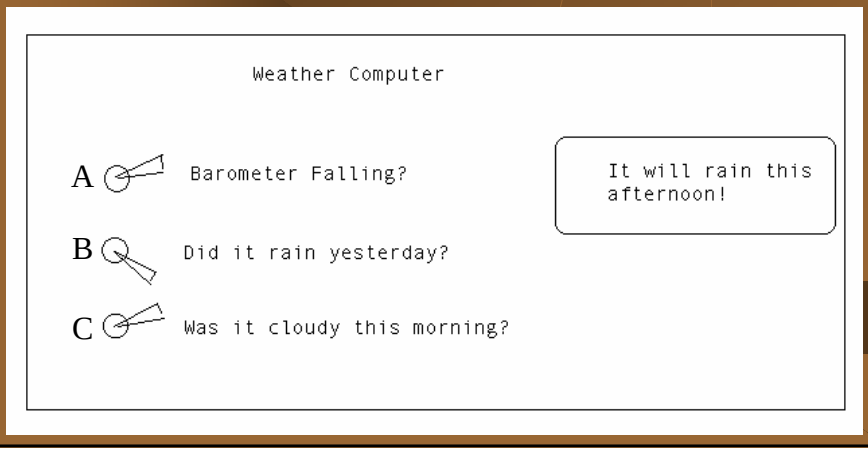
- A digital computer is a collection of electrical switches.
- The switches behave according to the laws of physics



A digital computer is a collection of electrical switches.



A program gives some meaning to the switches and actions.



Algorithm

The “computer” does not understand weather, it is simply computing an algorithm.

An algorithm is a sequence of precise instructions, which leads to a solution.

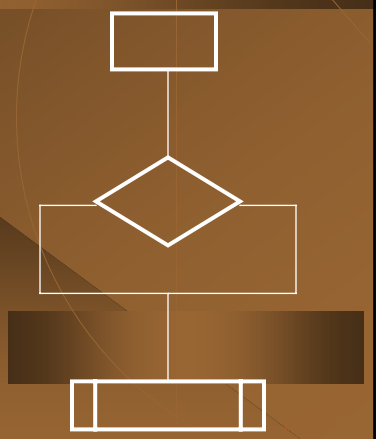
Example of an algorithm (page 14)

How many times does a name occur in a list of names:

1. Get the list of names
2. Get the name being checked
3. Set a counter to zero
4. Do the following for each name on the list:
   - Compare the name on the list to the name being checked
   - If the names are the same, add one to the counter
5. Announce that the answer is the number given by the counter

How do you write down an algorithm?

- Flow Chart
- Pseudo Code
- Others (Nasi-Schniderman)



Program design

- Problem solving phase (hard part)
  - Define the problem
  - Come up with an algorithm to solve the problem
  - Test the algorithm
- Implementation phase (easy part)
  - Translate algorithm to programming language
  - Test the program

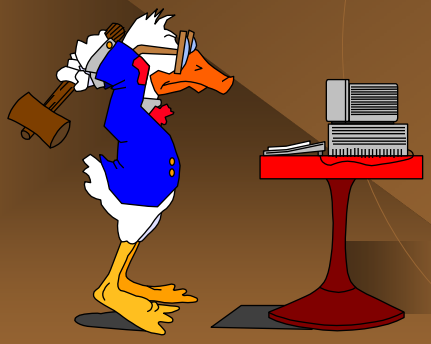
Quote from Donald Knuth

Computer programming is an art, because it applies accumulated knowledge of the world,

because it requires skill and ingenuity, and especially because it produces objects of beauty.

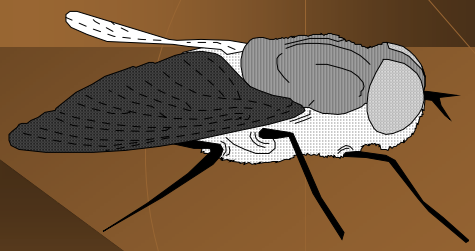
A programmer who subconsciously views himself (or herself) as an artist will enjoy what he (or she) does and will do it better.

Programming is an art form that fights back.



Computer Errors:

- Hardware flaws
- Algorithm Error
- Coding Error
- Input Error (bad data)
- Output Error (formatting)



Software Life Cycle

1. Analysis and specification of the task (problem definition)
2. Design of the software (algorithm design)
3. Implementation (coding)
4. Testing
5. Maintenance and evolution of the system
6. Obsolescence

EXAMPLE OF ASSEMBLER(6502)

(program adds N elements in a table, the first entry in the table is the number of elements)

```
            LDA     #0          initialize sum (load accumulator)
            STA     SUMLO       initialize sum
            STA     SUMHI       initialize sum
            TAY                 transfer accumulator into Y (init to 0)
            LDA     (BASE), Y   Get N (number of elements)
            TAY                 put N in register Y
            CLC                 clear carry for add
ADLOOP      LDA      (BASE), Y  get next element
            ADC     SUMLO       add SUMLO to accumulator
            STA     SUMLO       save result in SUMLO
            BCC     NOCARRY     branch on carry clear
            INC     SUMHI       SUMHI increased by one
            CLC                 clear for next sum
NOCARRY             DEY         next element (decrement Y)
            BNE     ADLOOP      back to ADLOOP if not equal to zero (branch)
            RTS                 return from subroutine
```

```
//Addition program in C++
//This program adds 10 integers

#include <iostream.h>

int main ()
{
      int integer, sum;                    // declaration

      sum = 0;                          // initialize sum

      for (n = 1 , n <= 10; n++)
      {
            cout << "Enter an integer\n";     // ask for an integer
            cin >> integer;                  // read an integer
            sum = sum + integer;         // add integer to the sum
      }

      cout << "Sum is " << sum << endl;   // print sum

      return 0; //indicate that the program ended successfully
}
```

Reading Assignment

- Read Chapter 1
- section 1.3 and 1.4, pages 18 – 31

- Read Chapter 2
- section 2.1, pages 37 – 45
- Section 2.3 pages 55 – 65
- Section 2.5 pages 83 - 88

Take a Break!

Lab:

In Class:
Programming project #1 - page 36 “Peapods”

Lab #1: *Programming Projects 2 & 3 page 37*

Return to Lynne's website

To return to Lynne Mayer's website, click on one of the following links:

Lynne's homepage

CIS110

CIS121