Some Basics

- Computerscan only understand *binary numbers*, 0 and 1 - these numbers are said to be *base 2* numbers.

- Each number, 0 or 1, is referred to as a *binary digit* or *bit*.

- *Instructions* that computers understand are just a collection of binary bits (e.g., 1000110010100000) - tell the computer to add two numbers (an example).

- Programmers, in the early days of computers - 50's, communicated (programs and commands) with computers in binary numbers - *machine language programming*.

- Itis difficult to generate binary numbers for a program and easy to make errors and hard to detect them.

- **In order to slightly solve the problem, computer programmers invented a symbolic notation, closer to the way humans think, in order to instruct and program computers - ADD A, B - this is called *assembly language programming*.**

- **For computers to understand this symbolic notation, we need a program to translate this symbolic notation to binary numbers. This program is called an *assembler*.**

- **Assembly language programming is simpler than machine language programming. But, it is still difficult to use, to write and debug large programs (e.g., a single instruction per line).**

- **In order to solve all these problems, programmers invented *high-level languages* (e.g., Fortran, Pascal, C, C++, etc.) using natural notations (e.g., C = A + B) to write programs to computers - they are simple to use and to debug.**

➜ **Programmers can think in a more natural language (English).**

➜ **Allow the design of languages for specific uses (e.g., scientific, business).**

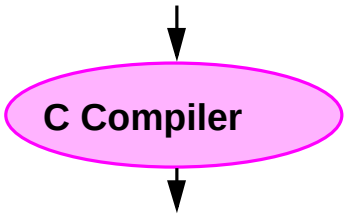
➜ **Improve programmers productivity.**

➜ **Allow programs to be independent of computers.**

- **Using high-level languages, we need programs to translate this language to assembly language - this program is called a *compiler*.**

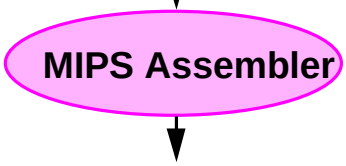
High-level language program (in C)

```
swap (int v[], int k);
{int temp;
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
}
```



Assembly language program (for MIPS)

```
swap:
multi $2, $5, 4
add $2, $4, $2
lw $15, 0($2)
lw $16, 4($2)
sw $16, 0($2)
sw $15, 4($2)
jr $31
```



Binary machine language program (for MIPS)

```
00000000101000010000000011000
00000000100111000011000000100
10001100011000100000000000000
10001100111100100000000000100
10101100111100100000000000000
10111000110001000000000000100
00000111110000000000000001000
```

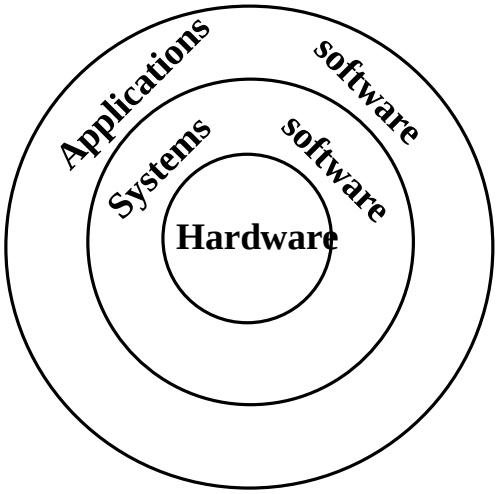
- **As programs become widespread, programmers saw the need to reuse some of the code they use. They invented *subroutine libraries* that contain programs that are frequently used by other programs.**

- **These subroutine libraries contain, among other things, programs for inputting and outputting data (e.g., keyboard, printers), controlling I/O devices (e.g., magnetic disks, monitors).**

- **Collectively these programs help the programmer operate a computer - they are called the *operating system* of the computer.**

- **Operating systems are programs that manage the resources of the computer.**

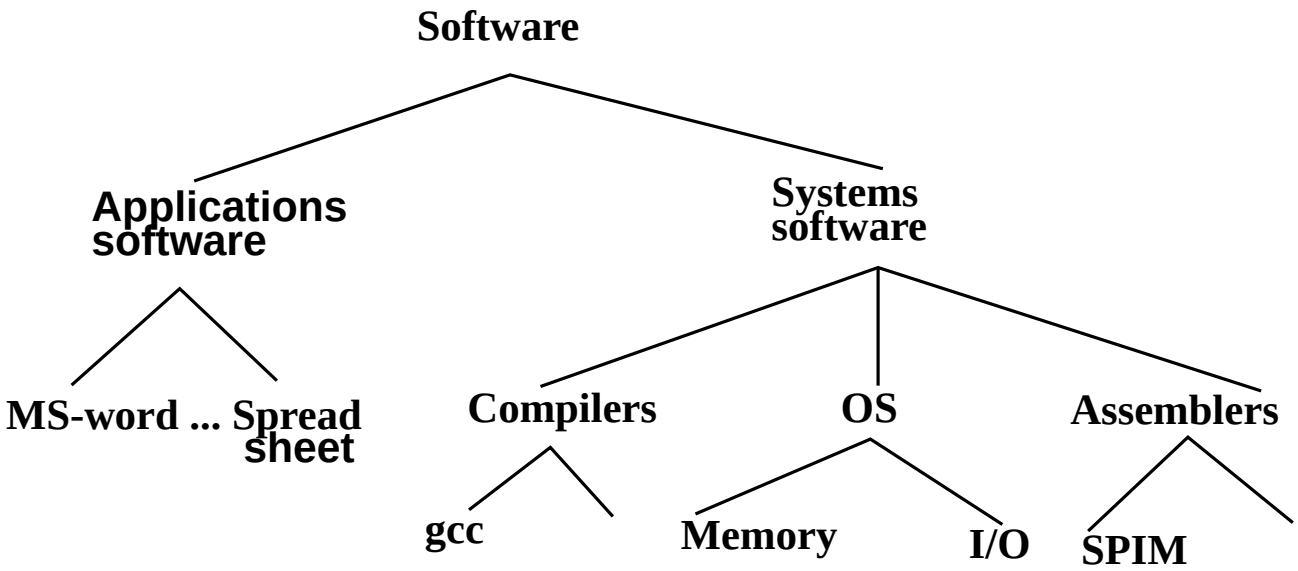
A classification of software:



Software can be categorized by its use:

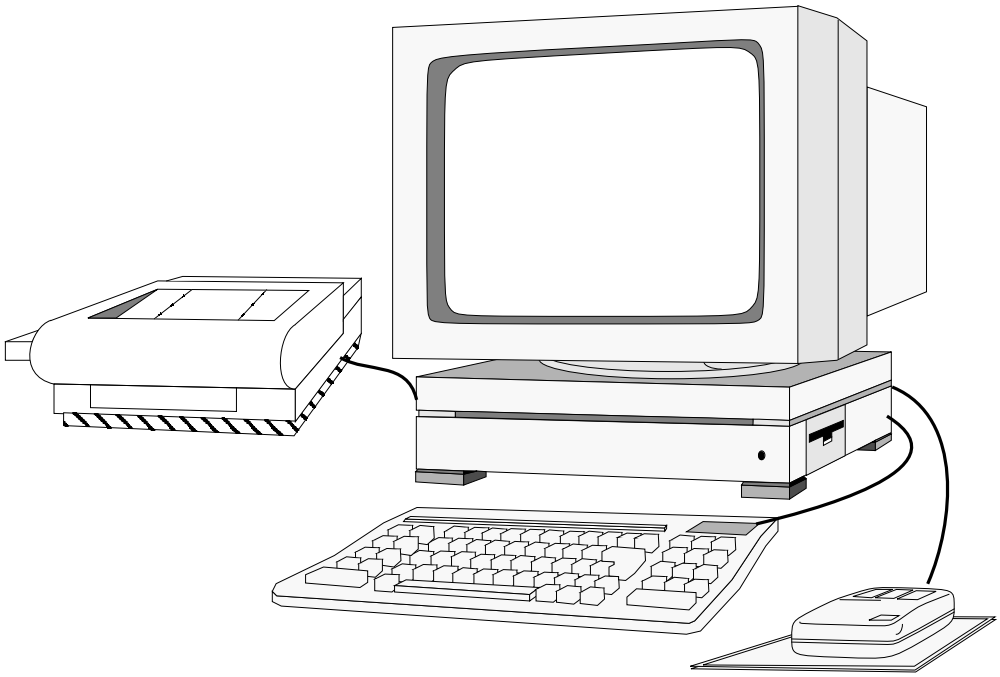
-> Software that provide services frequently useful is called *systems software* (e.g., operating systems, assemblers) - it is aimed at programmers.

-> Software that is aimed at computer users is called *application software* (e.g., spreadsheets, text-editors).



Inside your Computer

- **Atypical computer/workstation would have the following hardware components.**



- **A computer contains *input devices* (e.g., keyboard, mouse), and *output devices* (e.g., monitor, printer).**

- **There are some devices that provide *both* input and out put to the computer (e.g., disk, network).**

- **If we look inside the computer, then we find the *mother board*.**

- **The mother board contains *integrated circuits* or *chips*.**

- **The board is composed of three pieces: The piece connecting the I/O devices - called *interface* chips, the *memory*, and the *processor*.**

- **The memory is where the programs and data are kept when they are running.**

The Brain, the brain

The processor is the *brain* of the computer. It follows the instructions of the programs it is executing (e.g., add numbers, signals I/O devices, access memory, etc.).

- **The processor is also called *microprocessor* or *central processing unit* (*CPU*).**

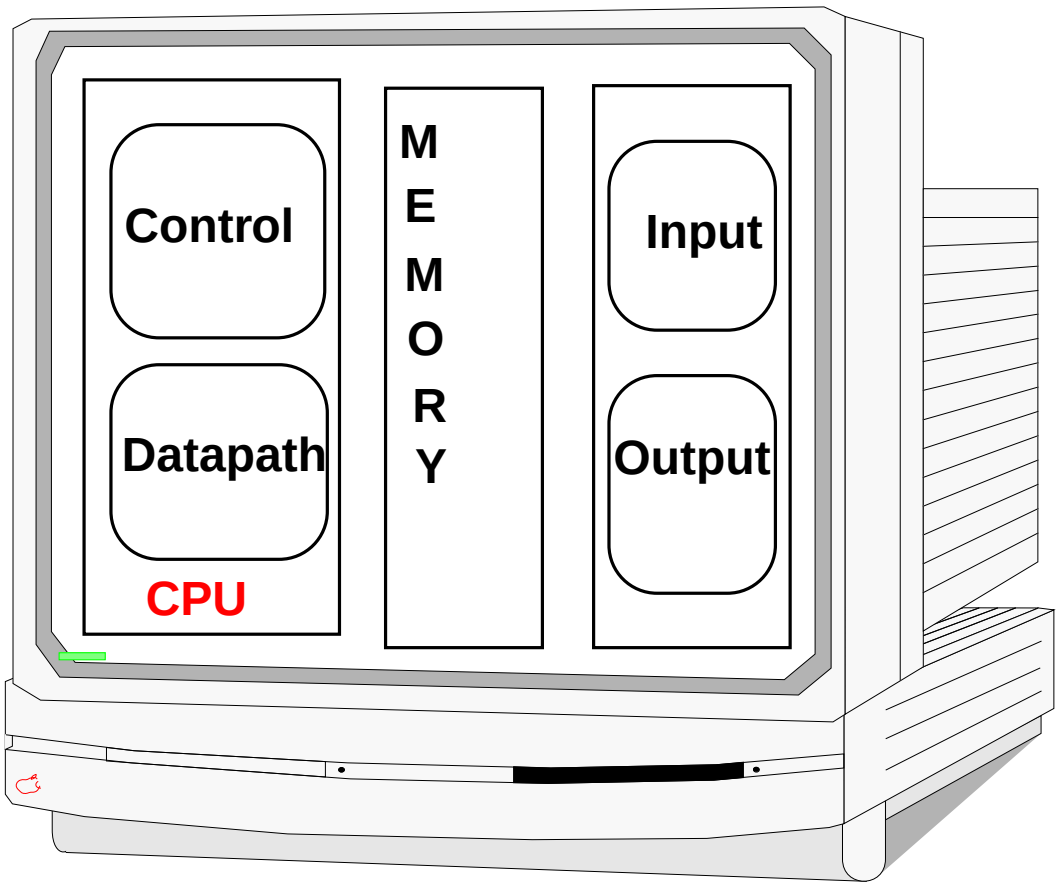
- **A processor has two main components:**

  ->A *datapath*: it performs the arithmetic operations.

  -> A *control*: it tells the datapath, memory, and I/O devices what to do according to the wishes of the instructions of the program running.

Overall Look

- A computer system contains five components: input, output, memory, datapath and control (processor).



Computer Architecture?

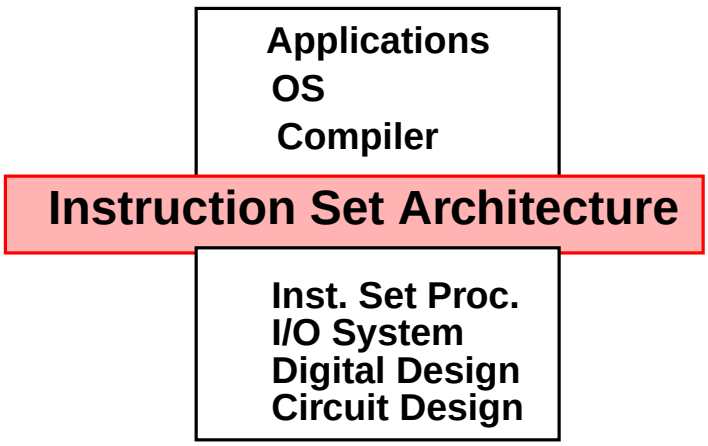
One of the most important issues in the design of a computer is the interface between the hardware and the lowest level software.

- **This is called the *instruction set architecture* or simply *architecture*. This what distinguishes, in big part, the performance and design of a computer (e.g., RISC vs. CISC).**

- **The instruction set architecture includes anything programmers need to know to make a binary machine language work correctly (e.g., instructions, I/O devices).**

Summary

Both hardware and software consist of hierarchical layers, with each layer hiding details from the layer above. One key interface between these levels is the *instruction set architecture*: The interface between the hardware and the low level software. This interface enables many *implementations* of varying cost and performance to run identical software.



Further Reading

Chapter 1. **David A. Patterson and John L. Hennessy. *Computer Organization & Design: The Hardware / Software Interface (Second Edition)*. Morgan Kaufman Publishers, 1998.**