Computers

• A computer performs 2 basic functions:
  1. executes a sequence of instructions (program)
  2. stores and recalls information (data)

• We can characterize a computer by:
  – its speed
  – the type of instructions it can execute
  – the capacity of its memory

• A computer system is made up of hardware and software.

Hardware

• The hardware refers to the electronic equipment that forms a computer system. These include:
  – Central processing unit (CPU):
    * responsible for executing programs
  – Input/output (I/O) devices:
    * keyboard, mouse, monitor, controllers etc.
  – Main memory
    * holds the program and data being processed by the CPU
  – Secondary memory or “storage” devices:
    * hard disks, floppy disks, etc.

The Central Processing Unit (CPU)

• The CPU reads and writes numerical values (instructions and/or data) to locations in the main memory.
• The CPU executes program instructions one at a time, and sequentially until directed otherwise.
• When we turn on our computer, the internal hardware gives the CPU the address of the first program instruction.
The CPU components

- **the control unit**: coordinates the transfer of data and instructions between main memory and the registers in the CPU
- **registers**: provides small memory storage in the CPU, e.g.:
  1. *instruction register*: holder for current instruction being executed
  2. *program counter*: holder for address of next instruction
  3. *general register(s)*: temporary storage for values as needed
- **arithmetic/logic unit**: performs calculations

Transmission of Data

- Data must be moved to and from these various memory locations, to and from the CPU.
- The electrical pathways within the computer that allow components to communicate with each other are called *buses*.
- Computer components (such as the CPU and the various types of memory), are laid out on a board, called the *motherboard*, with the buses connecting them together.
- All data, instructions and any other signals, travel on the bus.

System Clock

- A clock generates an electronic pulse at regular intervals which synchronizes the events of the CPU.
- The rate of the clock determines the *clock-speed*, and give an idea of how fast the CPU and process instructions (different processors may process a different number of instructions per clock cycle).

The Main Memory

- The main memory holds both programs and data.
- Information is stored in distinct "locations", specified by a unique numerical address.

![Memory Locations](image)

- In the earliest computers, programs and data did not share the same memory.
- In the late 1940s, John von Neumann first carried out the idea of using the same memory for both.
An Example Program

• How does the CPU interpret numeric values as instructions and data?
• Let’s look at a program that adds two numbers.
• An example program might look like this in main memory:

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1104</td>
<td>instruction</td>
</tr>
<tr>
<td>101</td>
<td>3105</td>
<td>instruction</td>
</tr>
<tr>
<td>102</td>
<td>2106</td>
<td>instruction</td>
</tr>
<tr>
<td>103</td>
<td>0000</td>
<td>instruction</td>
</tr>
<tr>
<td>104</td>
<td>1230</td>
<td>data</td>
</tr>
<tr>
<td>105</td>
<td>3017</td>
<td>data</td>
</tr>
<tr>
<td>106</td>
<td>0000</td>
<td>data</td>
</tr>
</tbody>
</table>

• Can you interpret this program?

Instructions for interpreting the program

• The first four locations are interpreted as instructions, and the last three as data.

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1104</td>
<td>instruction</td>
</tr>
<tr>
<td>101</td>
<td>3105</td>
<td>instruction</td>
</tr>
<tr>
<td>102</td>
<td>2106</td>
<td>instruction</td>
</tr>
<tr>
<td>103</td>
<td>0000</td>
<td>instruction</td>
</tr>
<tr>
<td>104</td>
<td>1230</td>
<td>data</td>
</tr>
<tr>
<td>105</td>
<td>3017</td>
<td>data</td>
</tr>
<tr>
<td>106</td>
<td>0000</td>
<td>data</td>
</tr>
</tbody>
</table>

• Meaning of instructions (for this example only):
  – The leftmost digit determines the instruction type:
    Digit | Meaning
    --- | ---
    "0"  | stop
    "1"  | load
    "2"  | store
    "3"  | add
  – The right three digits of the code indicate the memory location to use in the operation.

The program interpreted

• The program may be interpreted as follows:

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1104</td>
<td>LOAD 104</td>
</tr>
<tr>
<td>101</td>
<td>3105</td>
<td>ADD 105</td>
</tr>
<tr>
<td>102</td>
<td>2106</td>
<td>STORE 106</td>
</tr>
<tr>
<td>103</td>
<td>0000</td>
<td>STOP</td>
</tr>
<tr>
<td>104</td>
<td>1230</td>
<td>DATA</td>
</tr>
<tr>
<td>105</td>
<td>3017</td>
<td>DATA</td>
</tr>
<tr>
<td>106</td>
<td>0000</td>
<td>DATA</td>
</tr>
</tbody>
</table>

• The CPU does the following when executing the program:
  – place the contents of location 104 in a specific place in the CPU called a register.
  – bring the contents of location 105 into the CPU and add it to the data already stored in the register
  – the register now holds only the result of the addition—individual values are lost.
  – move the result into location 106 for storage

Program Before and After Execution

Before execution

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1104</td>
<td>LOAD 104</td>
</tr>
<tr>
<td>101</td>
<td>3105</td>
<td>ADD 105</td>
</tr>
<tr>
<td>102</td>
<td>2106</td>
<td>STORE 106</td>
</tr>
<tr>
<td>103</td>
<td>0000</td>
<td>STOP</td>
</tr>
<tr>
<td>104</td>
<td>1230</td>
<td>DATA</td>
</tr>
<tr>
<td>105</td>
<td>3017</td>
<td>DATA</td>
</tr>
<tr>
<td>106</td>
<td>0000</td>
<td>DATA</td>
</tr>
</tbody>
</table>

After execution

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1104</td>
<td>LOAD 104</td>
</tr>
<tr>
<td>101</td>
<td>3105</td>
<td>ADD 105</td>
</tr>
<tr>
<td>102</td>
<td>2106</td>
<td>STORE 106</td>
</tr>
<tr>
<td>103</td>
<td>0000</td>
<td>STOP</td>
</tr>
<tr>
<td>104</td>
<td>1230</td>
<td>DATA</td>
</tr>
<tr>
<td>105</td>
<td>3017</td>
<td>DATA</td>
</tr>
<tr>
<td>106</td>
<td>4247</td>
<td>DATA</td>
</tr>
</tbody>
</table>
A Higher Level Program

The same program in a higher level programming language such as Java, is much more human friendly!

```java
int a = 1230;
int b = a + 3017;
//b now holds the value 4247
```

Software

The software refers to the programs that are executed by a computer.

An essential software of the computer system is the operating system (OS) which

1. provides a user interface, typically a graphical user interface (GUI), allowing interaction between the user and the computer.
2. manages computer resources such as the CPU and main memory, e.g.:
   - defines how files are organized
   - how memory is managed,
   - how the CPU is used
3. provides system calls (e.g. memory allocation, opening, reading, and saving files, etc.) for software applications.

Analog Signals

- Analog signals are continuous in time.

![Analog Signal Example]

Figure 3: The electrical signal used to represent the pressure variations of a sound wave is an analog signal.

- The term “analog” refers to the fact that it is analogous of the signal it represents.
- A continuous signal cannot be stored, or processed, in a computer since it would require an infinite amount of data.
- Analog signals must therefore be discretized, or digitized, to produce a finite set of numbers, for computer use.

Digital Signals

- The process of taking individual values on a continuous signal at regular time intervals is called sampling.
- Music on compact discs is stored digitally. The sampling rate is typically 44,100 samples per second (or 44.1 kHz).

![Digital Signal Example]

Figure 4: A digital, or sampled, signal.
How is Data Stored?

- There are several ways to represent numbers.
- Normally we, humans, use the *decimal system* (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, ...).
- Computers on the other hand, represent numbers in *binary form*.
- Data in *binary form* and can represent any kind of information (sound, music, text, numbers, pictures, video, etc.) or operating instructions.
- To get an idea how the binary system works, lets first look at the more familiar decimal system.

Decimal System

- The decimal system is based on the number 10.
- The value of an individual digit, depends on its relative position in a group of other digits. Each position is worth 10 times as much as the position to the right.
- The number 126 is evaluated as:
  \[ 1 \times 10^2 + 2 \times 10^1 + 6 \times 10^0 = 126 \]

Binary System

- The binary system is based on the number 2.
- Each bit position in the group is worth twice as much as the position to the right.
- The binary number 1001 is evaluated as
  \[ 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 9 \]
- Other systems include octal and hexadecimal.

Bits

- The smallest unit of memory is the bit, representing the position, either “on” or “off”, of an electronic switch inside the computer hardware.
- A bit can represent either “0” or “1” depending on the state of the switch. The term is a contraction of the term “binary digit”.
- Two (2) bits can assume 4 unique patterns and thus 4 unique values.

<table>
<thead>
<tr>
<th>Binary form</th>
<th>Decimal form</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
</tr>
</tbody>
</table>

- In general, \( n \) bits can assume \( 2^n \) unique values.
Bytes

- The byte is used to measure memory capacity, and is nearly always defined as 8 bits.
- A byte can therefore assume \(2^8 = 256\) unique values.
- The number of bytes used to represent instructions and datum vary, e.g., one byte is used to store the letter G: 01000111.
- The most widely used coding system is ASCII (American Standard Code for Information Interchange). It consists of 128 numerical codes that represent alphanumeric characters, punctuation marks, and special control codes.
- Numerical values generally occupy from 1 to 8 bytes.

Numeric Representation

- Integers represented using 16 bits (called shorts) provide a range of -32,768 to +32,767.
- One bit is allocated for the sign, usually the most significant bit, so that a leading '0' indicates a positive number and '1' a negative number.
- For the sake of saving space, let’s look at only 8 bits.
- For positive numbers, there are 7 bits to represent the magnitude, allowing for a highest value of:
  \[01111111 = 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 127\]

Two’s Complement

- There are several ways of representing negative numbers, the most common being two’s complement.
  - take the binary representation of the magnitude
  - take the complement of all the bits (change 0s to 1s and 1s to 0s)
  - add 1
- The number -128 would then be
  - 10000000 (binary representation of the magnitude)
  - 01111111 (taking the complement bits)
  - 10000000 (adding 1)
  which is negative because of the leading '1'.
- The lowest number that can be represented using 8 bits is therefore -128. Why is it not 11111111?

Floating Point Representation

- The floating point representation was developed to represent numbers with fractional parts.
- To get an idea of how the floating point representation works, consider a number 824.68, which can be written using a fractional part and an exponent:
  \[824.68 = 0.82468 \times 10^3\]
- The exponent, 3, is the number of places the decimal point had to be shifted to the left in order to be in front of the first digit.
- The fractional part, 0.82468, known as the mantissa.
Floating Point Numbers

- A number less than 0.1, such as 0.068514, can be similarly represented as
  \[0.068514 = 0.68614 \times 10^{-1},\]
  where the negative exponent indicates that the decimal point must be moved to the right.
- A computer stores a floating-point number as a separate mantissa and exponent (though usually in binary form) using 4 bytes:
  - 1 byte for the exponent
  - 3 bytes for the mantissa gives a resolution of 7 decimal digits (including sign).
- Double precision uses a total of 8 bytes and give a little over 15 decimal digits of resolution.
- Operations on floating-point and double precision numbers are usually performed more slowly than with integers (unless the computer has special hardware).

Units of binary storage

- Because computer memory is based on the binary number system, all units of storage are powers of two.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Symbol</th>
<th>Number of Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td></td>
<td>(2^0 = 1)</td>
</tr>
<tr>
<td>kilobyte</td>
<td>KB</td>
<td>(2^{10} = 1024)</td>
</tr>
<tr>
<td>megabyte</td>
<td>MB</td>
<td>(2^{20} = 1,048,576)</td>
</tr>
<tr>
<td>gigabyte</td>
<td>GB</td>
<td>(2^{30} = 1,073,741,824)</td>
</tr>
<tr>
<td>terabyte</td>
<td>TB</td>
<td>(2^{40} = 1,099,511,627,776)</td>
</tr>
</tbody>
</table>

Where is Data Stored

- So now that we now how data is stored in a computer, let’s look a little closer at where it is stored.
- There are three primary types of memory:
  1. RAM (Random Access Memory)
  2. ROM (Read-Only Memory)
  3. Storage memory

RAM (Random Access Memory)

- RAM is the “fast” memory in which data and instructions are stored when being used by the CPU. Also referred to as “main memory”.
- It is volatile: when the power is shut off, the memory is wiped clean.
### Types of RAM

- **DRAM (Dynamic RAM):** Data is erased once it is read, and must be constantly rewritten (once read) to keep it present. Relatively inexpensive.
- **SRAM (Static RAM):** Data does not need to be constantly rewritten in order to keep it secure, but more expensive than DRAM.
- **VRAM (Video RAM):** Special memory (typically a type of SRAM) used for holding video. Used on graphics, video and accelerator cards.
- **Flash memory.** A cross between RAM and forms of ROM. Constructed like EEPROM, is non-volatile, but is fast like RAM. Can only erase multiple memory locations at a time: changing a byte is only possible by rewriting a whole block.

### ROM (Read-Only Memory)

- Memory in which instructions are permanently encoded: cannot be changed, and are not erased when the computer is shut down.
- Commonly used to start and set-up a computer: when the power is turned on, the instructions in ROM are loaded into the computer’s RAM.

### Types of ROM

- **PROM (Programmable ROM):** Can be programmed after the chip is produced. Instructions are “burned” into the chip electrically (as coded pulses of electricity), after which they cannot be changed.
- **EPROM (Erasable PROM):** Instructions can be changed by exposing the chip to ultraviolet light.
- **EEPROM (Electrically Erasable PROM):** Instructions can be erased using an electrical current rather than normal ultraviolet light: faster and can be done in-circuit.

### Storage (secondary) memory

- Non-volatile (stable), and used to store large quantities of data.
- Includes hard drives, diskettes, CD-ROMS, etc.
- Since it is slower than RAM and ROM, data is copied to RAM before being used by the CPU.
- **Virtual memory** is storage memory can be used as a slower RAM, thus simulating a computer with more RAM. Particularly useful when working with large files.
Cache Memory

- Used to allow more efficient operation of the CPU.
- The processor keeps the most recently used data and instructions in the cache, making it immediately available instead of having to make the slower request from storage memory or RAM.
- Usually a type of SRAM.
- Expensive, though usually smaller.