**Georgia Southern University**

**Allen E. Paulson College of Engineering and Computing**

**Information Technology Department**

**Dr. Atef Shalan**

**Course: IT2530-Operating Systems**

**Module 1-** Introduction to Computers and Operating Systems– reading material

Contents and Reading Material

[**1** Module 1-Computer organization and Architecture. 1](#_Toc88511388)

[1.1 Data representation in digital systems 2](#_Toc88511389)

[1.2 Introduction to logic circuit design 3](#_Toc88511390)

[1.3 Von Neumann computer architecture 3](#_Toc88511391)

[1.4 Introduction to Assembly language and basic instructions 4](#_Toc88511392)

[1.5 Computer system structure 4](#_Toc88511393)

[1.6 Memory, caching, and storage hierarchy 5](#_Toc88511394)

[1.7 Multiprocessor system architecture 6](#_Toc88511395)

[1.8 What is operating systems 6](#_Toc88511396)

[1.9 Operating System Components 7](#_Toc88511397)

[1.9.1 Process Management Overview 8](#_Toc88511398)

[1.9.2 Basics of Memory Management 8](#_Toc88511399)

[1.9.3 File System management 9](#_Toc88511400)

[1.9.4 I/O management 9](#_Toc88511401)

[1.9.5 Network management 9](#_Toc88511402)

[1.9.6 OS Protection and Security 10](#_Toc88511403)

# Module 1-Computer organization and Architecture.

The first step towards Operating Systems (OSs) learning is understanding what systems are and how computer systems work. We will also take one further step back to learn the basics of data representation in the digital systems and computing world. This start will enable building OS skills on top of a strong basis of foundational knowledge related to computer systems. Module 1 is an introductory module that sets the stage for learning advanced OS and system administration skills by covering selective basic foundational knowledge areas in data representation, processor design and architecture, computer model and machine code, system organization and hardware design, computer components, computer, and multiprocessor architectures.

## Data representation in digital systems

Data exist in our world in various forms such as numbers, text, images, videos, and others. In the computer world, *data representation* is the method that can be used for encoding various types of data into digital forms that are suitable for computer processing. At the data processing level, computers can only distinguish two types of data values: 0 and 1. This is also analogous to *False* and *True*, respectively at the system design level, and low and high voltage, respectively at the hardware level. Therefore, all types of data must be encoded into sequences of 0’s and 1’s before it can be processed in computers. We call this encoding scheme *binary system*. While the binary representation of data includes numerous patterns and methods depending on the type of data, precision, range limitation, etc. Therefore, we first need to gain some knowledge about [the meaning of binary representation](https://computersciencewiki.org/index.php/Data_representation) and [how to use it to represent numbers](https://www.mathsisfun.com/binary-number-system.html). We also need to explore how text data is represented in binary format by using character encoding schemes and understanding the [basic ASCII table](https://www.asciitable.com/) and [Unicode characters](https://www.lookuptables.com/text/unicode-characters). In this part, we also need to be familiar with image representation in binary systems. In its simple form, an image is a rectangular grid of tiny squares called pixels. These pixels are adjacent to each and each one of them is completely a separated light device that takes only one color depending on the resolution of the image and the coloring scheme utilized.

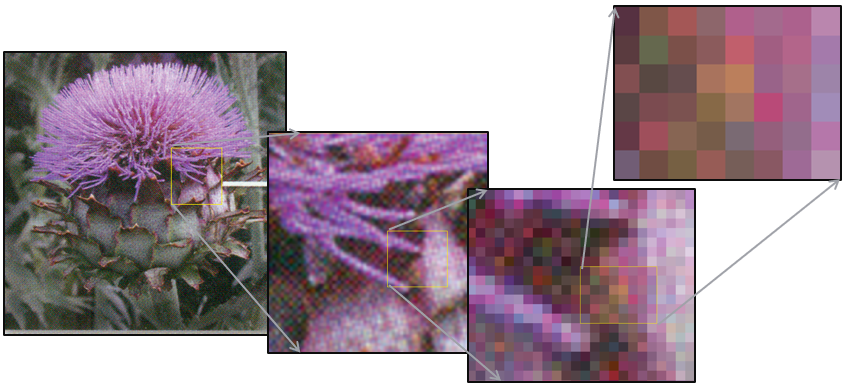


Figure 1-Three level zoom into an image to see pixels

Figure 1, displays an image of a cactus flower and three zoom levels into an image of a cactus flower in four squares from left to right: Square 1, 2,3, and 4, respectively.

Square 1 on the left displays the whole image of the flower. Square 2 (Second square towards the right) zooms over the yellow rectangle indicated on square 1. Recursively square 3 zooms over the yellow rectangle indicated on square 2, and finally, square 4 zooms over the yellow rectangle indicated on square 3. This magnification and zooming sequence illustrates the pixels in square 4 that represent a tiny part of the image. If the image size in pixels is 1200 X 1920, then the image includes 1200 times 1920, i.e., 2304000 pixels in its grid, where each pixel has a specific color. There are multiple schemes for representing the color in computers, the most common scheme is called [Red-Green-Blue (RGB) color scheme](https://www.w3schools.com/colors/colors_rgb.asp). Now we understand that all data can be converted into binary sequences of binary digits. In the following, we will learn how to build small devices that can process this digital data.

## Introduction to logic circuit design

In our daily lives, we process information at all levels either manually or automatically by using computing devices, e.g., calculating, reading, rewriting, rendering, playing, etc. At the computer level, since all data is represented in binary, all the above operations must be automated and aggregated through a simplified model that allows processing data at the digital level. [Logic circuit design](https://www.britannica.com/technology/logic-design) is the basic organization of the circuitry of a digital computer. The logic design allows using very simple components called logic gates to build simple [computing devices (*digital circuits*)](https://www.sciencedirect.com/topics/engineering/logic-design) such as [full adder](https://www.sciencedirect.com/topics/computer-science/full-adder), [SR-Latch & D-Latch](https://en.wikibooks.org/wiki/Digital_Circuits/Latches), decoder, multiplexer, and many others, by implementing them as logic functions. The mix and match of these digital circuits into larger digital components allow the provisioning of high-level operations such as the previously mentioned high-level operations.

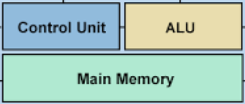
A [computer microprocessor](https://informerguru.com/how-microprocessor-works-and-its-components/) is a miniature device that contains thousands of logic circuits and millions of logic gates. Microprocessors provide the arithmetic, logic, in addition to the necessary control circuitry to perform the data processing functions. As shown in Figure 2, the main components of a microprocessor are [Arithmetic and Logic Unit (ALU)](https://informerguru.com/how-microprocessor-works-and-its-components/#1_Arithmetic_Logic_Unit), [Control Unit (CU)](https://informerguru.com/how-microprocessor-works-and-its-components/#2_Control_Unit), and [register memory](https://informerguru.com/how-microprocessor-works-and-its-components/#3_Registers) are the main components of a Center Processing Unit (CPU). In this section, we also need to learn about the [X86 microprocessor family](https://en.wikibooks.org/wiki/X86_Assembly/X86_Family) which is currently ubiquitous in stationary and portable computers, midrange computers,  servers, and most High-Performance Computing (HPC) clusters.

Figure - CPU Structure

## Von Neumann computer architecture

With the help of computer main memory, the CPU can execute program instructions and process program data through the logic circuitry in the ALU, CU, and register memory. [Von Neumann Architecture](https://history-computer.com/the-complete-guide-to-von-neumann-architecture/) provides a computing model for exciting memory-based on computer CPU one instruction at a time.

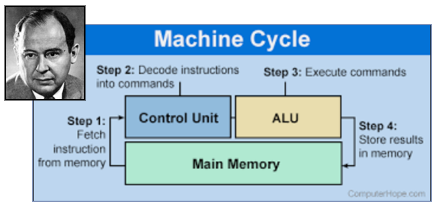


Figure 3- Von Neumann Computer Architecture - Fetch and Execute cycle.

Figure 3 describes the [fetch-excite cycle](https://www.thecomputingteacher.com/csc/index.php/sa-d/hardware/fetch-execute-cycle) in which the CPU reads and executes one instruction at a time and writes the results back to memory before cycling into the next instruction.

Programs are executed in computer CPUs as Assembly language or machine code. In the following section, we will learn about Assembly languages and their basic instructions.

## Introduction to Assembly language and basic instructions

Each [processor family](https://en.wikichip.org/wiki/list_of_microprocessor_families) (e.g., Intel, AMD, and Cyrix) is designed with a specific [set of instructions](https://semiengineering.com/knowledge_centers/compute-architectures/instruction-set-architecture-isa/) that can handle various operations on these processors. Consequently, there is a list of [Assembly languages](https://www.tutorialspoint.com/assembly_programming/assembly_introduction.htm) (Assemblers) for each of these processor families. Examples of Assemblers include [MASM](https://docs.microsoft.com/en-us/cpp/assembler/masm/masm-for-x64-ml64-exe?view=msvc-170) from Microsoft, [AT&T](https://gist.github.com/mishurov/6bcf04df329973c15044) from AT&T company, and [IBM assembler HLASM](https://www-01.ibm.com/servers/resourcelink/svc00100.nsf/pages/zOSV2R3sc264940/$file/asmr1023.pdf) from IBM. Most of these assembly languages share a similar design to the core instruction set. [Processor registers](https://en.wikipedia.org/wiki/Processor_register) are a quickly accessible memory location available to a processor CPU. The number of bits characterizing the memory addressing in computing systems is a major design factor and currently, most systems are either 32 bits or 64 bits wide. 32-bit systems use 4 bytes (32 bit) registers and address buses. Processor registers are the components of assembly programs. In the [32-bit systems](https://www.eecg.utoronto.ca/~amza/www.mindsec.com/files/x86regs.html), there are a total of 16 registers grouped into 3 categories general, control, and segment registers. An assembly program is constituted of a list of statements. An Assembly statement takes the following syntax (i.e., form):

|  |
| --- |
| [label] mnemonic [operands] [; comment] |

The label is an identifier, mnemonic is an assembly instruction, operands can be a list of variables, registers, label, or others, and comment is an optional text describing the code statement. The following are sample assembly code statements:

|  |
| --- |
| mov ecx, text; assign text to ecx  mov ebx, 1 ; assign 1 to ebx |

You can learn more about [Assembly language syntax](https://www.tutorialspoint.com/assembly_programming/assembly_basic_syntax.htm) to be familiar with the low level language structure and program execution.

## Computer system structure

A computer system is considered to include four main components: Computer hardware, Operating System (OS), Applications, and users. Computer hardware is the basic computing resources, e.g., CPU, memory, I/O devices. OS is the control system that coordinates the use of hardware resources among multiple applications and users, e.g., Windows, Linux, macOS. Application programs define how the system resources are used to solve the computing problems of the users, e.g., Microsoft Word, Google Chrome, SQL server. Finally, users are people, machines, other computers that can use the whole system.

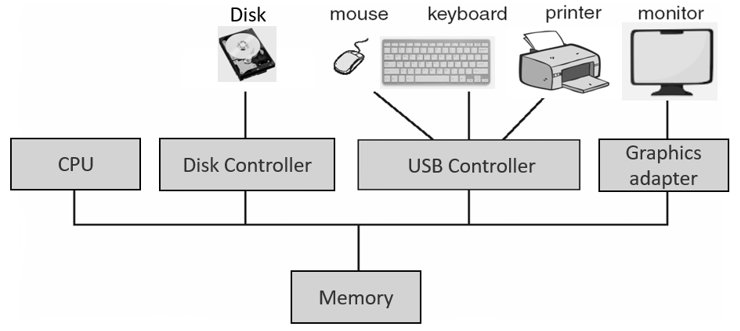


Figure 4- Computer hardware structure

Figure 4- displays the four main elements of [Computer hardware](https://en.wikipedia.org/wiki/Computer_hardware): processor, main memory, I/O devices, and system bus. Processors provide operation control and data processing. Main memory provides temporary storage for programs and data during execution. I/O devices allow interaction with the external environment of computers. Finally, allows controlled communication and data transfer among these computer components. Most [I/O devices are Interrupt-driven](http://inputoutput5822.weebly.com/interrupt-driven-io.html), i.e., their I/O activities are controlled through interrupt signals to the processors. Interrupts can be generated by hardware (e.g., I/O device failure or timer interrupt) and software (e.g., exception handler)

## Memory, caching, and storage hierarchy

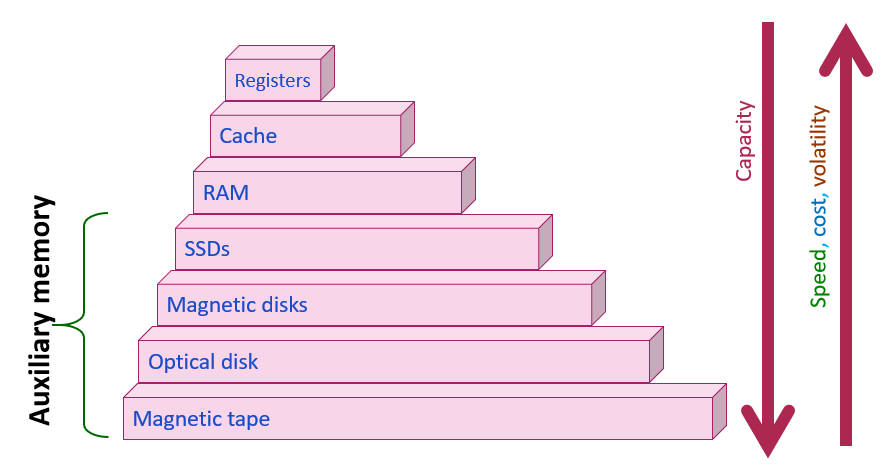


Figure 5- Memory Hierarchy

There are multiple different memory devices in computer systems. Some memory devices provide volatile storage (i.e., data only persists when computers are turned on) and some provide non-volatile (or auxiliary memory) storage (e.g., Magnetic disks and Solid-State Disks (SSDs)) which preserves data even if computers are turned off. Memory devices vary considerably concerning their capacity, speed, and cost. Therefore, technical documents describe computer storage through a [memory hierarchy](https://www.elprocus.com/memory-hierarchy-in-computer-architecture/) as shown in Figure 5. In this hierarchy, memory devices near the bottom are larger in capacity but lower in speed, cost, and volatility. Memory devices near the top are smaller in capacity, highly expensive, and considerably faster.

Memory does not only come in standalone devices. Most I/O devices have memory buffers to facilitate data transfer between a computer and its environment. OSs have a device driver (or device controller) that maintains a local buffer for each I/O device. During I/O operations, the CPU moves data between main memory and local device buffers. Relative to the amount of data that is often transferred in I/O operations, device buffers are relatively too small. Therefore, moving a large amount of data using interrupt-driven I/O can produce high overhead when used for bulk data movement. Caching (devising temporary storage between faster and slower devices) is the main solution to this problem. If data is needed from the slow device, a cache is checked first. If data exist, then data from the cache is used directly, otherwise, data is copied to cache first and then used. In Computer systems, there are three types of [cache memory](https://searchstorage.techtarget.com/definition/cache-memory), L1, L2, and L3 cache. L1 cache is extremely fast but relatively small, L2 is larger and slower than L1 but smaller and faster than L3 cache.

## Multiprocessor system architecture

A [computer system architecture](https://429151971640327878.weebly.com/blog/13-computer-system-architecture) can be characterized by the used number of general-purpose processors. Therefore, computer systems can be categorized into single-processor, multiprocessor, multicore, and clustered systems. In a single-processor system, there is only one main general-purpose processor. Multiprocessor systems (AKA parallel or tightly coupled systems) have two or more processors in a single machine. Multicore systems are like multiprocessor systems except that multicore system has two or more processor cores on a single chip. Clustered systems are also like multiprocessor systems except that, they are composed of two or more individual whole systems in communication through Local Area Network (LAN). Benefits of the multiprocessor and multicore systems include the following.

* Increased throughput
* Economy of scale
* Increased reliability
* Graceful degradation

Multiple-processor systems can be organized in two designs: Asymmetric and Symmetric multiprocessing or multi-clustering. In asymmetric multiprocessing, each processor is assigned a specific task, while in symmetric multiprocessing, each processor performs all tasks within the operating system. Symmetric and asymmetric multi-clustering are analogs to Symmetric and asymmetric multiprocessing except that jobs are distributed at a single computer level, not at a processor level.

## What is operating systems

[An operating system](https://edu.gcfglobal.org/en/computerbasics/understanding-operating-systems/1/) is a software program that makes computer systems usable and useful. It is considered an intermediary system between computer users and computer hardware. Operating systems make computer systems convenient to use and allow computer users to utilize the computer hardware efficiently. Operating systems provide an organized platform to execute user programs and interact with the computer environment via the I/O devices. Operating systems are also designed to allow the evolution of a computer’s capabilities by adding or removing system features and applications and/or installing or removing hardware components.

The functionality of OSs can be viewed from two perspectives: control and service. As a control system, OS provides the following main functions.

* Manages the computer service initiation process (using the [bootstrap program](https://www.techopedia.com/definition/3328/bootstrap)), [operating system kernel](https://afteracademy.com/blog/what-is-kernel-in-operating-system-and-what-are-the-various-types-of-kernel), and service startups.
* Manages and allocates computer resources to users and processes and maintains fairness, efficiency, and prioritization among system processes.
* Manages the communication between computer resources and resolves device conflicts.
* Manages user programs execution.
* Manages and accounts computer system interactions and architectural interfaces: [Application Binary Interface (ABI)](https://en.wikipedia.org/wiki/Application_binary_interface) and [Application Programming Interface (API)](https://www.ibm.com/cloud/learn/api).

As a service system, OS allows users to do the following.

* Utilize computer system resources in a reliable and efficient manner
* Deploy and create new systems and applications
* Access the I/O devices and interact with the surrounding environment
* Access systems resources e.g., processing, memory, storage, files, etc.
* Manage and evolve system software and hardware resources.

[Operating systems evolution](https://www.technologyuk.net/computing/computer-software/operating-systems/operating-system-evolution.shtml) describes the change of operating systems over time concerning their handling of multiple jobs and utilizing computer microprocessors. Earlier OS were serial processing systems in which a computer could only run a single task at a time. To improve the processor utilization, batch operating systems allowed computer operators to batch multiple jobs individually at specific time schedules. Multiprogramming operating systems allowed computers to run multiple jobs at the same time while keeping only one interactive job running on the system. Currently, time-sharing operating systems are capable to execute many processes interactively at the same time. These OSs slice the time (e.g., a second) into tiny intervals called time quantum and interleave the execution of the running processes over these time slices while maintaining the response time of these processes to less than one second.

## Operating System Components

Due to the large size and complexity of operating systems, they are usually designed and created in a layered structure by partitioning them into smaller parts. An [operating system structure](https://www.javatpoint.com/components-of-operating-system) includes the following list of components as part of their designs:

* Process Management
* Memory Management
* Storage Management
* I/O Device Management
* File Management
* Network Management
* Security Management

This course focuses on the core part of operating systems and therefore studies most of the above components in detail in separate modules. In the following sections, we briefly describe each component as an overview of OS component structure.

### Process Management Overview

Processes are running software programs and therefore, they are a fundamental element of OS structure. The [process management](https://www.guru99.com/components-of-operating-system.html) component of OSs is responsible for orchestrating and allocating resources for multiple processes execution simultaneously and efficiently. A process is an entity that can be executed on a processor and allocated resources. Thus, a process contains three main components:

1. An executable program
2. Process data (e.g., variables, workspace, buffers, etc.)
3. The execution context (e.g., process state, process control information, priority)

The following are the main process management activities related to process execution:

1. Process creation and deletion
2. Processes suspension and resuming
3. Process synchronization
4. Process communication
5. Resolving process deadlocks

[Multithreading](https://www.educative.io/blog/multithreading-and-concurrency-fundamentals) is a technique that allows dividing a process into smaller and lightweight portions (called threads) that can run concurrently. In multithreading, the computer processor time is sliced and multiplexed among the executing processes/threads. Each thread is considered a mini process that has its program code counter, thread data, and thread context.

### Basics of Memory Management

Memory is required as the main resource to execute programs. In operating systems, [memory management](https://www.tutorialspoint.com/operating_system/os_quick_guide.htm) refers to the activities related to organizing a computer’s main memory (primary memory) and virtual memory (extended memory). Memory management includes the following main activities.

* Monitor memory utilization by tracking and accounting empty and used memory slots.
* Account the memory usage and maintain records of currently executing processes.
* Decide whether a process is allowed to move into (or move out of) memory.
* Allocate and deallocate memory space to process as needed.

[Memory management schemes](https://www.tutorialspoint.com/operating_system/os_memory_management.htm) provide an organization of memory address space and physically allocated process space in memory. Segmentation and paging are the main memory address schemes utilized by most operating systems. The [segmentation](https://en.wikipedia.org/wiki/Memory_segmentation) scheme allows processes to be allocated some variable memory chunks called segments. In segmentation, a memory reference consists of a segment number and an offset within that segment. [The paging](https://en.wikipedia.org/wiki/Memory_paging) scheme allows processes to be allocated of some fixed-sized memory blocks called pages. By using paging, a memory reference consists of a page number and an offset.

[Virtual memory](https://www.tutorialspoint.com/operating_system/os_virtual_memory.htm) allows computers to address more memory than the main memory capacity by utilizing an extended section of the computer’s secondary storage device (hard disk). Using virtual memory, programs can run on the microprocessor while only a program part exists in the main memory and the other part is still on the hard disk storage. Therefore, virtual memory extends the memory addressing by extending the physical memory address to a larger address space that includes both physical and logical memory. Virtual memory is implemented using demand segmentation and demand paging.

Operating systems provide storage activities to allow non-volatile storage of data on secondary storage devices. Each storage device is controlled by a device driver maintained by the computer operating system. Depending on the storage device, operating systems provide two types of data storage: mass storage and file system storage. Operating systems also protect data on these storage devices by using access control mechanisms.

### File System management

[File system management](https://www.tutorialspoint.com/operating_system/os_file_system.htm) organizes data into logical entities (files) in a directory structure and maintains abstract physical properties for these files and directories. Files and directories are system properties that have owners, and their access is controlled by an access control scheme organized by the operating system. Main file management activities include

* Create and delete files and directories
* Provide primitives to manipulate files and directories
* Allocate and organize storage space for files on secondary storage devices

### I/O management

Operating systems provide [I/O management](https://applied-programming.github.io/Operating-Systems-Notes/8-IO-Management/) activities to control and enable user interaction with the computer environment. Through dedicated handlers to the I/O devices implemented in the I/O device drivers as an integral part of a computer operating system. Operating systems allow the I/O devices to raise interrupts, transfer data, and run commands to conduct specific tasks related to their environments. Device drivers provide standard interfaces to deal with I/O devices, maintain their status, and control their memory access. I/O devices are classified by their data access methods into the following:

* Block-oriented I/O devices, e.g., hard disks.
* Character-oriented I/O devices, e.g., keyboards.
* Stream-oriented I/O devices, e.g., network cards.

### Network management

[Network management](https://www.techopedia.com/definition/20974/network-management) is a major integral part of operating systems in which a broad range of functions to facilitate and administrate computer communication with external devices connected through networks are provided. In operating systems, network management is mainly concerned with the provisioning of network operation, configuring network resources, and providing network programming interfaces.

A [network operating system (NOS)](https://www.technologyuk.net/computing/computer-software/operating-systems/network-operating-systems.shtml) is a specialized operating system designed to provide functionality for sharing networked resources and peripheral hardware devices. Examples of NOS include Novell Open Enterprise Server, Microsoft Windows Server. Network operating systems provide the following main functions:

* Network user account management and access control
* Network configuration and resource management
* Support to the network protocol ([TCP/IP](https://www.javatpoint.com/computer-network-tcp-ip-model)) stack
* Network monitoring and maintenance management

### OS Protection and Security

Security is a major component and design constraint of operating systems. [Operating system security](https://www.tutorialspoint.com/operating_system/os_security.htm) aims to protect computer system resources (e.g., CPU, memory, I/O devices, and most importantly data stored in the system) from unauthorized access, malicious activities, and virus infections. Main security components of OS include

* User Identification, Authentication, and authorization
* Access control and account management.
* Virus and malware protection and detection
* Firewall and packet filtering

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_