CHAPTER 6

Intro: System Software & Virtual Machines

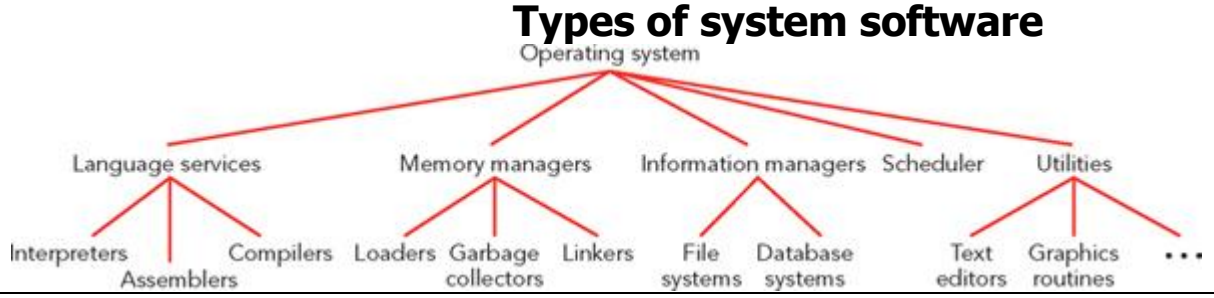
--Naked Machine- Computer without any helpful system software to facilitate its usage

--To make a Von Neumann computer usable, we must create an ***interface*** between the user & the hardware. This interface does the following things:

- Hides from user the messy details of the hardware
- Presents Understandable information about what is happening
- Allows easy user access to the resources available on this computer
- Prevents accidental or intentional damage to hardware, programs, and data

-Systems Software - collection of programs to manage the resources of a computer and facilitates access to those resources.

-***intermediary*** between the users and the hardware



-Virtual Machine- The computer system as perceived by the user vs. hardware that actually exists; services and resources created by the software / seen by the user. Also called a **Virtual Environment**

-Operating System- Main piece of system software; Helps to run and manage the computer system

The software packages that handle these requests include the following:

- ***User interface***—All modern operating systems provide a powerful **graphical user interface (GUI)** that gives the user an intuitive visual overview as well as graphical control of the capabilities and services of the computer.
- ****Language services***—Called ***assemblers***, ***compilers***, & ***interpreters***, allow you to write programs in a high-level, user-oriented language rather than the machine language, often include components such as **text editors** and **debuggers**.
- ****Memory managers***—Programs Allocate memory space for programs and data and retrieve this memory space when it is no longer needed. Loaders/Linkers/Garbage Collectors
- ****Information managers***—Programs Handle the organization, storage, and retrieval of information on storage devices: hard drives, DVDs, flash drives, and tapes, & in data centers (***cloud storage***). -Allow organization of information in an efficient hierarchical manner, using directories, folders, and files and to keep your personal data safe from accidental or intentional misuse.
- ***I/O systems***—Software packages allow you to easily and efficiently use the many different types of input and output devices that exist on a modern computer system.
- ****Scheduler***—System program keeps a list of programs ready to run on the processor, and selects the next one to execute. Allows for several different programs active at a single time, like, surfing the web while you are waiting for a file to finish printing.
- ****Utilities (*****program libraries*****.)***—Collections of library routines for wide range of useful services either to a user or to other system routines. Text editors, online help routines, image and sound applications, & control panels are utility routines.

On the virtual machine created by system software, it is much simpler:

Step	Task
1	Use a text editor to create program P written in a high-level, English-like notation rather than binary.
2	Use an information manager to store program P in a directory on your cloud storage account.
3	Use a language translator to translate program P from a high-level language into an equivalent machine language program M.
4	Use a scheduler to load, schedule, and run program M. The scheduler itself uses the memory manager to obtain memory space for program M.
5	Use the I/O system to display the output of your program on your screen.
6	If the program did not complete successfully, use a debugger to help you locate the error. Use a text editor to correct the program and the information manager to store the newly modified program.

6.3 Assemblers and Assembly Language

6.3.1 Assembly Language

What specifically are the problems with machine language?

- It uses binary. There are no natural language words, mathematical symbols, or other convenient mnemonics to make the language more readable.
- It allows only numeric memory addresses (in binary). A programmer cannot name an instruction or a piece of data and refer to it by name.
- It is difficult to change. If we insert or delete an instruction, all memory addresses following that instruction will change. For example, if we place a new instruction into memory location 503, then the instruction previously in location 503 is now in 504. All references to address 503 must be updated to point to 504. There may be hundreds of such references.
- It is difficult to create data. If a user wants to store a piece of data in memory, he or she must compute the internal binary representation for that data item. These conversion algorithms are complicated and time consuming.

Assembly languages (Low-Level Language) closely related to the machine language**)**were termed ***second-generation languages*** to distinguish them from *machine languages*, which were viewed as ***first-generation languages***.

-source program- The original program as written by the programmer
-object program-The translated source program produced by the assembler or compiler
-assembler- The program that translates a source assembly language program into machine language
-compilers- The program that translates a high-level language into machine language

2 advantages Symbolic labels over numeric addresses: ***maintainability*** & ***program clarity***.

pseudo-op- An assembly language command that does not actually produce a machine-language instruction but performs a service on behalf of the user

algorithmic problem-solving cycle- Sequence of designing an algorithm, coding it into a programming language, translating it into machine language, and then running it on a Von Neumann computer to solve the problem

An ***assembler*** must perform the following four tasks, none of which is particularly difficult.

1. Convert symbolic op codes to binary.
2. Convert symbolic addresses to binary.
3. Perform the assembler services requested by the pseudo-ops.
4. Put the translated instructions into a file for future use.

Assembler makes 2 passes at the Source Code: 1st source code, the assembler looks at every instruction, keeping track of the memory address where this instruction will be stored when it is translated and loaded into memory.

-Symbol Table- A table that contains the name of every symbolic variable in a program and its equivalent binary memory address
-Binding- The process of associating a symbolic name with a physical memory address

-2nd translates source program into machine language. Has op code table to translate mnemonic op codes to binary, & has symbol table to translate symbolic addresses to binary.

-Object File-The file produced by the assembler or compiler that contains the translated machine language instructions and the address of where each instruction is to be loaded

-Loader-System software, Reads instructions from object file and stores them into memory for execution

-System Commands- User Commands to operating system to perform a service on the user's behalf

5 Functions of an Operating System:

- User interface management (a receptionist)
- Control of access to the system and to data files (a security guard)
- Program scheduling and activation (a dispatcher)
- Efficient resource allocation (an efficiency expert)
- Deadlock detection and error detection (a traffic officer)

-Command Language- The language used to enter system commands.
-More commonly a set of actions, such as mouse clicks or finger taps

-Deadlock- A computer is frozen because 2 components are holding a resource needed by the other

-Batch Operating Systems- Type of operating system, batch of programs are collected and then run as a group, all at once, one after the other.

-Multi-programmed Operating Systems- Type of operating system which multiple user programs are loaded into memory at the same time, and the computer takes turns running them

-Time-Sharing System- Multiprogrammed operating system, user can interact w/ running program

A program can keep the processor until *either* of the following two events occurs:

- Initiates an I/O operation.
- Runs for a maximum length of time, called a ***time slice***.

-Network Operating System- operating system that manages the resources of both the local computer as well as providing efficient access to a collection of remote resources via a computer network

-Real-Time Operating System- Operating system that must provide access to particular resources or respond to system problems within a well defined time limit.

-Embedded Systems- Computational device (chip/processor/computer) embedded in another system.

The Future-

-Distributed Computing Environment- A system that hides the exact location of specific pieces of information and allows the user to view the system as simply one large collection of resources.

-Gesture-Based Computing- use of body and facial motions to communicate computational requests

CHAPTER 7
Computer Networks and Cloud Computing

-Computer Network— A set of independent computer systems connected together by telecommunication links for the purpose of sharing information and resources.

-Nodes- Individual computer on a network; also called **host**

Communication Links

-Switched, Dial-Up Telephone Lines- Common way to transmit data in early networking.
-Dial a telephone number, a circuit (i.e., a path) is temporarily established between the caller and callee.

-Modem- Device that *modulates*, alters, standard analog signal, a ***carrier wave***, & encodes binary info.
-a modem performs the inverse operation, which is called *demodulation*.

-Bandwidth- Capacity for transmitting data

-Broadband- Any communication link w/ transmission rate exceeding 256,000 bps, often up to 25 million bps.
Typical broadband options for home users are ***cable modems*** and ***digital subscriber lines (DSL)***

-Digital Subscriber Line (DSL)- Broadband technology, uses the same wires that carry regular telephone signals into home & is provided by either your local telephone company or someone certified as an intermediary. Although it uses the same wires, a DSL signal uses a different set of frequencies, and it transmits digital rather than analog signals.

-Cable Modem- Broadand technology that makes use of the links that deliver cable TV signals into your home, so it is offered by cable TV providers.

-Ethernet- The most widely used broadband technology

-Fast Ethernet- Transmits at 100 Mbps over coaxial cable, fiber-optic cable, or regular twisted-pair copper wire

-Gigabit Networking- Transmission lines that support speeds of 1 billion bits per second (Gbps)

-Gigabit Ethernet Standard- IEEE standard, supports communication on an Ethernet cable at 1,000 Mbps (1 Gpbs)

-IEEE (Institute of Electrical and Electronics Engineers)- International professional society responsible for Developing Industrial Standards in the area of Telecommunications

-Wireless Data Communication- Transmitting data using radio, microwave, and infrared signals

-Mobile Computing- The ability to deliver data to users regardless of their location

-Wireless Local Area Network (WLAN)- Wireless network; User transmits from Computer to a local wireless base station, called a wireless router, no more than a few hundred feet away. This base station is then connected to a traditional wired network, like DSL or cable modem, to provide full Internet access

-Wi-Fi- Wireless transmissions that use IEEE 802.11 standards for local wireless access .
-called **IEEE 802.11 wireless network standard**

-Wi-Fi Hot Spot- Wireless base station (router)

-Bluetooth- A low-power wireless standard used to communicate between devices located quite close to each other, typically no more than 30–50 feet (10–15 meters) apart

-Personal Area Network (PAN)- Collection of privately owned interconnected digital devices all located in close proximity

-Metropolitan Area Network (MAN)- Network whose scope is larger than the few hundred feet of a LAN, typically a few blocks up to an entire city. Its purpose is to provide full Internet connectivity to all computers within a neighborhood or metropolitan area.

-Wireless Wide Area Network (WWAN)- Nodes (Tablet || smart phone) transmit messages to a remote base station provided by a telecommunications company, may be located miles away. Base station, usually a large cellular antenna, placed on top of a tower or building, providing both long-distance voice and data communication services to any system within sight of the tower.

-Wide Area Network (WAN)- Connects devices that are not in close proximity but rather are across town, across the country, or across the ocean

-RFID -Radio Frequency Identifier - A tag that can be attached to || embedded within objects that includes a communications device that can send and receive wireless messages over a network

2Local Area Networks

-Local Area Network (LAN)- Network that connects hardware devices such as computers, printers, and storage devices. All in relatively close proximity, such as in a building or on a single campus

-Bus- (1) path for electrical signals; (2) LAN topology, all nodes are connected to single shared communication line

-Ring- LAN topology, connects the network nodes in a circular fashion, with messages circulating around the ring in either a clockwise or counterclockwise direction until they reach their destination

-Star- LAN topology, single central node connected to all other sites. Central node can route information directly to any other node in the LAN. Messages are 1st sent to central site, then forwards them to correct location.

-Shared Cable- A wire (such as twisted-pair copper wire, coaxial cable, or fiber-optic cable) is strung around and through a building. Users tap into the cable at its nearest point

-Repeater- Device, Amplifies and Forwards a signal

-Bridge- "Smarter" device than repeater. Has knowledge about the nodes located on each separate network. Examines every message to see if it should be forwarded from one network to another

-Switch- Device that allows you to build a LAN by connecting devices to an Ethernet interface in each room

-Broadcasting- Messages are sent by a node on a local area network to every other node on that same LAN

Wide Area Networks *wide area network* (*WAN*) connects devices that are not in close physical proximity.

-Dedicated Point-To-Point Lines- Line or wireless link that directly connects two machines

-Mesh Network-An interconnection system in which a node is connected to other nodes via direct links

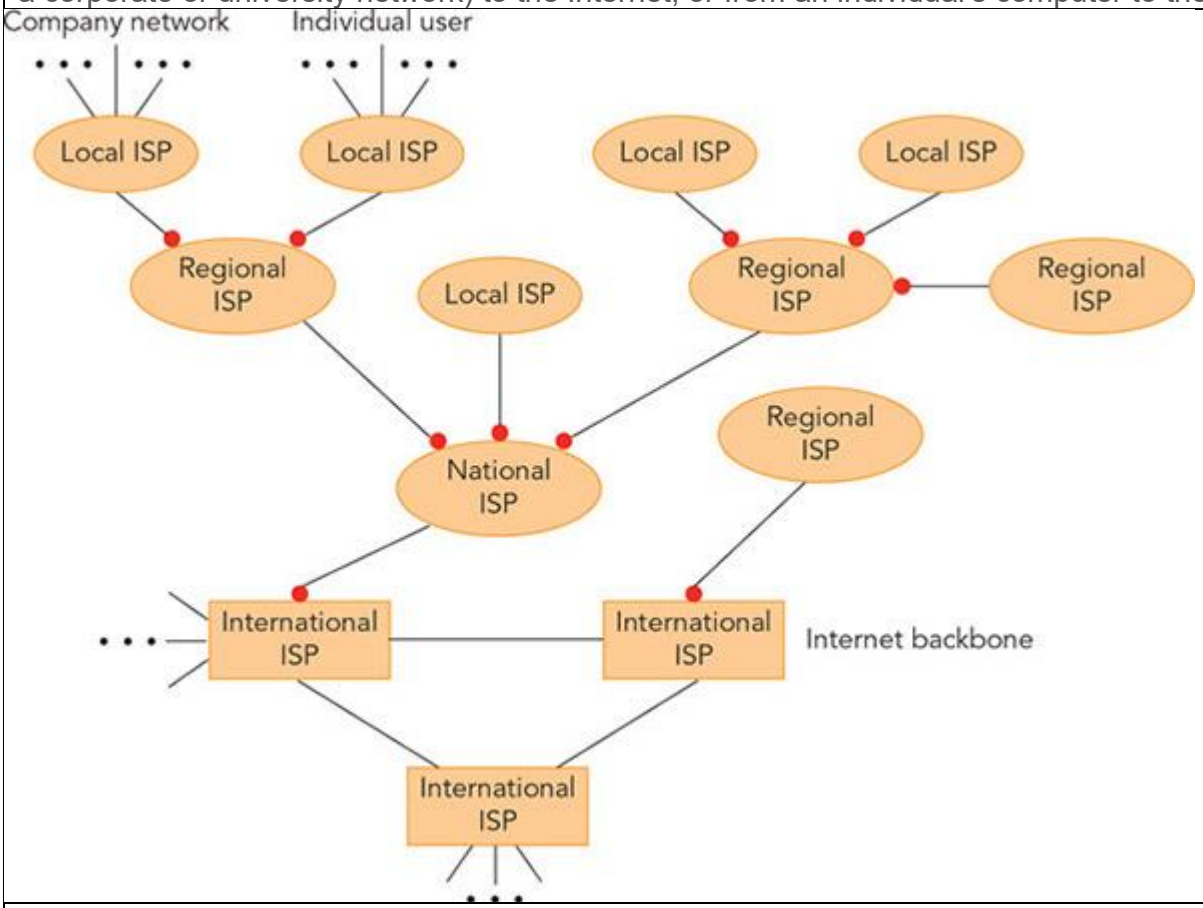
-Store-and-Forward, Packet-Switched- Technology used by WANs to deliver messages that "hop" from one node to another to make their way from source to destination

-Packet- Information block with fixed maximum size, transmitted through the network as a single unit

Overall Structure of the Internet **three classes of networks, LANs, MANs, and WANs**

-Router- Device that connects networks

-Internet service provider (ISP)- Business w/ purpose of providing access from a private network (such as a corporate or university network) to the Internet, or from an individual's computer to the Internet



An individual or a company network connects to a *local ISP*, the first level in the hierarchy. This local ISP typically connects to *regional* or *national ISP* that interconnects all local ISPs in a single geographic region or country. Finally, a regional or national ISP might connect to an *international ISP*, also called a *tier-1 network* or an *Internet backbone*, which provides global coverage.

-Internet- "Network of networks" that includes nodes, LANs, WANs, bridges, routers, and multiple levels of ISPs

-Firewalls- Software or hardware component that controls access from a network to a computer system

These firewalls implement the security guidelines set up by the company, guidelines such as ***address filtering***, specifying which Internet addresses to allow in and which to keep out, and ***service filtering***, listing exactly which applications on your system should or should not be accessed over the network.

Communication Protocols- **protocol**In networking, a mutually agreed upon set of rules, conventions, and agreements for the efficient and orderly exchange of information

Internet SocietyA nonprofit, nongovernmental, professional society composed of more than 100 worldwide organizations (e.g., foundations, governmental agencies, educational institutions, companies) in 180 countries and thousands of individual members united by the common goal of maintaining the viability and health of the Internet

protocol hierarchy(Protocol Stack) The hierarchical set of network protocols that are used to transmit messages across a network

Physical Layer- ***Physical layer protocols***- Govern the exchange of binary digits across a physical communication channel, such as a fiber-optic cable, copper wire, or wireless radio channel.

These protocols specify such things as:

- How we know when a bit is present on the line
- How much time the bit will remain on the line
- Whether the bit is in the form of a digital or an analog signal
- The physical quantities used to represent a binary 0 and a binary 1
- The shape of the physical connector between the computer and transmission line

Data Link Layer

error detection and correction The process of determining whether a message did or did not arrive correctly and, if not, getting a correct copy of that information

framingIdentifying the start and end of a message sent over a network

Medium Access Control protocols Network protocols that determine how to arbitrate ownership of a shared line when multiple nodes want to send messages at the same time.

collisionWhen two or more nodes on a shared link transmit at the same time and destroy each other's message

Logical Link Control protocols Ensure that the message traveling across this channel from source to destination arrives correctly

ARQ algorithm Automatic repeat request algorithm, the basis for all Data Link Control protocols in current use

Acknowledgment message (ACK)-A network control message that says that your message correctly arrived at its destination

Network Layer- Ensure that a message is delivered from the site where it was created to its ultimate destination

the two critical responsibilities of the Network layer are as follows:

- Creating a universal addressing scheme for all network nodes; and
- Delivering messages between any two nodes in the network.

IP- Internet Protocol- The network layer in the Internet

host names The symbolic, character-oriented name assigned to a host computer

IP address The 32-bit or 64-bit binary address that the Internet uses to actually identify a given host computer

Domain Name System (DNS) Internet application that converts from a symbolic host name such as macalester.edu to its 32-bit IP address

routingSelecting one specific path through the Internet among various nodes

Transport Layer

Transport layer protocols Create a "program-to-program" delivery service, in which we don't simply move messages from one host to another, but from a specific program at the source to a specific program at the destination

port numberA numeric identification of a program running on a host computer. It is used by the Transport layer protocols

TCP (Transport Control Protocol) The primary transport protocol on the Internet

Application Layer- **Application layer protocols**The rules for implementing the end-user services provided by a network

Uniform Resource Locator (URL)- A symbolic string that identifies a webpage

Hypertext Transfer Protocol (HTTP) The protocol used by the World Wide Web to transfer pages of information coded in hypertext markup language (HTML)

The following sequence of events now takes place:

1. Your browser scans the URL and extracts the host name of the machine to which it must connect—www.macalester.edu. (Let's disregard the issue of how this symbolic name is converted to its corresponding 32-bit or 128-bit binary IP address.)

2. Your browser asks TCP to establish a connection between itself and port 80 (the web server) of the machine called www.macalester.edu.
3. When the TCP connection between your browser and the web server is established, your browser scans the URL to identify the page you want to access. In this case, it is */about*. Your browser constructs an HTTP 'GET' message, which requests the contents of that webpage. This GET message looks something like the following:

GET /about HTTP/1.2
Host: www.macalester.edu

This message says that we want a copy of the webpage */about* located at www.macalester.edu, and it should be accessed using the http protocol, version 1.2. (An actual GET message is quite a bit more complex and includes a number of additional fields not shown here.)

4. The GET message in Step 3 is transmitted across the Internet from the client's web browser port at the source node to the web server port at the destination node using the services of TCP/IP as well as the Data Link and Physical layer protocols.
5. When the GET message arrives at the destination, it is delivered to the web server (which is listening on port 80). The web server locates the file named in the GET message (*/about*) and creates a response message containing a copy of the contents of that file. This response message looks something like the following:

HTTP/1.2 200
Connection: close
Date: Monday, 26 Mar 2018
Content Length: 53908
Content Type: text/html
… (the actual contents of the webpage go here) …

This response message says that the server successfully found the file (code 200), and it contains 53,908 bytes of text. It also says that after the webpage has been sent, the TCP connection between the browser and the server will be closed. Finally, there is a copy of the entire webpage. (Again, some fields in the response message have been omitted for clarity.)

6. The HTTP response message in Step 5 is transmitted across the Internet from the web server back to the port of the client's web browser using the services of TCP/IP as well as the Data Link and Physical layer protocols.
7. The message is delivered to your web browser, and the page is displayed on the screen. The TCP connection between the two programs is terminated.

Network Services and Benefits

Interpersonal Communications

Email (electronic mail) The single most popular application of networks for the last 35 years

spam Electronic junk mail

bulletin board system(BBS)- Application that flourished in the 1980s and 1990s that used shared public file(s) where anyone could post messages and everyone was free to read the postings of others

1.Internet forums Support the real-time exchange of messages. In addition to simply posting a message that can be read at a later time, they also support interactive messaging–what the sender types appears immediately on the screen of one or more individuals, allowing for the direct exchange of ideas

2. Chat rooms Service that supports real-time exchange of messages over the Internet

3. Texting Also known as SMS (Short Message Service), an application that allows for the rapid exchange of short messages, often using wireless cell phone technology.

Social Networking - Systems that create communities of users who share common interests and activities and that provide multiple methods of online interaction

Resource Sharing- The ability to share physical resources, such as a printer or storage device, as well as logical resources, such as software, data, and information

-Print servers- Shared printer available across a network

-**File server-** Shared data-storage disk(s) available across a network

-**Client/server computing-** The style of computing wherein some nodes provide services while the remaining nodes are users (or clients) of those services

distributed databaseA database distributed among the geographically dispersed sites of an organization and shared as needed

-data warehouse Network nodes that integrate massive amounts of information from a number of disparate sources and allow it to be electronically searched for specific facts or documents

-data martsA single-subject data warehouse

-groupware Software that facilitates the efforts of individuals connected by a network and working on a single shared project

- **wiki**A set of webpages that everyone is free to access, add to, or modify

Electronic Commerce- (**ecommerce**) is a general term applied to any use of computers and networking to support the paperless exchange of goods, information, and services in the commercial sector.

Cloud Computing- A computing system in which the user can be completely unaware of where data is stored and where services are being provided

-has the potential to *lower costs* since users pay for only what they need and what they use, not for what they have to buy.

- provides *elasticity of demand* as your needs fluctuate. Reliability / Improved Security

server farm An integrated collection of servers providing services over a network that would not be possible using only a single device

virtualizationThe separation of a service from the entity (or entities) providing that service

A History of the Internet and the World Wide Web

The Internet-

internetworkingConcept first developed by Robert Kahn of ARPA, that any WAN is free to do whatever it wants internally but at the point where two networks meet, both must use a common addressing scheme and identical protocols

gatewayDevice that makes internetwork connections and provides routing between different WANs

The World Wide Web- The hypertext information system developed by Tim Berners-Lee at CERN in the late 1980s

-hypertextA collection of documents interconnected by pointers, called links

- net neutrality The concept that that a public information network such as an Internet service provider (ISP) should treat all users, all platforms, and all content equally

CHAPTER 8

Information Security- The goal of keeping information protected from those who do not have permission to access it

Threats and Defenses

Authentication and Authorization

Authentication- The process of verifying who has the right to gain access to the computer.

-Encrypts To convert plain text into coded text that cannot be understood without the appropriate decryption algorithm

-Hash function An encryption process that is easy to apply but hard to undo; it takes the password the user originally chooses, chops it up, and stirs it around according to a given formula

-Hacker Originally, someone proficient at tinkering with computers; now someone who breaks into a computer system with malicious intent

-Cyberwarfare Computer attacks from one country on the computing resources of another country with intent to damage or destroy computer systems or steal sensitive information

-Password-cracking softwareA program that automates a brute-force approach to finding a password for a given user ID by encrypting in turn all words in its dictionary using the known hash function

-Social engineering The process of using people to get the information you want

-Password manager (Password Vault)- A central site that securely stores all your passwords in encrypted form, leaving only the one vault password for you to remember

-dual authenticationA second authentication measure after the user enters the user ID and password

Authorization.- Governs what an authenticated user of a computer is allowed to do

-**Access control lists (Permissions)-** Specifies exactly what an authenticated user is allowed to do with a data file or other resource

Depending on who the users are, they have various levels of privilege, such as the following:

- Read access (can read a particular file)
- Write access (can modify a particular file)
- Execute access (can run a particular program file)
- Delete access (can delete a particular file)

Threats from the Network

-Malware Malicious software designed to attack a computer

Virus- A computer program that embeds itself within another program or file in order to infect a host computer and then spread

Worm- Similar to a virus, but can send copies of itself to other nodes on a computer network without having to be carried by an infected host file. **spam**Junk email

Trojan Horse- A computer program that does some legitimate computational task but, unbeknownst to the user, also contains code to perform malicious attacks

-keystroke loggerA hidden program that captures the user's keystrokes, including passwords and credit card numbers, as they are typed

-drive-by download (drive-by exploit)- The process whereby an infected website downloads a Trojan horse to the computer of a user visiting that site.

Denial of Service- denial-of-service (DoS) attack- An attempt to disable a particular website by automatically directing many browsers to that site. If many machines are perpetrating this mischief, it's called a *distributed denial-of-service attack,* or *DDoS*.

-Botnet- (zombie army) - A collection of infected computers under the control of a central site that can direct a denial-of-service attack

-Ransomware Malware that invades a user's computer, encrypts all the files, then demands a ransom payment for the files to be released

-reflection and amplificationA distributed denial-of-service (DDoS) attack that alters the source address of DNS queries so that it appears that the source is the DNS server itself, resulting in flooding data back to the server

Phishing - The practice of sending emails purporting to be legimate to a large number of potential victims in the hopes of luring users to visit a fake website, where personal information can be collected

Defense against the Dark Arts --Make sure things are up to date.

-antivirus software- Software designed to catch any possible virus attacks before they occur or clean up the damage if they do occur

-Firewall- A software or hardware component that controls access from a network to a computer system

-antispyware Software that checks for the presence of a malicious program to capture personal information that has been entered on webpages

-*pop-up* blocker.

-**security patches**An update, generally to the operating system, that adds new or improved security measures

-Be Smart and Informed

White Hats vs. Black Hats

Encryption -Encryption Overview

-Cryptography The science of secret writing / The encrypted message is called *ciphertext*.

-encryption The process of using an algorithm to convert information into a representation that cannot be understood or utilized by anyone without the appropriate decryption algorithm

-decryption The process of reversing the effects of encryption, using an algorithm that converts the encoded ciphertext back into the original unencrypted plaintext.

-symmetric encryption algorithm An encryption algorithm that requires a secret key shared by sender and receiver.

- public key encryption algorithm--> -asymmetric encryption algorithm An encryption algorithm where the encryption key is made public but the decription key, which is different, is known only to the receiver; also known as public-key encryption algorithm.

Simple Encryption Algorithms

Caesar Cipher- An encryption algorithm that shifts each character in the message to another character some fixed distance farther along in the alphabet

-stream cipher Any encryption algorithm that encodes one character at a time

Block Cipher- Any encryption algorithm that encodes a block of characters together, so that each coded character is the result of several plaintext characters

DES - (**Data Encryption Standard**)A symmetric encryption algorithm developed in the 1970s but still widely used

Hiding in Plain Sight // *permutation* (rearrangement).

-Steganography- practice of hiding the very existence of a message, now usually within an image on the web

-AES (Advanced Encryption Standard)A symmetric encryption algorithm that is newer than DES and uses a longer secret key

Public-Key Systems – the encryption key for messages to go to a particular receiver is broadcast to everyone, but the decryption key cannot be derived from it and is known only by the receiver.

RSA- A common public-key encryption algorithm

Web Transmission Security

-SSL (Secure Sockets Layer)A series of protocols to ensure secure data transmission on the web

-TLS (Transport Layer Security) An improved version of SSL

-public-key certificate (digital certificate)- Sent by a server to the client to validate the server's identity

-handshake The exchange of setup information between a client computer and a web server computer preparatory to exchanging real data

-Virtual Private Network (VPN)A service that allows you to send encrypted messages to a VPN provider, who then sends them on identified only by a unique IP number assigned by the VPN provider

two advantages of using a VPN:

1. Your device's IP address is "hidden" behind the IP address assigned by the VPN server, so web traffic cannot easily be connected directly to you, that is, to the device you are using.
2. All network traffic between you & VPN server is encrypted, so at least that leg of the journey is secure.

-Remote Desktop Protocol (RDP) Software that allows a user to see and use the desktop of a remote computer

Embedded Computing-

Embedded computers Computational devices such as chips, processors, or computers that are embedded within another system

CHAPTER 9

Intro: High-Level Language Programming-

9.1The Language Progression

Where Do We Stand and What Do We Want?

To summarize, assembly language has the following disadvantages:

- The programmer must "manually" manage the movement of data items between and among memory locations and registers (although such data items can be assigned mnemonic names).
- The programmer must take a microscopic view of a task, breaking it down into tiny subtasks—ADD, COMPARE, INCREMENT—that direct what is going on in individual memory locations and registers.
- An assembly language program is machine specific.
- Statements are not natural-language-like (although operations are given mnemonic code words as an improvement over a string of bits).

we have the following expectations of a program written in a high-level language:

- The programmer need not manage the details of the movement of data items within memory or pay any attention to exactly where those items are stored.
- The programmer can take a macroscopic view of tasks, thinking at a higher level of problem solving (add *B* and *C*, and call the result *A*). The "primitive operations" used as building blocks in algorithm construction (see Chapter 1) can be larger.
- Programs are portable rather than machine specific.
- Programming statements are closer to natural language and use standard mathematical notation.

-third-generation languages Another name for **high-level programming** language as opposed to **machine language** (first generation) or **assembly language** (second generation)

9.1.2Getting Back to Binary

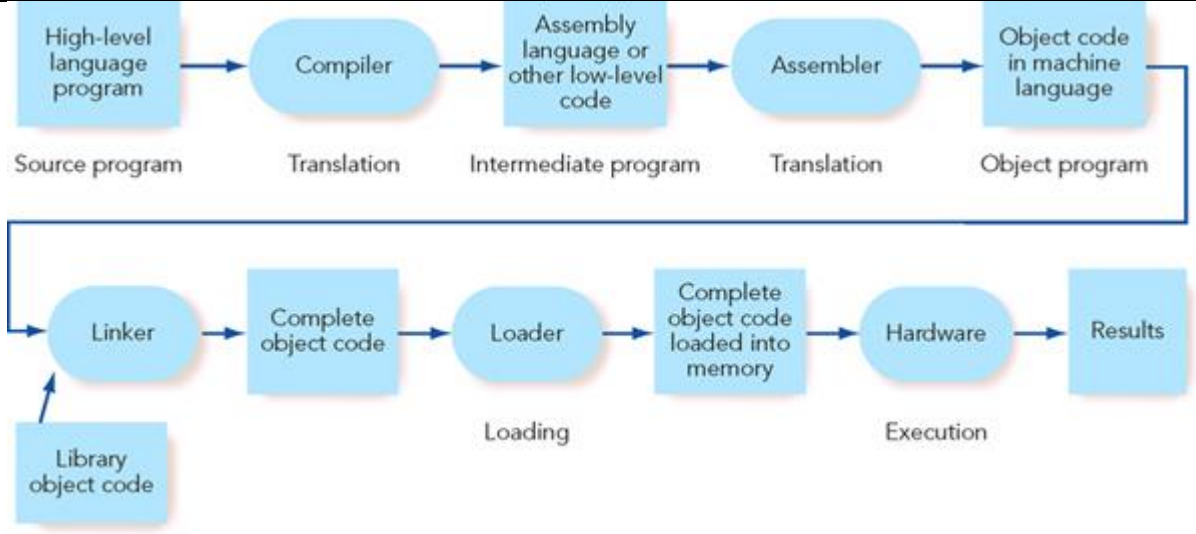
-object code Machine language instructions

-source code High-level language instructions

-Code Library Collection of thoroughly tested object code for various useful tasks

-LinkerA piece of system software that inserts requested object code from code libraries into the object code for the requesting program

-Executable module The resulting object code after a linker inserts requested code from code libraries



-

9.2A Family of Languages

-procedural languages- Programming language in which programs consist of sequences of statements that manipulate data items - (also called **imperative languages**)

Ada, C++, C#, Java, and Python Online Chapters

-Syntax The rules for exactly how programming statements must be written; the grammatical structure of a programming language

-**Semantics**The meaning of correctly written programming statements

9.5Meeting Expectations

-open source programming languageA programming language developed on the open source model, in which the source code is freely available and many individuals can contribute to and modify the code

9.6The Big Picture: Software Engineering

- software development life cycle The overall sequence of steps needed to complete a large-scale software project **software development life cycle STEPS:**

Before Implementation

1. **Feasibility study**
2. **Problem specification**
3. **Program design**
4. **Algorithm selection or development, and analysis**

Implementation

5. **Coding**
6. **Debugging**

After Implementation

7. **Testing, verification, and benchmarking**
8. **Documentation**
9. **Maintenance**

-software engineering An organized process for large-scale software development

9.6.2The Software Development Life Cycle

-feasibility study- A step in the software development life cycle that evaluates a proposed project and compares the costs and benefits of various solutions

-**Problem specification**A step in the software development life cycle that involves developing a clear, concise, and unambiguous statement of the exact problem the software is to solve.

-**program design phase**A step in the software development life cycle that plans the structure of the software to be written

--**divide-and-conquer**A program design strategy in which tasks are broken down into subtasks, which are broken down into sub-subtasks, and so on, until each piece is small enough to code comfortably. These pieces work together to accomplish the total job (also called **top-down decomposition**)

--**object-oriented programming**A programming approach that identifies appropriate objects, together with their data and the subtasks they must perform

--***program design document*** breaks the problem into subtasks and sub-subtasks, or into various classes.

-*Algorithm selection or development, and analysis*—Once the various subtasks have been identified, algorithms must be found to carry them out. If there is a choice of algorithms, the programmer must determine which is more suitable for this particular task and which is more efficient.

-**Coding**The process of translating the detailed designs into computer code. reusable code can be pulled from a program library, or a useful class can be employed.

-**Debugging**The process of locating and correcting program errors

--**syntax errors**Error that occurs because a program statement fails to follow the correct rules of syntax

--**runtime errors**Error that occurs when the program is run using certain sets of data that result in some illegal operation, such as dividing by zero

--**logic errors**Error in the algorithm used to solve a problem

-***Testing, verification, and benchmarking***—

--**empirical testing**Designing a special set of test cases and running the program using this test data

--**test suite**A set of special test data chosen to help determine the correctness of a software project

--**Unit testing**Testing each module of code as it is completed

--**integration testing**After unit testing, integration testing is done to see that the modules communicate the necessary data between and among themselves and that all modules work together smoothly

--**regression testing**If anything is changed on an already-tested module, regression testing is done to be sure that this change hasn't introduced a new error into code that was previously correct

--**Program verification**Used to prove that if the input data to a program satisfies certain conditions, then, after the program has been run on this data, the output data satisfies certain other conditions

--***Benchmarking***Running a program on many data sets to be sure its performance falls within required limits; timing the same algorithm on two different machines

-*Documentation*

--**internal documentation**Documentation that is part of the program code itself

--**External documentation**Any materials assembled to clarify the program's design and implementation

--**Technical documentation**Documentation that enables programmers who later have to modify the program to understand the code

--**User documentation**Documentation that helps users run the program

-**Program maintenance**The process of adapting an existing software product due to errors, new system requirements, or changing user needs

9.6.3Modern Environments

Development Environments. –

-integrated development environment, or IDEA collection of programs that support software development, such as debuggers, editors, toolkits, and libraries, that lets programmers perform several tasks within the shell of a single application

-prototypes In software development, sample graphical user interfaces (GUIs) created early in the software development process in order to get user input

-rapid prototyping The process of developing an early graphical user interface for a program in order to get feedback from the user

Code Repositories. -

The collection of versions of code, together with details about change,s that is maintained by a version control system

-version control systemA system for managing versions of a large program as changes are made, usually by multiple developers

9.6.4 Agile Software Development

-waterfall model The traditional model of software development that follows a series of steps. 1ST-LAST w/NO Flexibility. Prone to Risk and Failure because there is no product to test until the end.

-agile software development An approach to software development that emphasizes a flexible and ready response to meet a shifting target

-Pair programming Involves two programmers at a single workstation. At any given point in time, one is writing code and the other is actively observing, watching for possible errors but also thinking about the overall approach.

CHAPTER 10

The Tower of Babel: Programming Languages-

10.2 Procedural Language

-procedural languages- Programming language in which programs consist of sequences of statements that manipulate data items - (also called **imperative languages**)

10.2.1 Plankalkül- As the very first attempt to design a high-level programming language for computers, built a computer in Germany during World War II.

10.2.2 Fortran- derives from ***For*mula *Trans*lating System.**

-external libraries- Well-written, efficient, and thoroughly tested code module that is separately compiled and then drawn on by any program that wishes to use its capabilities

10.2.3 COBOL- COBOL derives from ***CO*mmon *B*usiness-*O*riented *L*anguage.** -USED FOR: managing inventories and payrolls.

-legacy code Old computer code that is still in use

10.2.4 C/C++ - used language for writing system software because it combines the power of a high-level language with the ability to circumvent that level of abstraction and work at the assembly-language-like level.

-includes a data type called ***pointer***.

-**device driver**-A program to interact with an I/O device

10.2.5 Ada -~1975- U.S. armed services set about trying to develop a common high-level programming language for use by defense contractors.

- Not only by the defense industry, where its use was mandated by the U.S. Department of Defense, but also for other technological applications and as a general-purpose language as well. Ada is known for its multiprocessing capability—the ability to allow multiple tasks to execute independently and then synchronize and communicate when directed. It is also known as a strongly object-oriented language.

-**SPARK 2014** is a programming language that is a well-defined subset of Ada 2012.

10.2.6 Java -designed to be small, inexpensive, easy to use, reliable, and equipped with software that could function over the multiple hardware platforms of the consumer electronics market at that time.

-Java applications- Complete standalone program that resides and runs on a self-contained computer; in general, any computer program written to perform a useful task.

-**applets**(Java) A small application, designed to run from webpages
-Java Web Start (JAWS)- A process whereby Java code is accessed from the web but, for security reasons,is executed in a restricted environment outside the web browser itself
- bytecode(Java) Low-level code that can be easily translated into any specific machine language
- **Java bytecode interpreter**Software that translates bytecode into machine language and executes it

10.2.7Python

-Ease of Use - an ***interpreted language***, meaning it is translated from source to object code at every execution. Python was originally used for system administration tasks and as a web interface language. But with the development of an extensive library of supporting code, Python has become a powerful language for more general use. As an example, SciPy is an open-source Python library with code designed to support scientific and engineering computing needs.

10.2.8C# and .NET-

- designed to make some improvements over C++ in safe usage, and it shares many features with Java.
- **garbage collection**Reclaiming memory no longer needed by a program
- **Microsoft .NET Framework**A collection of tools for software development designed so that traditional text-based applications, GUI applications, and Web-based programs can all be built with equal ease
- **Microsoft Intermediate Language (MSIL)**Low-level code for a .NET language program that can be easily translated into any specific machine language
- **just-in-time (JIT) compiler**Part of the Microsoft .NET framework that compiles MSIL code into object code on the user's machine

10.3Special-Purpose Languages

-designed for only one particular task. These four are merely representative; many other specialized languages exist.

10.3.1SQL

- ***Structured Query Language***. SQL is designed to be used with databases
- An SQL query does not give specific directions as to how to retrieve the desired result.

10.3.2HTML

-***H*yper*T*ext *M*arkup *L*anguage**. used to create HTML documents that, when viewed with web browser software, become **webpages**.
-Tags- Special characters in HTML and other markup languages that achieve formatting, special effects, and links to other documents or webpages

10.3.3JavaScript

- **scripting language** is a "lightweight" language that is interpreted (translated, then executed, statement by statement).
- event handlers Code that responds to a particular "event," often a user action
- ***XML*** (**e*X*tensible *M*arkup *L*anguage**) is a newer markup language. It is a "metalanguage," that is, a markup language for markup languages. Using XML, the writer can create his or her own tags;

10.3.4R

- a specialized programming language designed for statistical computing and graphics.
-code libraries Collection of thoroughly tested object code for various useful tasks

10.4Alternative Programming Paradigms

-Paradigm- A model or mental framework for representing or thinking about something
--Programming in a procedural language consists of 1. Planning the algorithm &
2. Capturing the "unambiguous and effectively computable operations" as program instructions

--Alternative paradigms for programming languages include viewing a program's actions as:

- A combination of various transformations on items (functional programming)
- A series of logical deductions from known facts (logic programming)
- Multiple subtasks of the same problem being performed simultaneously by different processors (**parallel programming**)

10.4.1 Functional Programming

- Views every task in terms of functions. In this context, **function** means something like a mathematical function—a recipe for taking an argument (or possibly several arguments) and doing something with them to compute a single value

-Certain functions, called *primitive functions* or just *primitives,* are defined as part of the language.

- applicative languages Another name for a functional programming language, so called because it repeatedly applies functions

- recursive Something that is defined in terms of "smaller versions" of itself, as in a recursive function

- side effect Occurs when a function, in the course of acting on its argument values to produce a result value, also changes other values that it has no business changing

10.4.2 Logic Programming

- A language that, based on facts that are asserted to be true, can infer or deduce other facts; sometimes called **declarative language**

-Prolog fact -Statement that expresses a property about a single object or a relationship among several objects. Prolog programs consist of *facts* and *rules*.

-Prolog ruleA declaration of an "if A then B" form, which means that if A is true (A is a fact), then B is also true (B is a fact)

- knowledge base(Prolog) Facts and rules about a certain domain of interest

- inference engine Software that is supplied as part of an expert system or as part of the compiler or interpreter for a logic programming language that can access a knowledge base, and contains its own rules of deductive reasoning based on symbolic logic. Also called a query interpreter

- modus ponensA rule of deductive reasoning that states that "if A then B" together with "A" must result in "B"

10.4.3 Parallel Programming

-Parallel processing- a catchall term for a variety of approaches to computing architectures and algorithm design.

-Multicore Computing In which two or more processors are packaged together on a single integrated circuit

-Cluster Computing In which independent systems such as mainframes, desktops, or laptops are interconnected by a local area network (LAN) like the Ethernet or a wide area network (WAN) such as the Internet

- divide-and-conquer modelA problem-solving approach that successively partitions the problem into smaller and smaller parts; used in MIMD processing to divide tasks among multiple processors

10.5 New Languages Keep Coming-

new languages and new paradigms continue to be developed, in the hope of the "silver bullet" of an easy-to-use, efficient, general-purpose computing language that enforces good programming practices to allow quick production of error-free code on all computing platforms.

10.5.1 Go

- ***Golang***, a programming language developed at **GOOGLE**. -Like C++

-Dropbox, Uber, Netflix use Go. - *Efficient compilation / Efficient execution / ease of programming;*

10.5.2 Swift -

APPLE designed for building apps on the iOS and macOS operating systems, in others words, for the iPhone/iPad and the Mac.

-Like Objective-C , but adds modern programming constructs & makes it harder to do "unsafe" things.

10.5.3 Milk

-Created by MIT's Computer Science and Artificial Intelligence Laboratory

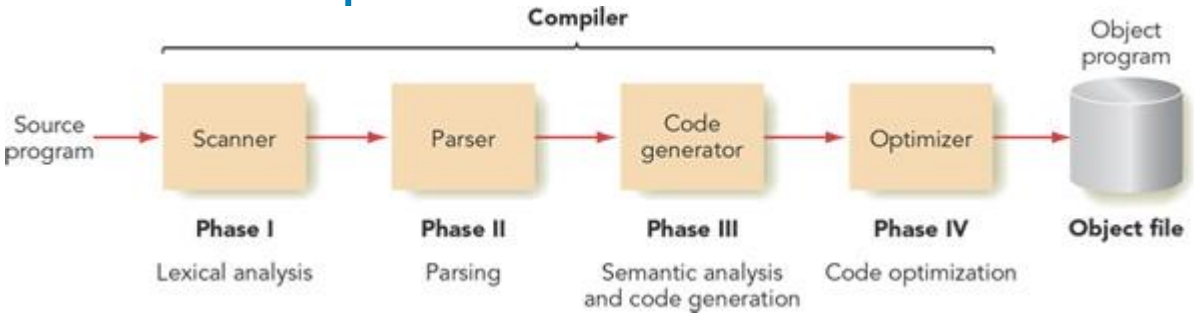
- geared to "big data" an interest in many applications, attempts to detect connections and patterns between data points.

-Assumes multicore processors use with local cache memory. Core processor instructed to fetch a data item from main memory, First, adds memory address item to a growing list in its local cache.
- The benefit appears to be that programs run 3-4 times faster.

CHAPTER 11
Compilers and Language Translation-

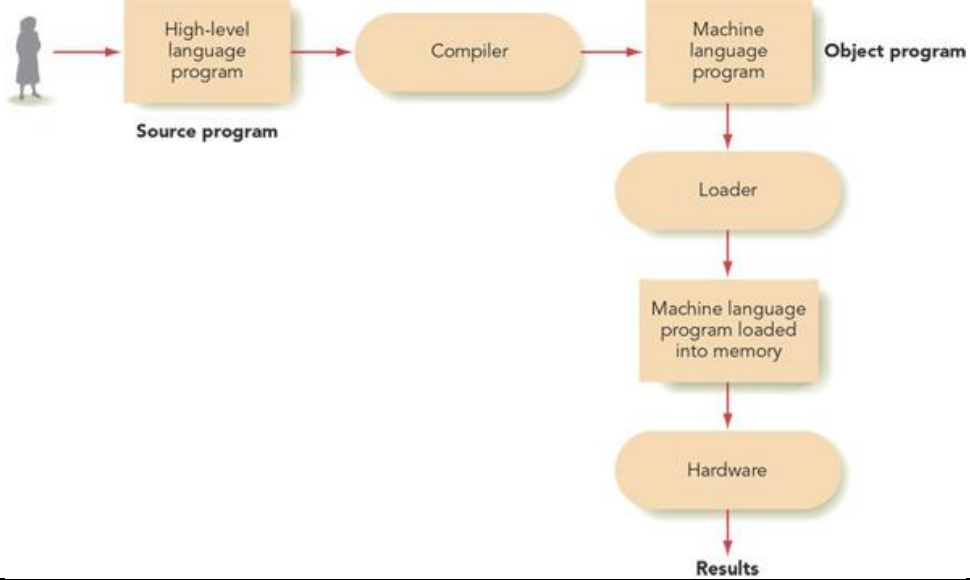
-High-level languages must also be translated into machine language prior to execution—in this case by a special piece of system software called a ***compiler***.

11.2 The Compilation Process



The four phases of compilation listed in Figure 11.1 are the following:

- ***Phase I: Lexical analysis***—Compiler examines individual characters in program / groups them into syntactical units, called **tokens**, that will be analyzed in succeeding stages. This operation is analogous to grouping letters into words prior to analyzing natural language text.
- ***Phase II: Parsing***—The sequence of tokens formed by the scanner is checked to see whether it is syntactically correct according to the rules of the programming language. This phase is roughly equivalent to checking whether individual words in a natural language text are connected together in a way that forms grammatically correct sentences.
- ***Phase III: Semantic analysis and code generation***—If the high-level language statement is structurally correct, then the compiler analyzes its meaning and generates the proper sequence of machine language instructions to carry out the intended actions.
- ***Phase IV: Code optimization***—The compiler takes the generated code and sees whether it can be made more efficient, either by making it run faster, having it occupy less memory, or possibly both.



11.2.1 Phase I: Lexical Analysis

-Lexical analysis The process of grouping individual characters into grammatical units called tokens

-**Lexical analyzer (Scanner) -** Program that performs lexical analysis

-**Tokens-** Syntactical unit, treated as a single entity for the purposes of translation -Operators/Operands

-what a scanner must do: *The input to a scanner is a high-level language statement from the source program. Its output is a list of all the tokens contained in that statement, as well as the classification number of each token found.*

- every scanner performs virtually the same set of operations:

(1)It discards blanks and other nonessential characters and looks for the beginning of a token;
(2)when it finds the beginning, it puts characters together until
(3)Detects end of the token / classifies the token and begins looking for the next one.

-(1)Discard blanks until you find a nonblank character;
-(2)group characters together until
-(3)you encounter either a blank or the character ".".

11.2.2 Phase II: Parsing

-parse tree- A structure that starts from the individual tokens in a statement and shows how these tokens can be grouped into predefined grammatical categories

-parsing The process of diagramming a high-level language statement and building a parse tree

-parser A program that parses high-level language statements

Grammars, Languages, and BNF. -

The parser must be given a formal description of the *syntax*—the grammatical structure—of the language that it is going to analyze.

-BNF (Backus-Naur Form) - Most widely used notation, represents syntax of a programming language

-Rules (productions)- A description of how to group syntactic elements in a programming language to produce a new grammatical construct; also called production

- grammar The collection of all rules that define the syntax of a programming language

- BNF rules use two different types of objects, called **terminals** and nonterminals

-**Terminals** In parsing, an actual token of the language recognized and returned by a scanner

-nonterminal In parsing, not an actual element of the language but an intermediate grammatical category used to help explain and organize the language

-goal symbol In parsing, the final nonterminal; the nonterminal object that the parser is trying to produce as it -builds the parse tree

-language defined by a grammar The set of all sentences that can be successfully parsed by a grammar

- terminals never appear on the left-hand side of a BNF rule, whereas nonterminals must appear on the left-hand side of one or more rules.

- metasymbols Symbol of one language, such as BNF, that is used to describe the characteristics of another language

-null string A string that is empty (contains nothing)

Parsing Concepts and Techniques.

-fundamental rule of parsing follows.

If, by repeated applications of the rules of the grammar, a parser CAN convert the complete sequence of input tokens into the goal symbol, then that sequence of tokens is a syntactically valid statement of the language. If it CANNOT convert the input tokens into the goal symbol, then this is not a syntactically valid statement of the language.

- look-ahead parsing algorithms A parsing algorithm that examines the upcoming tokens to see what would happen if a certain choice is made. It does this to try to avoid making incorrect choices

Biggest problems building a Compiler for a programming language. Designing a grammar that:

- Includes every valid statement that we want to include in the language, and
- Excludes every invalid statement that we do not want to include in the language.

- recursive definition In BNF, the definition of a grammatical element in terms of itself with an *arbitrary and unbounded* number.

-A parse tree serves to prove a statement is correct / assigns a specific *meaning*, or *interpretation*.

-Ambiguous- Grammar that allows the construction of 2+ distinct parse trees for the same statement

11.2.3 Phase III: Semantics and Code Generation

- The next phase of translation, during which a compiler examines the semantics of a programming language statement, deals with this issue. It analyzes the meaning of the tokens and tries to understand the *actions* they perform. If the statement is meaningless, as "bees bark." is, then it is semantically rejected, even though it is syntactically correct. If the statement is meaningful, then the compiler translates it into machine language.

-semantic records A data structure that stores information about a nonterminal, such as the actual name of the object and its data type

-semantic analysis A pass over the parse tree to determine whether all branches of the tree are semantically valid

-code generation A second pass over the parse tree, not to determine correctness but to produce the translated code.

11.2.4 Phase IV: Code Optimization

-From Assembly Language to FORTRAN- with efficiency—the ability to write highly optimized programs that contained no wasted microseconds or unnecessary memory cells.

Used to Increase Programmer Productivity

-code optimization Polishing and fine-tuning code so that it runs a little faster or occupies a little less memory

-visual development environments Software development tools that use graphics and video to let the programmer see what is happening

-online debuggers Web-based tools to help programmers locate and correct errors

-and reusable code libraries, which contain a large collection of prewritten and fully debugged program units.

- When a compiler is embedded within a collection of supporting software development routines such as debuggers, editors, toolkits, and libraries, it is called an integrated development environment (IDE).

- A little bit of effort by a compiler can often pay large dividends in reduced memory space and lower running time.

-There are two types of compiler optimizations: local optimization and global optimization.

-local optimization The compiler looks at a very small block of instructions, typically from one to five, and tries to determine how it can improve the efficiency of this local code block without regard for what instructions come before or after

3 possible local optimizations:

1. **Constant evaluation**-If possible, arithmetic expressions are evaluated at compile time vs. execution time
2. **Strength reduction-** Slow arithmetic operations is replaced with faster ones
3. **Eliminating unnecessary operations**- correct, Unnecessary instructions are discarded

---

-global optimization- Rare & Expensive. Compiler looks at large segments of the program, not just small pieces, to determine how to improve performance

"Now I Understand," Said the Machine

-natural language understanding Getting computers to understand and use natural language

CHAPTER 12

Models of Computation-

12.2 What Is a Model?

1. Captures the essence—the important properties—of the real thing;
2. Probably differs in scale from the real thing;
3. Omits some of the details of the real thing; and
4. Lacks the full functionality of the real thing.

12.3 A Model of a Computing Agent

12.3.1 Properties of a Computing Agent

To summarize, we require that **any computing agent be able to do all of the following:**

1. Accept input.
2. Store information in memory and retrieve it from memory.
3. Take actions according to algorithm instructions; the choice of what action to take may depend on the present state of the computing agent, as well as on the input item presently being processed.
4. Produce output.

12.3.2 The Turing Machine - theoretical model of a computing agent in which symbols are written in cells of a hypothetical infinite tape and are read and changed by a read/write unit according to the input symbol, the state of the read/write head, and the set of rules for the Turing machine.

-tape alphabet - The allowable symbols that can be written on a Turing machine tape

-The Turing machine is designed to carry out only one type of primitive operation.

Each time such an operation is done, three actions take place:

1. Write a symbol in the cell (replacing the symbol already there).
2. Go into a new state (it might be the same as the current state).
3. Move the "read head" one cell left or right.

-Turing machine instructions Given the current tape symbol and the state of the read/write unit, describes the symbol to write, the next state, and the direction of move of the read/write unit

Shorthand notation for Turing machine instructions. 5 components:

Current state / Current symbol / Next symbol / Next state / Direction of move

If no rule for current state & input, Machine HALT.

how well does the Turing machine stack up against our list of required features for a computing agent?

1. *It can accept input*—The Turing machine can read symbols on its tape.
2. *It can store information in memory and retrieve it from memory*—The Turing machine can write symbols on its tape and, by moving around over the tape, can go back and read those symbols at a later time. The tape serves as the Turing machine memory.
3. *It can take actions according to algorithm instructions, and the choice of action to take may depend on the present state of the computing agent and on the input item presently being processed*—Certainly the Turing machine satisfies this requirement insofar as Turing machine instructions are concerned; the present state and present symbol being processed determine the appropriate instruction, and that instruction specifies the actions to be taken.
4. *It can produce output*—The Turing machine writes symbols on its tape in the course of its normal operation. If (when?) the Turing machine halts, what is written on the tape at that time can be considered output.

-Turing machine program A set of Turing machine instructions that allows a Turing machine to carry out a certain task - there is *no limit* to the amount of memory available to the machine.

12.4 A Model of an Algorithm

an algorithm must

1. Be a well-ordered collection;
2. Consist of unambiguous and effectively computable operations;
3. Halt in a finite amount of time; and
4. Produce a result.

Collection of Turing machine instructions and see whether it exhibits these properties of an algorithm.

1. *Be a well-ordered collection*—The Turing machine must know which operation to carry out first and which to do next at any step. We have already specified the initial conditions for a Turing machine computation: that the Turing machine must begin in state 1, reading the leftmost nonblank cell on the tape. We have also insisted that in any collection of Turing machine instructions, there cannot be two different instructions that both begin with the same current state and current symbol. Given this requirement, there is never any confusion about which operation to do next. There is *at most* one instruction that matches the current state and current symbol of the Turing machine. If there is one instruction, the Turing machine executes the operation that instruction describes. If there is no instruction, the Turing machine halts.
2. *Consist of unambiguous and effectively computable operations*—Recall that this property is *relative* to the computing agent; that is, operations must be understandable and doable by the computing agent. Each individual Turing machine instruction describes an operation that (to the Turing machine) is unambiguous, requiring no additional explanation, and any Turing machine is able to carry out the operation described. After all, Turing machine instructions were explicitly designed for Turing machines to execute.
3. *Halt in a finite amount of time*—For a Turing machine to halt when executing a collection of instructions, it must reach a configuration where no appropriate instruction exists. This depends on the input given to the Turing machine—that is, the contents initially written on the tape. Consider the following set of Turing machine instructions:

4. *Produce a result*—We have already imposed the requirement that the Turing machine instructions must lead to a halting configuration when executed on input appropriate to the problem being solved. Whatever is written on the tape when the machine halts is the result.

12.5 Turing Machine Examples

12.5.1 A Bit Inverter

-state diagram A visual representation of a Turing machine algorithm

12.5.2 A Parity Bit Machine

-An extra bit, called ***odd parity bit***, can be attached to the end of a string of bits. The odd parity bit is the total number of 1s in the whole string of bits, including the parity bit, is odd.

-Parity bits A single bit added to the end of a binary string to set the total number of 1s, including the parity bit, to an even or odd number (even or odd parity bit), which allows detection of single-bit errors in transmission

12.5.3 Machines for Unary Incrementing

-unary representation Using only one symbol to represent whole numbers

12.5.4 A Unary Addition Machine- written to add two numbers

12.6 The Church–Turing Thesis - The thesis that if there exists an algorithm to do a symbol manipulation task, then there exists a Turing machine to do that task

-Turing Award The most prestigious technical award given by the Association for Computing Machinery; it is for "contributions of lasting and major technical importance to the computer field"

-computability That which can be done by symbol manipulation algorithms

-uncomputable or unsolvable A problem for which no solution algorithm exists

12.7 Unsolvable Problems

-halting problem Decide, given any collection of Turing machine instructions together with any initial tape contents, whether that Turing machine will ever halt if started on that tape

-proof by contradiction Assume the opposite of what you want to prove and follow the logical consequences until you reach a contradiction, thereby showing that your assumption is incorrect

Unsolvable problems, related to the halting problem, have the following consequences:

- No program can be written to decide whether any given program always stops eventually, no matter what the input.
- No program can be written to decide whether any two programs are equivalent (will produce the same output for all inputs).
- No program can be written to decide whether any given program run on any given input will ever produce some specific output.

Chapter 12 Class notes

-**Model**: MOST IMPORTANT: Has to capture the essence of the real thing.
-Models predict behavior.
-Models can be tested without building it. Created at a much smaller scale.
-EX: Graphical User Interface
-

TEST STUFF -> 2/3 ?'s per chapter (Short Answer)– 2 Python programs. Learning Objectives. There are Slides???

CH6- Differnt types of software. Assembly vs. High-level lang. vs. Machine language.
How assembly translates
What is an OS/how's it used. Virtualized Environments? Convert Code into Assembly
CH7- Different technologies (dsl/broadband/wireless). Network types. Topologies(good/bad characteristics).
5 layers of network model. Cloud Computing? Internet (Sum of ISPs/Networks) vs. World-wide web(HTML/Helped share information/The Web browser = HTML).
CH8-Authentication (Username/Password) vs Authorization (Permissions). Use of hash function to encrypt. Types of cyber-attacks & what they do. Ways to increase security. What are ciphers(DES/RSA). Embedded systems security(Hacked through thermostat).
CH 9.6 -Steps of Software Development life cycles(Waterfall vs. Agile). Cost of change of Waterfall = Increase of Time
CH10-Purpose & Differences of Languages. Procedural / Special-Purpose / Functional / Logic languages. MIMD Model for Parallel processing.
CH11-Compiler & how it works. Stages of Compiler. Difference of Syntax(Structure) & Semantics(Grammar). Code Optimization (Less needed today)
CH12- what is a Turing machine, char of computing agent, is Turing machine a valid model. What is a model/how its used.