



Master of International Business and Entrepreneurship

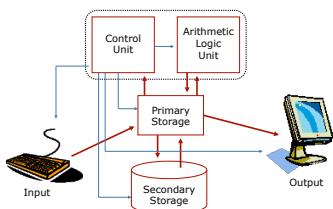
## Software Core Concepts

Information Systems for Managers

## Learning Objectives

- ❖ Be able to define what software is, and be familiar with the different classes of software in use today.
- ❖ Be able to define what an interface is, and be familiar with the different types of interfaces in use today.
- ❖ Be familiar with the process of software creation.
- ❖ Be able to define what an algorithm is and be familiar with the different types of programming language in use today.
- ❖ Understand the underlying logical structure of modern software applications.

## The Stored Program Concept

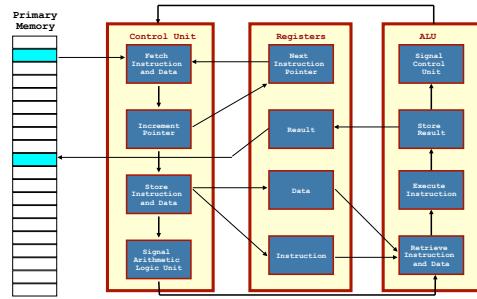
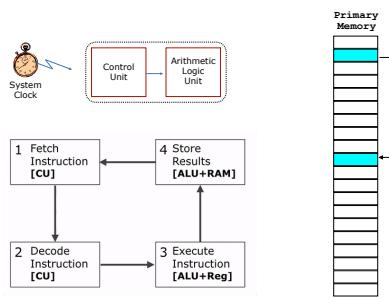


‘ The instructions which govern this operation must be given to the device in absolutely exhaustive detail. [...] Once these instructions are given to the device, it must be able to carry them out completely without any need for further intelligent human intervention. ’



Photography by designhomse.it  
Engineering by Anton Georgiev

## The Processing Cycle



## Instruction Set

- ❖ Can you upgrade the instruction set of your iPhone?
- ❖ Can you explain the difference between a software update and an instruction set change?



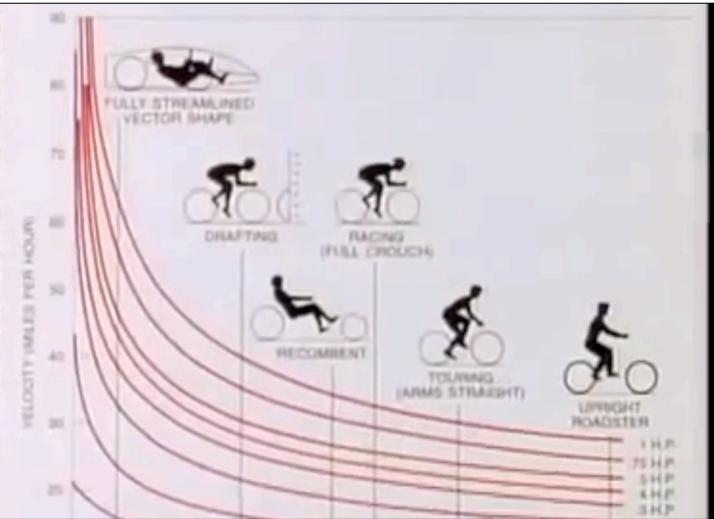
## The Stored Program Concept

- ❖ Digital computers are multi-purpose devices
- ❖ Humans and computers are complementary
  - ❖ Humans are creative problems solvers
  - ❖ Computers execute tasks with speed and precision
- ❖ A computer is “like a bicycle for the mind”

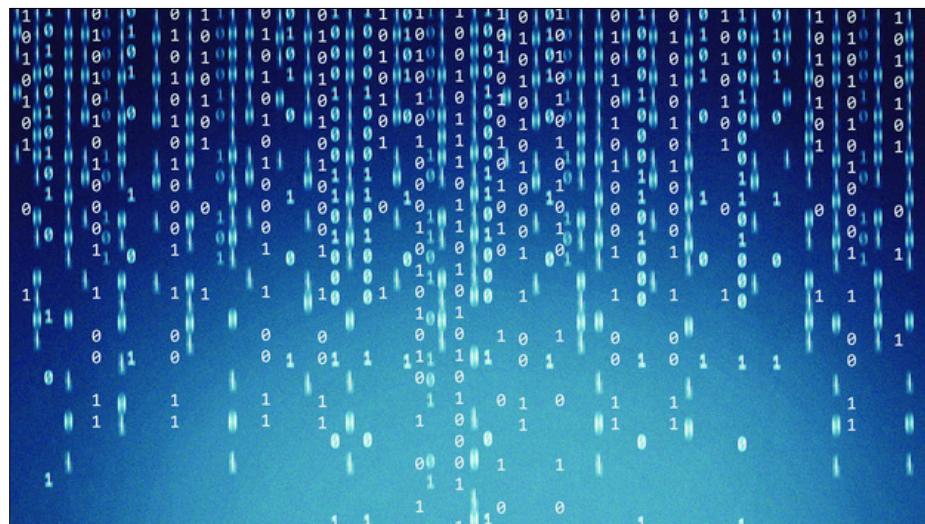
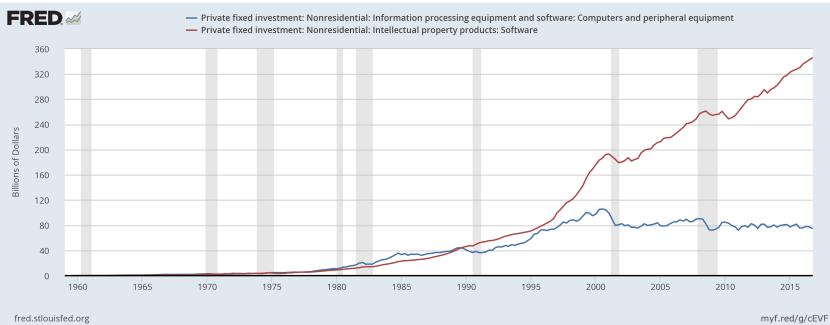


In pursuit of performance, a crouched rider on a racing bicycle could attain a velocity of about 34 m.p.h. per input of 1 h.p. On the same rider, making a turn could achieve 38 m.p.h. In the world of noncompetitive cycling, one need only want a more efficient aerodynamic profile. They can be had at a cost, beginning with aero fairing such as the one and manufactured in Santa Cruz, Calif. It is a front, streamlined shield for the rider. For about \$100, the aerodynamic efficiency is improved by 10 percent, achieving a velocity of some 2.5 m.p.h. for a given power output.

One way of reducing drag is to ride a recumbent bicycle. This would cost several thousand dollars more than a basic bike. The gains, however,

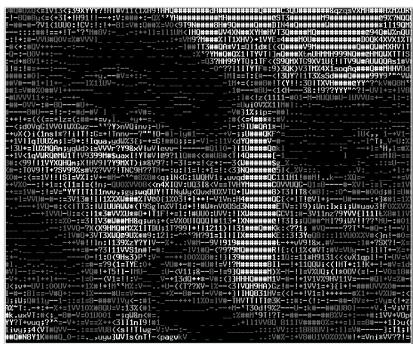


## The Software Industry



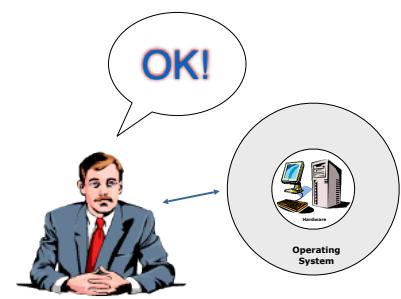
## Classes of Software Programs

- ❖ Operating System
- ❖ System Software
- ❖ Application Software

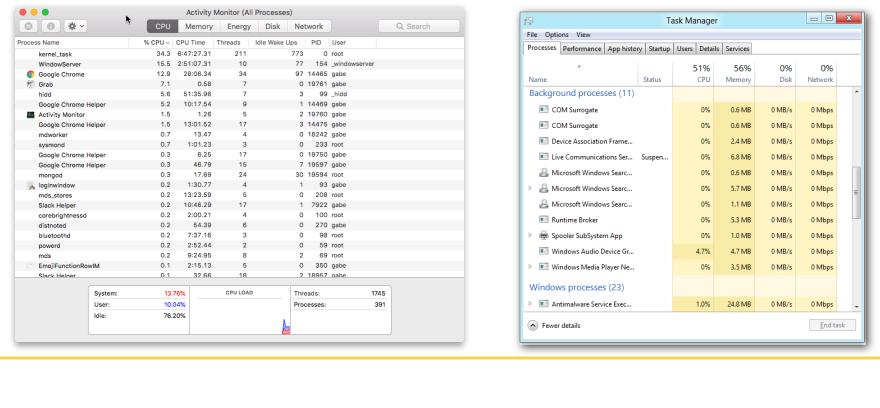


## Operating System

- ❖ It is the software program that *manages* a computer system and its operations
- ❖ It allocates and assigns system resources (e.g., manage user credentials, memory addressing)
- ❖ It schedules access to resources (e.g., CPU time, printer queue)
- ❖ It monitors system's activities (e.g., status of jobs)



## Monitor Resources



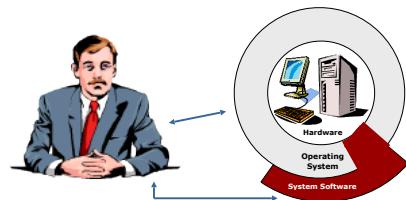
## Operating System

- ❖ It is the software program that *manages* a computer system and its operations
- ❖ It allocates and assigns system resources (e.g., manage user credentials, memory addressing)
- ❖ It schedules access to resources (e.g., CPU time, printer queue)
- ❖ It monitors system's activities (e.g., status of jobs)
- ❖ It is a platform for software applications
- ❖ It exposes Application Programming Interfaces (API)



## System Software

- ❖ Software you wouldn't need if you didn't have a computer.
- ❖ A category of software programs designed to:
  - ❖ extend the functionality of the operating system (e.g., device drivers)
  - ❖ provide specific system management functionality (e.g., antivirus)
  - ❖ provide the facilities for software development (e.g., an Integrated Development Environment).



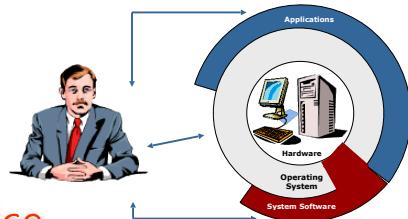
## From System Software to Operating System

- ❖ Apple: Time Machine (automatic backups)
- ❖ Apple: Compression utility
- ❖ Windows: Firewall
- ❖ Windows: Disk Cleanup utility
- ❖ Android: Google Allo
- ❖ iOS: Siri



## Application Software

- ❖ A software program that enables a user to perform an activity they would perform without a computer:
  - ❖ Excel - data analysis
  - ❖ Word - communication
  - ❖ GroupMe - verbal communication
  - ❖ Fortnite - entertainment
  - ❖ Software Suites

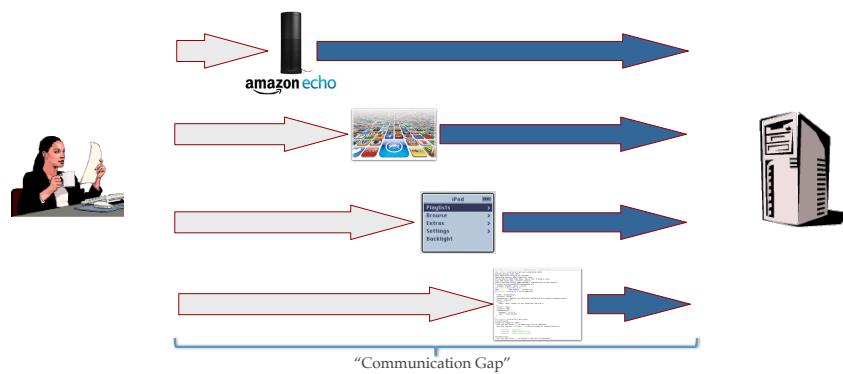


## User Interface

- ❖ Interface: The point of contact between two systems or two components of a system
- ❖ Types of UI:
  - ❖ Command line
  - ❖ Menu driven
  - ❖ Graphical (GUI)
  - ❖ Conversational Interfaces



## User Interface Evolution



## Platforms

- ❖ Technical: The “environment” in which a software program is executed:
  - ❖ Hardware architecture (e.g., Intel, ARM)
  - ❖ Operating System (e.g., Wintel, iOS)
- ❖ Application Programming Interfaces: protocols that regulate how a software application can use external resources.



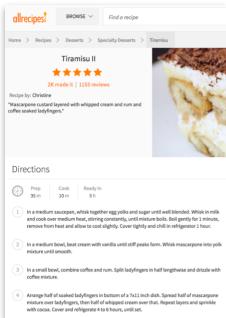


“[Creating software] is like building something where you do not have to order cement. You can create a world of your own, your own environment, and never have to leave the room.”

**Ken Thompson, Pioneer of the Unix Operating System**

## Programming a Digital Computer

- ❖ Software program: The set of instructions which govern the operations of a digital computer.
- ❖ Programming: The art and science of writing software programs
  - ❖ Algorithm design
  - ❖ Coding
- ❖ Algorithm: A set of rules for solving a problem, or achieving a result, in a finite number of steps.
- ❖ Algorithm design: The activity of envisioning and specifying the set of steps the computer must perform in order to produce the expected output (i.e., the algorithm).
- ❖ Coding: The activity of writing the algorithm as a set of precise instructions for the computer to execute - using a specific programming language.

Mohammed ibn-Musa al-Khwarizmi

## The Stored Program Concept



John von Neumann, computer engineering pioneer

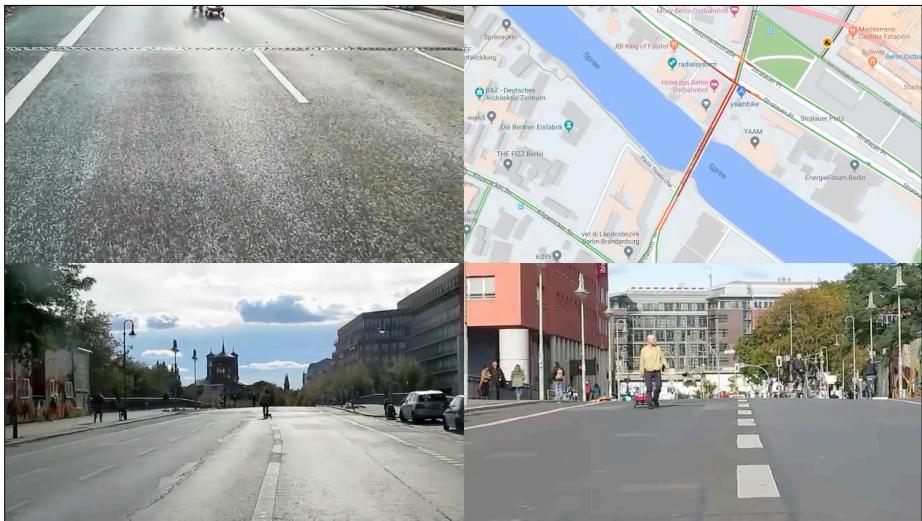
“The instructions which govern this operation must be given to the device in absolutely exhaustive detail. [...] Once these instructions are given to the device, it must be able to carry them out completely without any need for further intelligent human intervention.”

## Algorithm Design Challenge

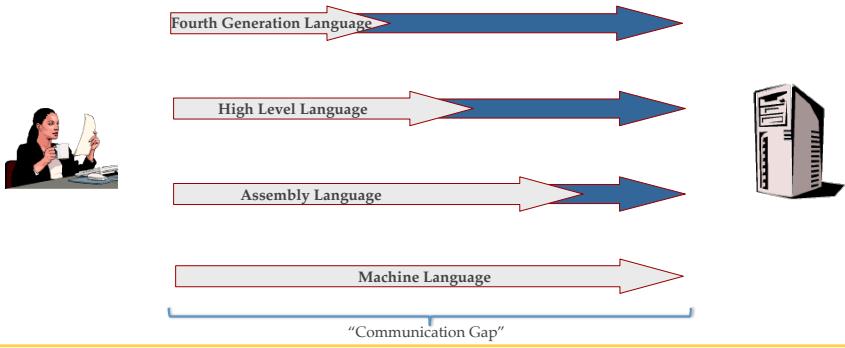


A man with dark hair, wearing a blue and white striped shirt, is pointing his index finger towards a virtual keyboard overlay. The keyboard is displayed on a white background and includes the letters Q, W, E, R, T, Y, U, I, O, P, A, S, D, F, G, H, J, K, L, Z, X, C, V, B, N, M. The keys T, H, and G are highlighted in blue. The WSJ logo is in the top right corner of the keyboard area. Below the keyboard, there is Korean text: “터치를 인식하는 영역은 커지는 거예요” (The area that recognizes touch is expanding).

A screenshot of a Twitter post by user 'Cabel Sasser'. The post includes a video thumbnail showing a horse in a game environment, with the caption "Almost a flawless heist.. problematic horse turning radius.. #PS4share". Below the video are two screenshots from the game "Assassin's Creed Origins" showing a character interacting with a horse in an Egyptian setting. The post has 2 retweets and 0 likes. The URL https://twitter.com/cabel/status/954980953964736512 is at the bottom.

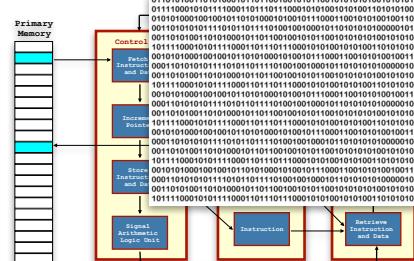


## Programming Languages Evolution



## Machine Language

- ❖ The lowest level language
  - ❖ Instructions are sequences of impulses for the CPU to process, represented as:
    - ❖ Binary representation
  - ❖ Explicit memory addressing.
  - ❖ Hardware specific: It maps directly to the **microprocessor's instruction set**.
  - ❖ Very difficult to use. The programmer has to bridge the full 'communication gap'.
  - ❖ Difficult to maintain programs over time.



## Assembly Language

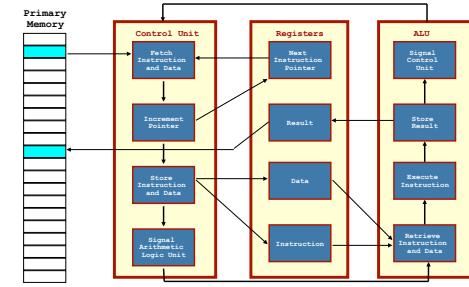
- ◊ Hardware specific low level language.
- ◊ One level of abstraction above machine language. It uses:
  - ◊ Mnemonic opcodes: Abbreviations to represent instructions (e.g., ADD, SUB)
  - ◊ Symbolic addresses: “meaningful” identifiers for memory locations (e.g., AX, BX, DI, SI).
- ◊ Requires knowledge of the hardware but easier than machine language (e.g., ADD AX BX).
- ◊ Assembler: The software program that converts assembly language code into machine language.
- ◊ Source code: Human readable software program
- ◊ Object code: Machine executable software program

Assembly Language

CU	Clear Integer	CLI	IF/H	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)
PUSH	Push onto stack	PUSH Source	DEC SP	ESP Decrease							
POP	Pop from stack	POP Dest	INC SP	ESP Increase							
PUSHA	Push all general registers	PUSHA		AX, CX, DX, BX, SP, BP, SI, DI							
PUSCI	Push all control registers	PUSCI		CS, DS, SS, ES, FS, GS							
POPA	Pop all general registers	POPA		AX, CX, DX, BX, SP, BP, SI, DI							
POPCI	Pop all control registers	POPCI		CS, DS, SS, ES, FS, GS							
IN	I/O Input	IN Dest, Port	ALU/MEM								
OUT	I/O Output	OUT Port, Source	Registers								

```
// Assume a is at address 120
// Assume F is at address 129
0  CONB 1 // a=1;
1  SAVEB 128
2  CONB 1 // F=1;
3  SAVEB 129
4  LOADA 128 // if a > 5 the jump to 17
5  CONB 5
6  COM
7  JG 17
8  LOADA 129 // f=f*a;
9  LOADA 128
10 MUL
11 SAVEC 129
12 LOADA 128 // a=a+1;
13 CONB 1
14 ADD
15 SAVEC 128
16 JUMP 4 // loop back to if
17 STOP
```

## “Assembly Language” Programming



## High Level Languages

- ◊ Also known as Third Generation Languages (3GL)
- ◊ Higher abstraction than Assembly Language.
- ◊ One to many mapping to machine language
- ◊ Platform independent
- ◊ It still must be converted into machine language in order to be executed
- ◊ Compiler: Source to object all at once
- ◊ Interpreter: Source to object one line at a time



High Level Language

```
test.py
1 import sys
2 import dlib
3 import cv2
4 from skimage import io
5
6 # Take the image file name from the command line
7 file_name = sys.argv[1]
8
9 # Create a HOG face detector using the built-in dlib class
10 face_detector = dlib.get_frontal_face_detector()
11
12 #win = dlib.image_window()
13
14 # Load the image into an array
15 image = io.imread(file_name)
16
17 # Run the HOG face detector on the image data.
18 # The detect_faces() method will return the bounding boxes of the faces in our image.
19 detected_faces = face_detector(image, 1)
20
21 print("I found {} faces in the file {}".format(len(detected_faces), file_name))
22
23 # Open a window on the desktop showing the image
24 #win.set_image(image)
25
26
27 # Loop through each face we found in the image
28 for i, face_rect in enumerate(detected_faces):
29     # Detected faces are returned as an object with the coordinates
30     # of the bounding box around them.
31     print("Face #{} found at Left: {} Top: {} Right: {} Bottom: {}".format(i, face_rect.left(), face_rect.top(),
32                                                                           face_rect.right(), face_rect.bottom()))
33
34     # Draw a box around each face we found
35     win.add_overlay(face_rect)
36
37
38 r = 1000.0 / image.shape[1]
39 dim = (1000, int(image.shape[0] * r))
40 resized = cv2.resize(image, dim, interpolation = cv2.INTER_AREA)
41
42 for i, face_rect in enumerate(detected_faces):
43     cv2.rectangle(resized, (int(face_rect.left()*r)), (int(face_rect.top()*r)), (int(face_rect.right()*r)), (int(face_rect.bottom()*r)), (0, 255, 0), 1)
44
45
46 cv2.imshow("Faces found", resized)
47 cv2.waitKey(0)
48 cv2.destroyAllWindows()
```

## Fourth Generation Languages

### SQL (4GL):

```
SORT TABLE Accounts  
ON Account_Num
```

### C (3GL):

```
Quicksort(left,right:integer;var arr:array[left..right] of integer);  
begin  
  i:=left;  
  j:=right;  
  m:=arr[trunc((left+right)/2)]; (* arr[left],arr[right],... *)  
  repeat  
    while (arr[i]<m) do i:=i+1;  
    while (arr[j]>m) do j:=j-1;  
    if (i<j) then begin  
      swap(arr[i],arr[j]);  
      i:=i+1;  
      j:=j-1;  
    end;  
  until (i>j);  
  if (left<j) then quicksort(left,j,arr);  
  if (i<right) then quicksort(i,right,arr);  
end;
```

- ❖ Higher abstraction than 3GL.
- ❖ Non-procedural English-like languages
- ❖ Outcome-driven: Programmer specifies what (the outcome) not how (the procedure)

Fourth Generation Language

### Watch Microsoft Power Apps video demos

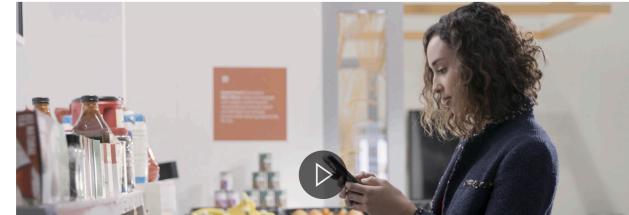
Learn how to easily build professional-grade apps from a template or scratch, add advanced AI capabilities, create custom APIs, and enhance UI with custom code components.

### Select a Power Apps demo

Empower everyone to  
build apps

Add AI features to your  
apps quickly

Extend Microsoft Power  
Platform with custom APIs



## Programming Languages Evolution

Fourth Generation Language

High Level Language

Assembly Language

Machine Language



"Communication Gap"

## Summary

- ❖ In general, higher level of abstraction imply:
- ❖ Greater ease of use
- ❖ Greater portability across hardware platforms
- ❖ Lower control by the programmer over hardware resources
- ❖ Lower efficiency of the resulting software program

Language	Portable	Concise	Mnemonic	Procedural
Machine Language	no	no	no	yes
Assembly Language	no	no	yes	yes
High Level Languages	yes	yes	yes	yes
Fourth Generation Languages	yes	yes	yes	no
Natural Language	yes	yes	yes	no

## What We Learned

- ❖ Be able to define what software is, and be familiar with the different classes of software in use today.
- ❖ Be able to define what an interface is, and be familiar with the different types of interfaces in use today.
- ❖ Be familiar with the process of software creation.
- ❖ Be able to define what an algorithm is and be familiar with the different types of programming language in use today.
- ❖ Understand the underlying logical structure of modern software applications.