جامعة التنمية البشرية
UNIVERSITY OF HUMAN DEVELOPMENT

COMPUTER PROGRAMMING CONCEPTS

CS&IT 1101
Instructor: Shakar H. Salih
E-mail: shakar.salih@uhd.edu.iq

WHO IS THE TARGET AUDIENCE?

- This course is for anyone with zero programming knowledge
- This course is for those who want to take programming as their career
- This course is for those who want to improve their programming skill with practice

OBJECTIVES

- Learn about exciting recent developments in the computer field.
- Learn computer hardware, software
- Understand the different types of programming languages.
- Learn Internet and web basics.
- Learn some key recent software technologies.
- See how to keep up-to-date with information technologies.

HOW TO PERFORM WELL

1. Revise your lecture in the same day.
2. Complete given task before next lecture.
3. Consult your teacher or your friend if you are unable to understand a concept.

THREE MOST IMPORTANT THINGS

- **Practice**
- **Practice**
- **Practice**

OUTLINES

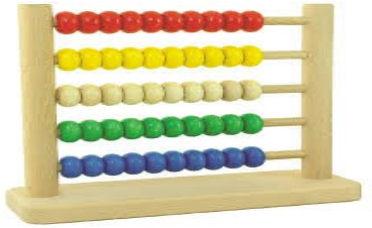
- Evolution of computers
- Generation of computers
- Anatomy of computer
- Generation of Programming Language
- Program Translators
  - Compiler
  - Interpreter

INTRODUCTION TO COMPUTERS

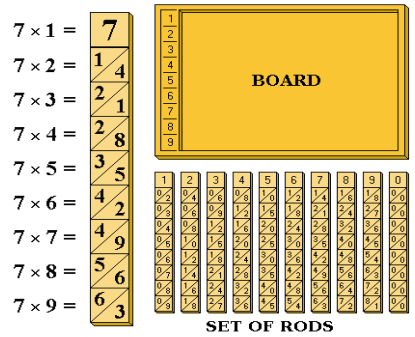
- Computer: Is a machine that stores data (numbers, words, pictures), interacts with devices (the monitor, the sound system, the printer), and executes programs.

EVOLUTION OF COMPUTERS (1/3)

1. Abacus (Counting Frame) 1250-1500 AD
   sliding beads arranged on rack

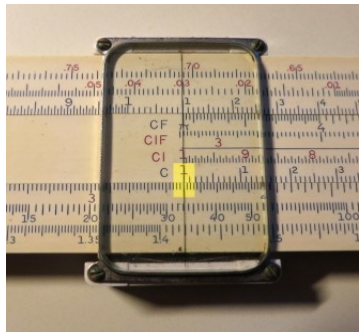


2. Napier Bones (1617)
   Uses multiplication and addition



3. Slide Rule (1632) NASA engineer

Multiplication and division , root , logarithms



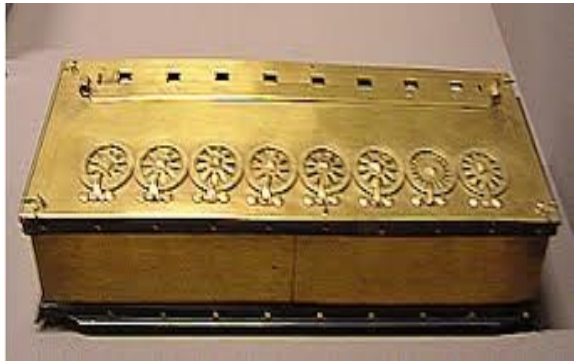
EVOLUTION OF COMPUTERS (2/3)

4. Pascaline (Arithmetic Machine) 1642

   Disadvantages:

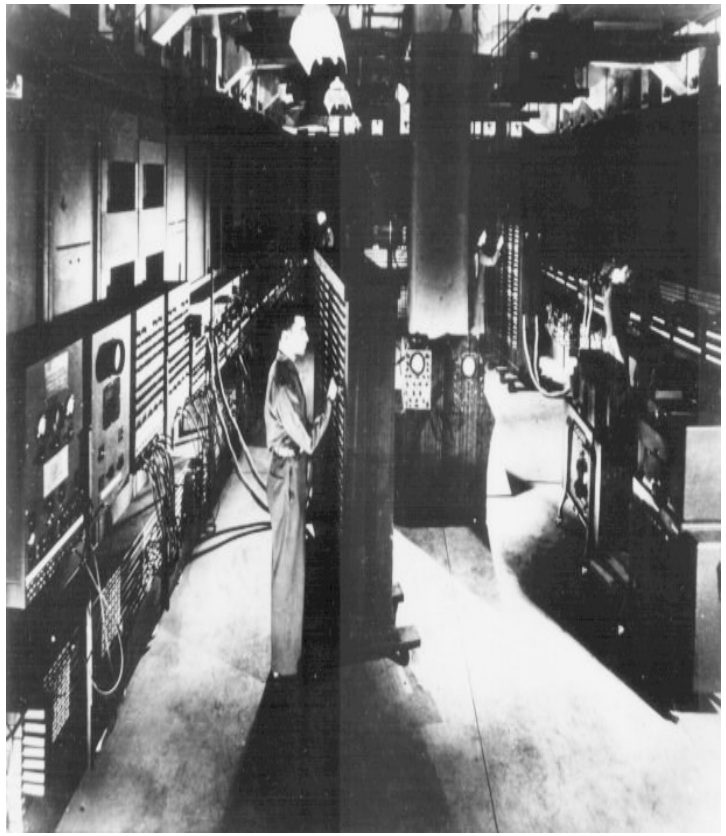
   - Costly
   - Unreliable

   .
   .
   .
   .
   .
   .



EVOLUTION OF COMPUTERS (3/3)

- ENIAC (1946) University of Pennsylvania
  - Full name
    - Electronic Numerical Integrator and Calculator
  - Inventors
    - John Mauchly and J. Presper Eckert
  - Weight 2700kg
  - 18000 Vaccum tube
  - Cost $6Lakh



GENERATION OF COMPUTERS(1/7)

- First Generation (1940 - 1956) Vacuum Tubes
  - ✓ Specification:
    1. vacuum tubes: circuitry
    2. magnetic drums: memory
    3. Input: punched cards and paper tabe
    4. Output: printouts
  - ✓ binary coded language (0s & 1s)
    - perform operations, solve one problem at a time

GENERATION OF COMPUTERS(2/7)

- ✓ Example:
  1. ENIAC
  2. EDVAC
  3. UNIVAC
- ✓ Disadvantage
  - Difficult to program
  - Expensive, huge conception of electricity and Big

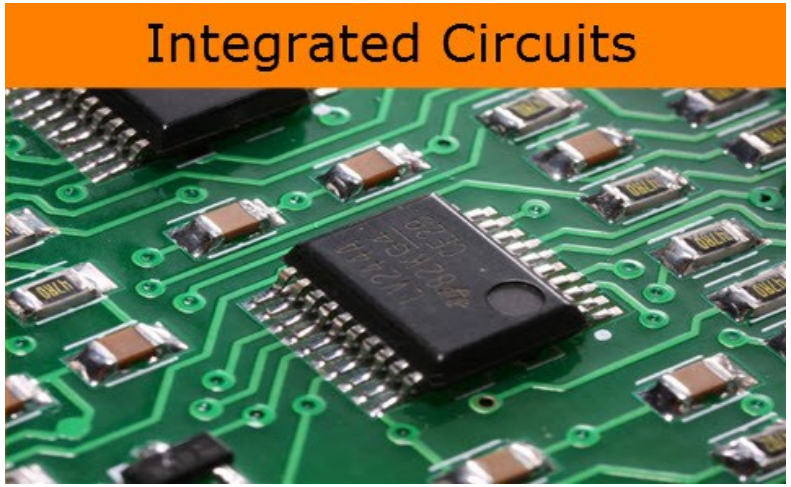
GENERATION OF COMPUTERS(3/7)

- Second Generation (1956 - 1963) Transistors
  - ✓ Transistor
    - used to relay and switch electronic signals
  - ✓ Assembly Language
  - ✓ Specification
    - Punched cards for input
    - Printout for output
    - Transistor for Circuits
    - Magnetic core technology for memory

Computers smaller, faster, cheaper, more energy -efficient

GENERATION OF COMPUTERS(4/7)

- Third Generation (1964 - Early 1970s) IC
  - ✓ Integrated circuits
    - Transistors were miniaturized and placed on silicon chips called semiconductor
  - ✓ High Level Language
  - ✓ Specification
    - Keyboard as input
    - Monitor as output
    - Operating System
      - Central program that control the device
  - ✓ Advantages
    - Speed, Efficiency



GENERATION OF COMPUTERS(5/7)

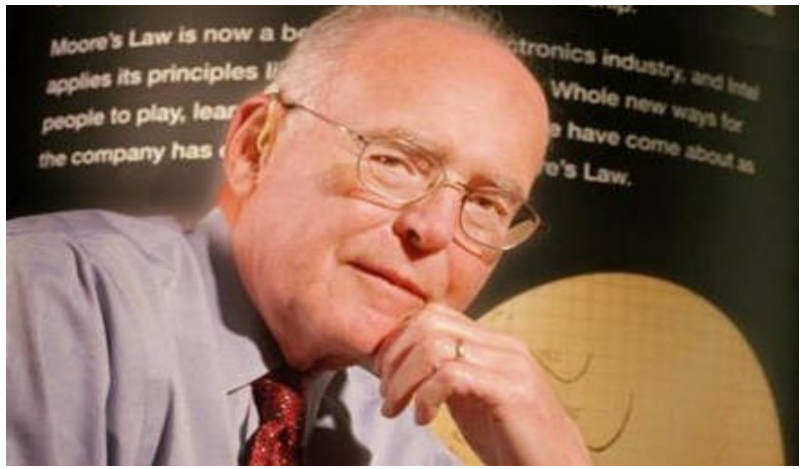
- Fourth Generation (Early 1970s – Till Date) Microprocessors
  - ✓ Data communications
  - ✓ Microprocessor
    - 1000s of ICs were built onto a single silicon chip
  - ✓ Properties
    - Instruction set, Bandwidth, Clock speed
  - ✓ Example
    - Intel 4004 chip, 1984 Apple introduced the Macintosh
  - ✓ Specification
    - Microprocessor, Mouse and other handled devices, CPU and ALU, RAID – Redundant array of Independent Disk

GENERATION OF COMPUTERS(6/7)

- Fifth Generation (Present and Beyond) Artificial intelligence
  - ✓ Artificial intelligence
    - Game playing
    - Expert System
  - ✓ Robotic
  - ✓ Voice Recognition
  - ✓ Example
    - AI computers
    - 1997, an IBM super-computer called Deep Blue defeated world chess champion Gary Kasparov in a chess

GENERATION OF COMPUTERS(7/7)

- Moore's Law
  - This law states that processor speeds, or overall processing power for computers will double every two years.
  - To break down the law even further, it specifically stated that the number of transistors on an affordable CPU would double every two years.



ELEMENTS OF A COMPUTER SYSTEM (1/6)

1. **Hardware**

- Hardware is a machine itself and its various individual equipment.
- It includes all mechanical, electronic and magnetic devices such as monitor, printer, electronic circuit, floppy and hard disk.

ELEMENTS OF A COMPUTER SYSTEM (2/6)
HARDWARE TYPE

- Computers can be envisioned as divided into various logical units or sections:
  a. **Central Processing Unit (CPU)**
     - Many of today's computers have multiple CPUs and, can perform many operations simultaneously.
     - Today's desktop computers have processors that can execute billions of instructions per second.

ELEMENTS OF A COMPUTER SYSTEM (3/6)
HARDWARE TYPE

b. **Storage**
   - Primary Storage(temporary)
   - Secondary Storage (permanently)

c. **Input Devices**

d. **Output Devices**

ELEMENTS OF A COMPUTER SYSTEM (4/6)

2. Software

- Software refers to the set of computer programs, which are used in applications and operating systems.
- Software guides the computer at every step where to start and stop during a particular job.
- The process of the software development is called *programming*

ELEMENTS OF A COMPUTER SYSTEM (5/6)
SOFTWARE TYPES

1. **Application Software:**

Application software is divided in two types :

a. **User Application**

- Is a set of programs for a specific application.
- Is useful for word processing, accounting, producing statistical report, Graphic, Excel and Data Base.

b. **System Application**

- Programming language: COBOL, FORTRAN, C++, VB, Java..

ELEMENTS OF A COMPUTER SYSTEM (6/6)
SOFTWARE TYPES

2. System Software:

- When you switch on the computer the programs written in ROM is executed which activates different units of your computer and makes it ready for you to work.
- This set of programs can be called system software.
- System software are general programs designed for performing tasks such as controlling all operations required to move data into and out of the computer.
- System software allows application packages to be run on the computer.
- DOS, UNIX and WINDOWS are some of the widely used operating system software.

GENERATION OF PROGRAMING LANGUAGE (1/15)

- Generation 1: Machine Level Programming Language (late 1940s)
- Generation 2: Assembly Language (early 1950s)
- Generation 3: High Level Language (mid 1950s to present)
- Generation 4: Specification/ Query Language, Report Generation (1970s to present)
- Generation 5: Artificial Intelligence

GENERATION OF PROGRAMING LANGUAGE (2/15)

- To build programs, people use languages that are similar to human language. The results are translated into machine code, which computers understand.
- Programming languages fall into three broad categories:

1. Machine languages
2. Assembly languages
3. Higher-level languages

GENERATION OF PROGRAMING LANGUAGE (3/15)
MACHINE LANGUAGE

- Machine languages (first-generation languages) are the most basic type of computer languages, consisting of strings of numbers the computer's hardware can use.

- Different types of hardware use different machine code. For example, IBM computers use different machine language than Apple computers.

GENERATION OF PROGRAMING LANGUAGE (4/15)
MACHINE LANGUAGE

- Native language of computers.
- Based on binary language; every instruction and data should be written using 0's and 1's.
- Instruction in m/c language consists of two parts:

OPCODE (Operation Code)	OPERAND (Memory Location)

  - Opcode tells the computer what functions are to be performed.
  - Operand tells the computer where to find or store the data on which the desired function is to be performed.

GENERATION OF PROGRAMING LANGUAGE (5/15)

Giving Instructions to Microprocessor

- It's very hard for human beings to remember different patterns of 0's and 1's
- It is almost impossible for a human being to write a computer program consisting only of 0's and 1's
- There was a need to define a language which is easier to understand and work with

GENERATION OF PROGRAMING LANGUAGE (6/15)
ASSEMBLY LANGUAGE

Solution: Assembly Language

- Assembly languages (second-generation languages) are only somewhat easier to work with than machine languages.

- To create programs in assembly language, developers use cryptic English-like phrases to represent strings of numbers.

  Example: ADD → 00101101

GENERATION OF PROGRAMING LANGUAGE (7/15)
ASSEMBLY LANGUAGE

Remember: Microprocessor can only understand machine language (0s & 1s)

- How assembly language will work then?
- Solution: Assembler
- The code is then translated into object code, using a translator called an assembler.
- An assembler converts the assembly code into binary code.
- Used abbreviations for the instructions.

```
;CLEAR SCREEN USING BIOS
CLR: MOV AX,0600H        ;SCROLL SCREEN
     MOV BH,30           ;COLOUR
     MOV CX,0000         ;FROM
     MOV DX,184FH        ;TO 24,79
     INT 10H             ;CALL BIOS;
;INPUTTING OF A STRING
KEY: MOV AH,0AH          ;INPUT REQUEST
     LEA DX,BUFFER       ;POINT TO BUFFER WHERE STRING STORED
     INT 21H             ;CALL DOS
     RET                 ;RETURN FROM SUBROUTINE TO MAIN PROGRAM;
; DISPLAY STRING TO SCREEN
SCR: MOV AH,09           ;DISPLAY REQUEST
     LEA DX,STRING       ;POINT TO STRING
     INT 21H             ;CALL DOS
     RET                 ;RETURN FROM THIS SUBROUTINE;
```

Assembly code

Assembler

```
0001010010110101010101010101010100010
1110110101010101010101110010100010110
0010100101010010111101011101011101010
1001010010110101010101010101010110110
0110100100110010111101011101010100010
0001000101011101010101000101010111010
1010100101010010101101011101011101011
0001010010110101010101010101010100010
```

Object code

GENERATION OF PROGRAMING LANGUAGE (9/15)
ASSEMBLY LANGUAGE

Problem with Assembly Language

- It is more close to machine rather than human beings
- Human mind does not work so simply as the machine does
- It is still very different from the languages that human beings speak
- It is still difficult for human beings to solve a complex problem by writing a computer program in assembly language

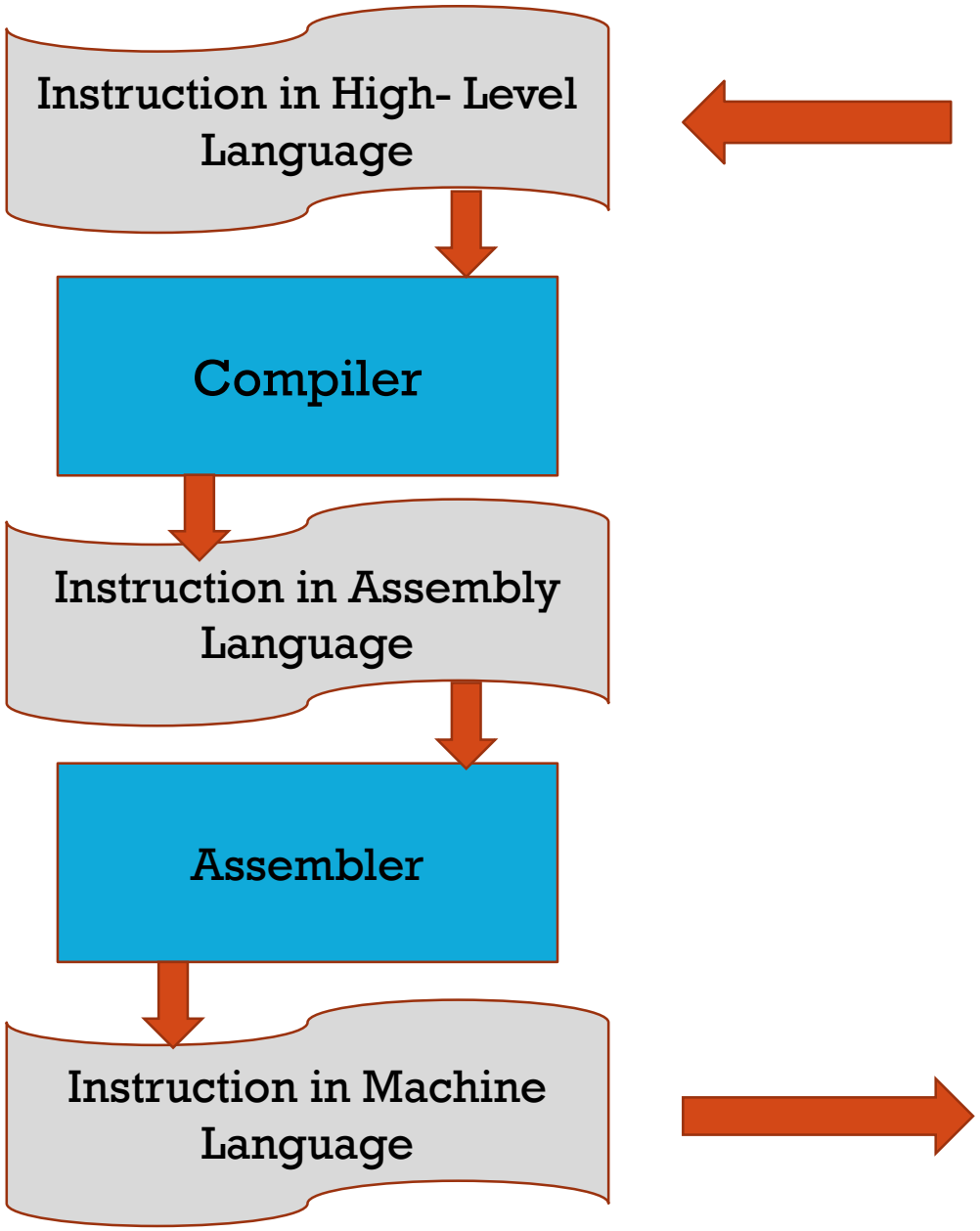
GENERATION OF PROGRAMING LANGUAGE (10/15)
HIGHER-LEVEL LANGUAGES

- Solution: High Level Languages
- The programming languages which are closer to human languages and further from machine languages
- In contrast, assembly languages are considered low-level because they are very close to machine languages

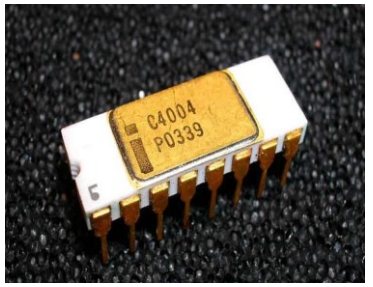
GENERATION OF PROGRAMING LANGUAGE (11/15)
HIGHER-LEVEL LANGUAGES

Remember: Microprocessor can only understand machine language (0s & 1s)

- How a high-level language will work then?
- Solution: Compiler
- Compiler : a translator to convert instructions written in a high-level language to low-level language



Human being



Microprocessor

GENERATION OF PROGRAMING LANGUAGE (12/15)
HIGHER-LEVEL LANGUAGES

Higher-level programming languages are divided into three "generations," each more powerful than the last:

1. Third-generation languages
2. Fourth-generation languages
3. Fifth-generation languages

GENERATION OF PROGRAMING LANGUAGE (13/15)
THIRD-GENERATION LANGUAGES

- Third-generation languages (3GLs) are the first to use true English-like phrasing, making them easier to use than previous languages.

- 3GLs are portable, meaning the object code created for one type of system can be translated for use on a different type of system.

- The following languages are 3GLs:

FORTAN	C
COBOL	C++
BASIC	Java
Pascal	ActiveX

GENERATION OF PROGRAMING LANGUAGE (14/15)
FOURTH-GENERATION LANGUAGES

- Fourth-generation languages (4GLs) are even easier to use than 3GLs.
- 4GLs may use a text-based environment (like a 3GL) or may allow the programmer to work in a visual environment, using graphical tools.
- The following languages are 4GLs:

  Visual Basic (VB)

GENERATION OF PROGRAMING LANGUAGE (15/15)
FIFTH-GENERATION LANGUAGES

- Fifth-generation languages (5GLs) are an issue of debate in the programming community – some programmers cannot agree that they even exist.
- These high-level languages would use artificial intelligence to create software, making 5GLs extremely difficult to develop.
- Solve problems using constraints rather than algorithms, used in Artificial Intelligence
  - Prolog
  - LISP

PROGRAM TRANSLATORS (1/3)

- ✓ Assembler
- ✓ Compiler
- ✓ Interpreter

PROGRAM TRANSLATORS (2/3)
COMPILER

- A compiler is a program translator that translates a program written in high-level language into machine language program.
- During the translation process, the compiler reads the source program and checks for syntax errors. In case of any errors, the compiler will not create the object code until all the errors are rectified.
- Once the program has been compiled, the resulting m/c code is saved separately, and can be run on its own at any time. But, if the source code is modified then it is necessary to recompile the program again to reflect the changes.

PROGRAM TRANSLATORS (3/3)
INTERPRETER

- Also a language translator.
- Translates high-level language into machine language.
- Unlike compilers, it translates a statement in a program and execute immediately, i.e., before translating the next source language statement.
- When an error is encountered, the execution of the program is halted and an error message is displayed.
- Languages such as BASIC and LISP has its own interpreters.

DIFFERENCE BETWEEN COMPILER AND INTERPRETER

Basis	Compiler	Interpreter
Object Code	A compiler provides a separate object program.	An interpreter does not generate a permanent object code file.
Translation Process	Converts the entire program into machine code at one go.	Translates the source code line-wise.
Debugging Ease	Removal of errors (debugging) is slow.	Debugging becomes easier because the errors are pointed out immediately.
Implementation	Compilers are complex programs. They require hard-core coding. Also require more memory to execute a program.	Interpreters are easier to write because they are less complex programs. Also require less memory for program execution.
Execution Time	Compilers are faster as compared to interpreters because each statement is translated only once and saved in object file, which can be executed anytime without translating again.	Interpreters are slower as compared to compilers because each statement is translated every time it is executed from the source program.

REVIEW QUESTIONS

1. What are the five basic operations performed by any computer system?
2. What was the first computer programing language.
3. The first Generation Computer used _____ for circuitry and _____ for memory
4. _____ Language were used in First Generation Computers

THANK YOU.....

DO YOU HAVE ANY QUESTIONS ?