Programming Fundamentals

Module A - Introduction

Le The Anh
anhlt161@fe.edu.vn

FPT UNIVERSITY

Objectives

- Define some concepts related to programming
- Explain how to make a good software
- Understand steps to develop a software
- Explain ways for representing data
- Answer why C is the first language selected
- Understand how a C program can be translated and execute
- Discuss about notable features of the C language
- Understand a C program structure

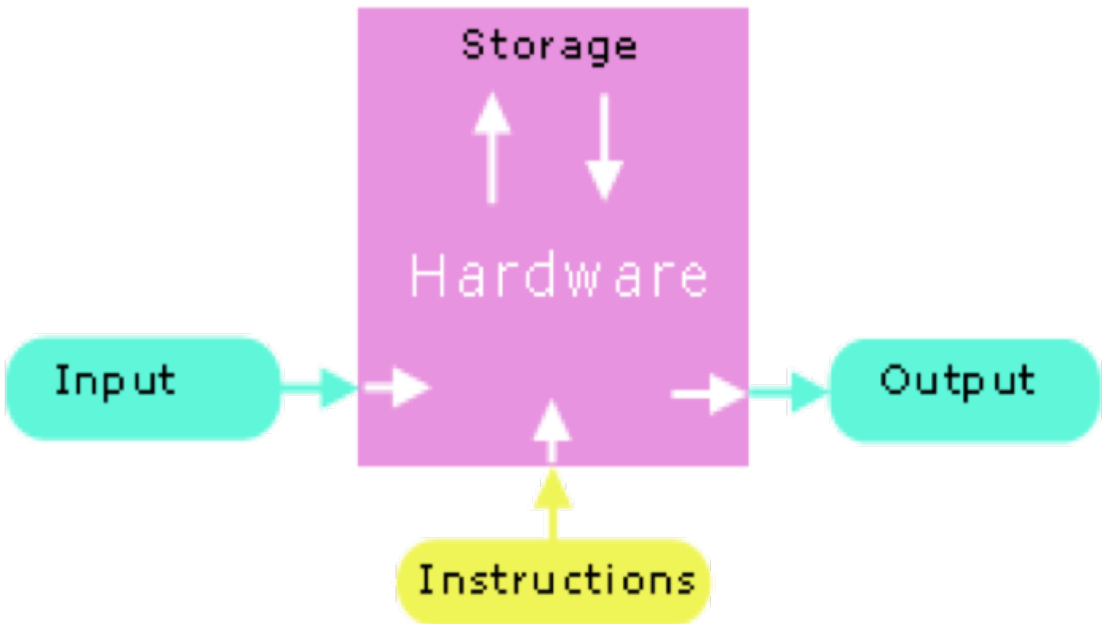
Contents

Basic concepts

- Information: Knowledge about something
- Data: Values are used to describe information. So, information can be called as the mean of data.
- Problem: A situation in which something is hidden
- Solve a problem: explore the hidden information
- Algorithm: A finite sequence of well-defined instructions, typically used to solve a class of specific problems or to perform a computation
- Program: A sequence of steps to find out the solution of a problem. A program is an implementation of an algorithm.

Basic concepts

- Computer program: A program executed using a computer; a set of instructions that computer hardware will execute.
- The instructions transform raw data (the input) into information and eventually into comprehensible results (the output).
- The hardware stores the data internally during this process.



- Computer software: A set of related programs.

How to make a good software?

Issues for a program/software:

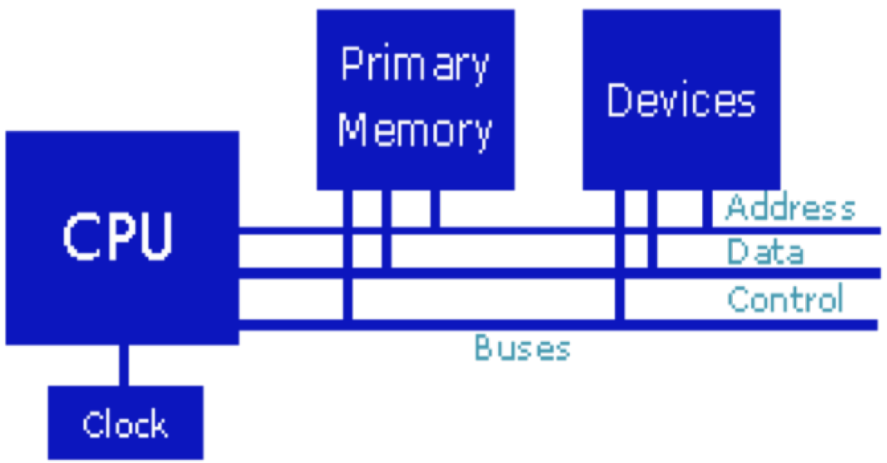
- Usability: Users can use the program to solve the problem
  - Robust and user-friendly interfaces
- Correctness: Solution must be correct
  - Comprehensive testing
- Maintainability: The program can be modified easily
  - Understandability
    - Structured programming
    - Internal documentation
- Modifiability
  - Standards compliance
- Portability: The program can run in different platforms
  - Standards compliance $\rightarrow$ Needed modifications are minimum (platform: CPU + operating system running on it)

Steps to develop a software

1. Requirements: Understand the problem
2. Analysis: Identify data and tasks
3. Design: Organize folder, files, ..
4. Coding: Implementation
5. Testing: Check whether requirements are satisfied or not
6. Deploying: Install the program to user computers
7. Maintenance: Make needed modifications

Computer hardware (read yourself)

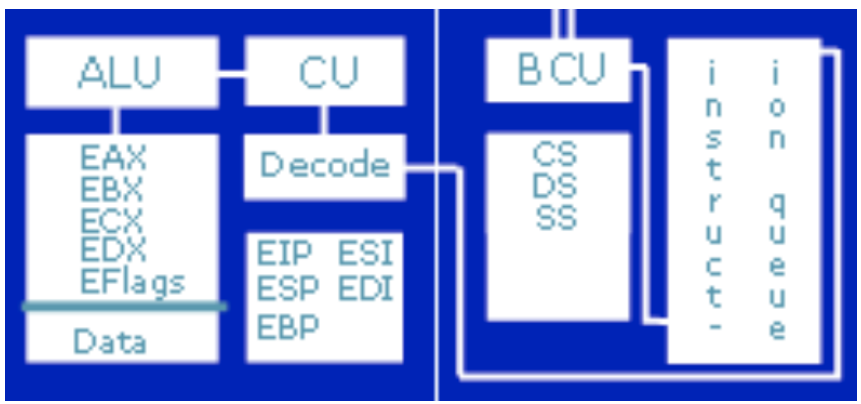
- Main hardware components: CPU, primary memory, a set of devices
- These components are interconnected by buses (an address bus, a data bus, and a control bus).
- The buses carry information between the hardware components.



Computer hardware

Center Processing Unit

- CPU = {EU, BIU}. EU executes the instructions one at a time. BIU manages the transfer of information along the data bus to and from EU.
- BIU = {Bus Control Unit (BCU), Segment Address Registers (CS, DS, SS, ..), Instruction Queue} fetches instructions and places them in the instruction queue.
- EU = {Control Unit (CU), Decode Unit, Arithmetic & Logic Unit (ALU), General Registers (EAX, EBX, ..), Address Registers (EIP, ESP, ..), Data Cache}
- CPU memory is volatile - the contents of the registers are lost as soon as power is turned off.



- Primary memory holds the information accessed by the CPU.
- Primary memory is also volatile.
- The popular term for primary memory is RAM (Random Access Memory).
- Primary memory holds the program instructions and the program data.

Computer hardware

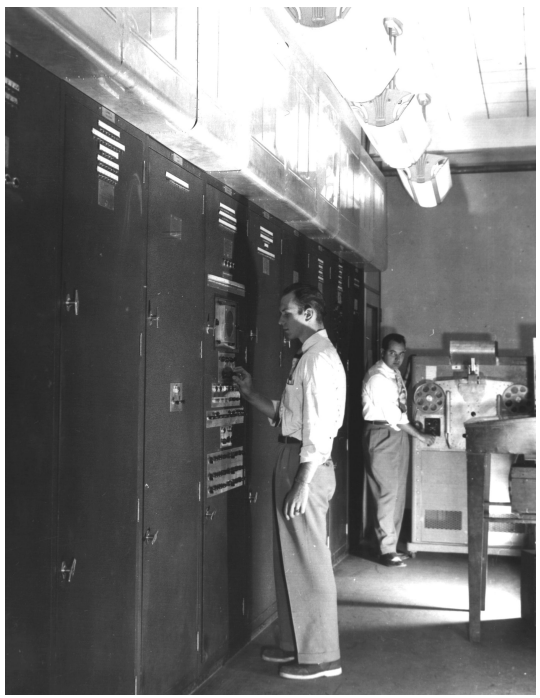
Devices

- I/O devices such as a keyboard, a monitor and a mouse ...
- Storage devices such as a floppy drive, a hard drive and a CD-ROM drive (secondary storage).
- Each device interfaces with the system buses through a device controller.
- The most expensive and fastest memory - registers - is reserved for the CPU.
- CPU transfers information at less than 10 nanoseconds[1]
- Primary memory transfers information at about 60 nanoseconds
- A hard disk transfers information at about 12,000,000 nanoseconds

[1] A nanosecond is 10^{-9} seconds.

Data units

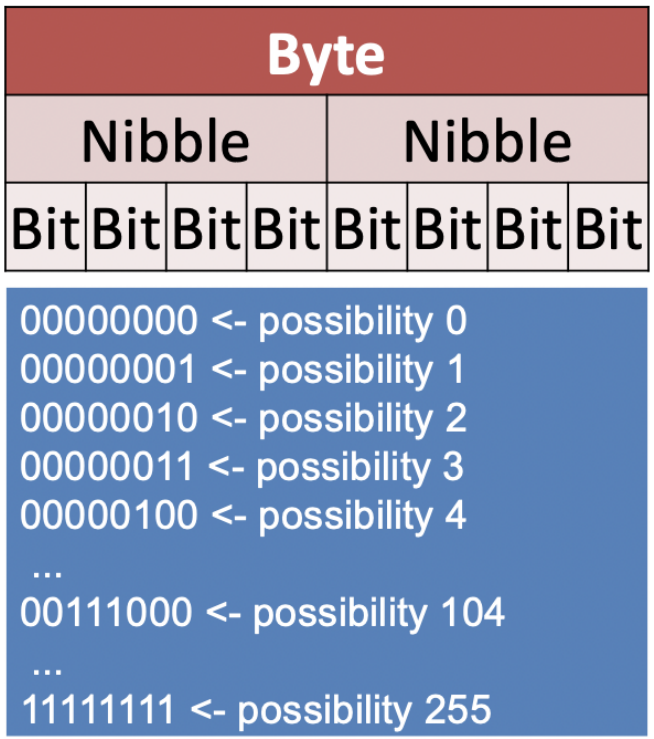
- John von Neumann selected binary (base 2) digits as the EDVAC's fundamental unit.
- The vast majority of modern computers process and store information in binary digits.
- We call a bi nary digi t a bit .



The EDVAC as installed in Building 328 at the Ballistic Research Laboratory (borrowed from Wikipedia)

Data units

- Nibble = 4 consecutive bits
- Byte = 2 nibbles = 8 bits
- Unit of memory is Byte



Data units

Word - CPU unit

- The natural unit of the CPU is a word.
- A word is the size of the general registers - the unit of memory within the CPU.
- Word size varies from manufacturer to manufacturer. On 16-bit machines, a word is 2 bytes. On 32-bit machines, a word is 4 bytes.
- Word length can be 8, 16 (old CPUs), 32, 64 (current CPUs)

Data representations

- Binary: 0, 1 (e.g, 101111, 101111b)
- Octal: 0, 1, .., 6, 7 (e.g, 057, 57q[2])
- Decimal: 0, 1, .., 8, 9 (e.g, 47, 47d)
- Hexadecimal: 0, 1, .., 9, A, B, C, D, E, F (e.g, 0x2F, 2Fh)

0 0 0 0	0	1 0 1 0	A (10)
0 0 0 1	1	1 0 1 1	B (11)
0 0 1 0	2	1 1 0 0	C (12)
0 0 1 1	3	1 1 0 1	D (13)
0 1 0 0	4	1 1 1 0	E (14)
0 1 0 1	5	1 1 1 1	F (15)
0 1 1 0	6		
0 1 1 1	7		
1 0 0 0	8		
1 0 0 1	9		

[2] use q instead of o being mistaken for a zero

Data representations

Dec to Bin, Oct, Hex

- Dec to Bin: $26 = ?$

$$\begin{array}{ccccccccccc} 26 & \xrightarrow{/2} & 13 & \xrightarrow{/2} & 6 & \xrightarrow{/2} & 3 & \xrightarrow{/2} & 1 & \xrightarrow{/2} & 0 \\ & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ 11010 & \leftarrow & 0 & \leftarrow & 1 & \leftarrow & 0 & \leftarrow & 1 & \leftarrow & 1 \end{array}$$

- Dec to Oct: $26 = ?$

$$\begin{array}{ccccc} 26 & \xrightarrow{/8} & 3 & \xrightarrow{/8} & 0 \\ & & \downarrow & & \downarrow \\ 32 & \leftarrow & 2 & \leftarrow & 3 \end{array}$$

- Dec to Hex: $26 = ?$

$$\begin{array}{ccccc} 26 & \xrightarrow{/16} & 1 & \xrightarrow{/16} & 0 \\ & & \downarrow & & \downarrow \\ 1A & \leftarrow & 10 & \leftarrow & 1 \end{array}$$

Data representations

Bin to Dec, Oct, Hex

- Bin to Dec: $101111 \rightarrow 2^5 + 2^3 + 2^2 + 2^1 + 2^0 = 47$
- Bin to Oct: $101111 \rightarrow 101|111 \rightarrow 57$
- Bin to Hex: $101111 \rightarrow 10|1111 \rightarrow 2F$

Data representations

Oct to Bin, Dec, Hex

- Oct to Bin: $57 \rightarrow 5|7 \rightarrow 101|111 \rightarrow 101111$
- Oct to Dec: $57 \rightarrow 5 * 8^1 + 7 * 8^0 = 47$
- Oct to Hex:
  - Oct $\rightarrow$ Bin $\rightarrow$ Hex: $57 \rightarrow\rightarrow 101111 \rightarrow 2F$
  - Oct $\rightarrow$ Dec $\rightarrow$ Hex: $57 \rightarrow 47 \rightarrow 2F$

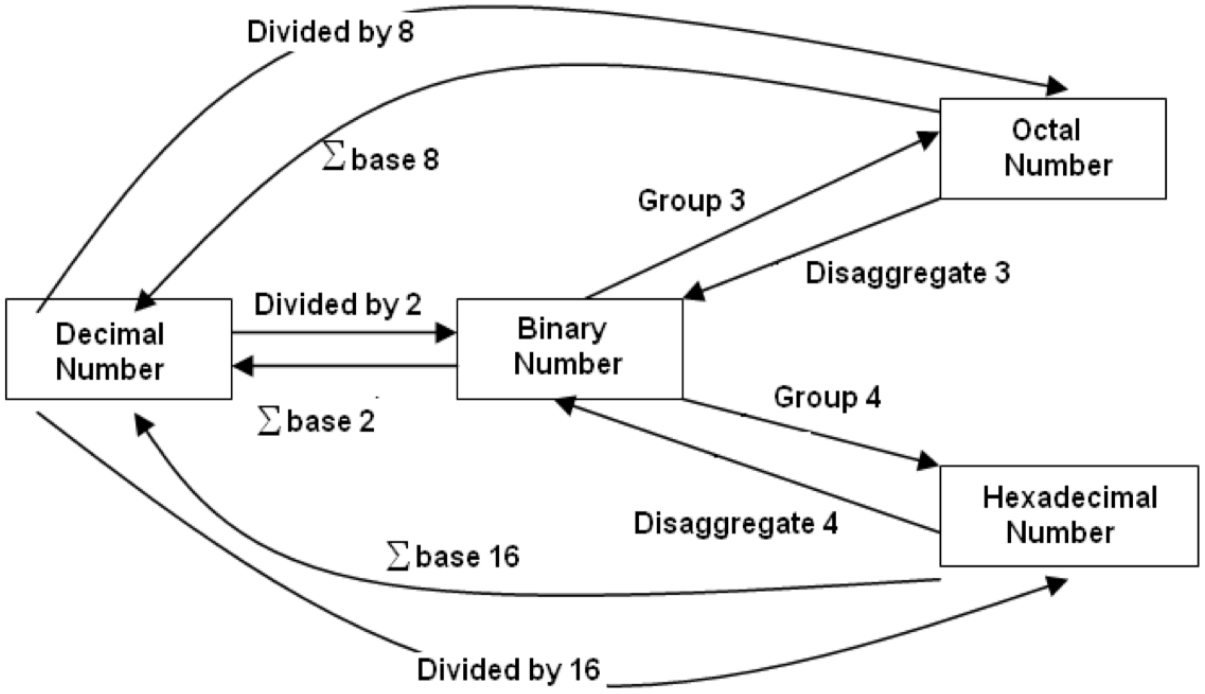
Data representations

Hex to Bin, Dec, Oct

- Hex to Bin: $2F \rightarrow 2|F \rightarrow 10|1111 \rightarrow 101111$
- Hex to Dec: $2F \rightarrow 2 * 16^1 + F * 16^0 = 47$
- Hex to Oct:
  - Hex $\rightarrow$ Bin $\rightarrow$ Oct: $2F \rightarrow\rightarrow 101111 \rightarrow 57$
  - Hex $\rightarrow$ Dec $\rightarrow$ Oct: $2F \rightarrow 47 \rightarrow 57$

Data representations

Conversion Tips



Bin	Oct	Dec	Hex
110101	?	?	?
?	62	?	?
?	?	42	?
?	?	?	C1

Data representations

Conversion exercises

Fill the corresponding binary expansions of the following decimal number:

Decimal	4-bit Binary	Decimal	8-bit Binary	Decimal	16-bit Binary
9	1001	7	0000 0111	255	0000 0000 1111 1111
7	0111	34	0010 0010	192	0000 0000 1100 0000
2	0010	125	0111 1101	188	0000 0000 1011 1100
15	1111	157	1001 1101	312	0000 0001 0011 1000
12	1100	162	1010 0010	517	0000 0010 0000 0101
11	1011	37	0010 0101	264	0000 0001 0000 1000
6	0110	66	0100 0010	543	0000 0010 0001 1111
5	0101	77	0100 1101	819	
8	1000	88	0101 1000	1027	
13	1101	99	0110 0011	2055	
14	1110	109	0110 1101	63	

Data representations

Conversion exercises

Fill the blank cells:

Decimal	Binary	Hexa.	Decimal	16-bit Binary	Hexa.
9	1001	9	255	0000 0000 1111 1111	00FF
127	0111 1111	9F	192		
125			188		
157			312		
162			517		
37			264		
66			543		
77			819		
88			1027		
99			2055		
109			63		

Data representations

Binary operations: addition and subtraction

- Binary addition:

	Bin				Dec
	1	0	0	1	9
+	1	0	1	1	11
1	0	1	0	0	20

- Binary subtraction:

	Bin				Dec
1	0	1	0	0	20
-	1	0	1	1	11
	1	0	0	1	9

- Binary multiplication (association of addings and shifts): 1001×110

$$\begin{array}{cccccc} & & 0 & 0 & 0 & 0 \\ + & 1 & 0 & 0 & 1 & \\ \hline 1 & 0 & 0 & 1 & & \\ \hline 1 & 1 & 0 & 1 & 1 & 0 \end{array}$$

- Binary division (association of subtractions): $101111 : 1001$

$$\begin{array}{cccccccc|cccc} & 1 & 0 & 1 & 1 & 1 & 1 & & 1 & 0 & 0 & 1 \\ \hline & 1 & 0 & 1 & 1 & & & & 1 & & & \\ - & 1 & 0 & 0 & 1 & & & & & & & \\ \hline & 0 & 0 & 1 & 0 & & & & & & & \\ & & & 1 & 0 & 1 & & & & 0 & & \\ & & & 1 & 0 & 1 & 1 & & & & 1 & \\ & & - & 1 & 0 & 0 & 1 & & & & & \\ \hline & & & 0 & 0 & 1 & 0 & & & & & \end{array}$$

10: remainder 101: quotient

Data representations

Bit operators: and, or, xor, not

Operator	Example	In-class practice
a b a and b	1 0 0 1 0 1	1 0 1 0 0 1 1
0 0 0	and	and
0 1 0	0 0 1 1 0 1	1 0 1 0
1 0 0	----------	----------
1 1 1	0 0 0 1 0 1	???
a b a or b	1 0 0 1 0 1	1 0 1 1 0 0 1
0 0 0	or	or
0 1 1	1 1 0 1	1 0 1 0 0
1 0 1	----------	----------
1 1 1	1 0 1 1 0 1	???
a b a xor b	1 0 0 1 0 1	1 0 1 1 0 1 1
0 0 0	xor	xor
0 1 1	1 1 0 1	1 0 1 1 0 0
1 0 1	----------	----------
1 1 0	1 0 1 0 0 0	???
a not a	(not) 1 0 0 1 0 1	(not) 1 0 1 1 1 0 0
0 1	----------	----------
1 0	0 1 1 0 1 0	???

Data representations

Exercises

- 101101b + 101b
- 101101b - 111b
- 101101b * 1011b
- 1001010b : 101b
- Oct, Hex Operations (Homework)
  - 3245q + 247q = ?
  - 134q +, -, *, : 67q = ?
  - 13Ah +, -, *, : 6Eh = ?

Data representations

Signed Integers

- Use the leftmost bit to indicate the sign (0: positive, 1: negative)
  - 67d = 01000011b
  - -67d = 11000011b
  - Check: 67d + (-67d) = 0 ?

```
  0 1 0 0 0 0 1 1
+ 1 1 0 0 0 0 1 1
-----------------
1 0 0 0 0 0 1 1 0 (not equal to 0d)
```

- Solution: use 2-complement format
  - 67d = 01000011b (suppose 8 bit are used to represent a signed number)
  - Represented -67d in the 2-complement format

```
  0 1 0 0 0 0 1 1 (+67d)
  1 0 1 1 1 1 0 0 (1-complement/ reverse bit/ not operator)
+               1
  ---------------
  1 0 1 1 1 1 0 1 (-67d: 2-complement format)
```

  - Check 67d + (-67d) = 0d ?

```
  0 1 0 0 0 0 1 1 (+67d)
+ 1 0 1 1 1 1 0 1 (-67d)
-----------------
1 0 0 0 0 0 0 0 0 (use 8 bits, the leftmost bit is removed)
  0 0 0 0 0 0 0 0 (0d)
```

Data representations

Exercises

1. Represent the following numbers in the 1-byte unsigned binary format: 251, 163, 117
2. Show the binary format of 2-byte unsigned numbers: 551, 160, 443
3. Show binary formats of 1-byte signed numbers: -51, -163, -117, 320
4. Show the decimal values of 1-byte unsigned representations: 01100011b, 10001111b, 11001010b, 01001100b
5. Show the decimal values of 1-byte signed representations: 01100011b, 10001111b, 11001010b, 01001100b

Addressing Information

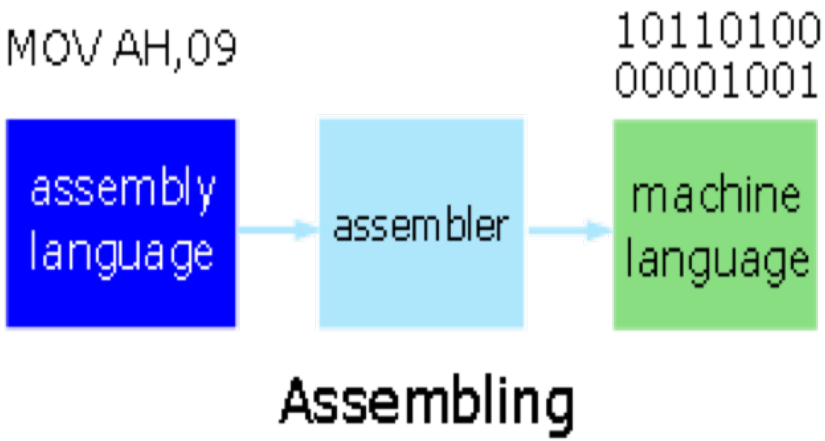
- To access data in primary memory, program instructions refer to the addresses where that data is stored.
- Each byte of primary memory has a unique address. Addressing starts at zero, is sequential and ends at the size of primary memory less 1.
  - Exabyte (E), Petabyte (P), Terabyte (T), Gigabyte (G), Megabyte (M), Kilobyte (K), Byte (B), Bit (b)
  - 1E $= 2^{10} = 1024$P, 1P = 1024T, .., 1k = 1024 bytes, 1 byte = 8 bits.

Size:	1 Byte		1 Byte		1 Byte		...	1 Byte	
Hex:	1 Nibble	1 Nibble	1 Nibble	1 Nibble	1 Nibble	1 Nibble	...	1 Nibble	1 Nibble
Contents:							...		
Binary:	00000000_2		00000001_2		00000010_2		...	$1...111111111_2$	
Hex:	0x0000000		0x0000001		0x0000002		...	0xFFFFFFF	

- The maximum size of addressable primary memory depends upon the size of the address registers (Pentium machines where the address registers hold 32 bits the maximum size of addressable memory is about 4GB, ranging from 0 to $2^{32} - 1$)

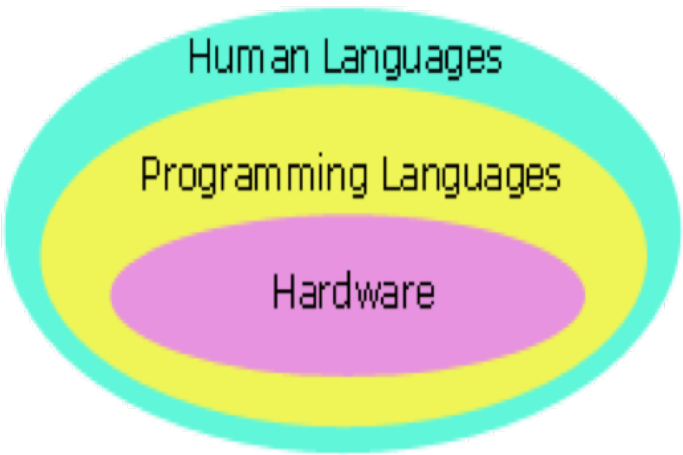
Program Instructions

- Each program instruction consists of an operation and operands.
- The CPU performs the operation on the values stored as operands or on the values stored in the operand addresses.
- The addresses are either register names or primary memory addresses.



Programming Languages

- Programs that perform relatively simple tasks and are written in assembly language contain a large number of statements.
- The higher the level, the closer to the human languages and the further from native machine languages

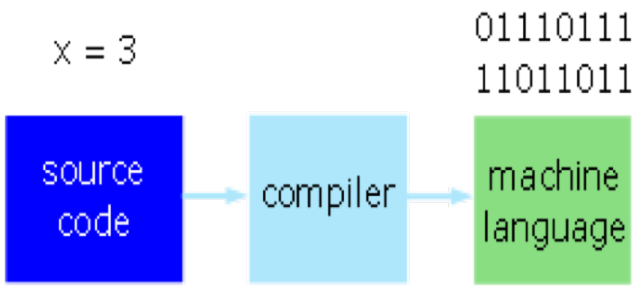


Programming Languages

- Machine languages (1940s): Native machine language. Each manufacturer has its own unique set of instructions.
- Assembly languages (late 1940s): These are readable versions of the native machine languages. The manufacturer supplies an assembly language for its machines.
- Third-generation languages. These are languages with instructions that describe how a result is to be obtained (Eg. FORTRAN 1950s; Pascal, C - 1970s)
- Fourth-generation languages. These are languages with instructions that describe what is to be done without specifying how it is to be done (Eg. SQL, Matlab)
- Fifth-generation languages are the closest to human languages. They are used for artificial intelligence, fuzzy sets, and neural networks (Eg. Mercury, ICAD).
- Each third generation language statement $\approx$ 5-10 machine language statements.
- Each fourth generation language $\approx$ 30-40 machine language statements.

Translating and executing a program

- When we code a program in a high level language, we write source code that can't be run by computers. The source code must be translated to binary code (machine code). This is done by using either an interpreter, or a compiler.
- Interpreters: Translate and execute one-by-one statement.
- Compilers: Translate the entire set of high level statements into an equivalent set of machine language statements (without executing any of the statements) and produce a separate executable file. We execute that file separately afterwards. Compilers can produce optimized results.



Compiling

Why C as a First Language

- C is one of the most popular languages in use globally (TIOBE index)
- Some reasons for learning programming using the C language include:
  - C is English-like,
  - C is quite compact - has a small number of keywords,
  - A large number of C programs need to be maintained,
  - C is the lowest in level of the high-level languages,
  - C is faster and more powerful than other high-level languages, and
  - The UNIX, Linux and Windows are written in C and C++.
- Comparative times for Sieve of Eratosthenes test

Language	Time to Run
Assembly	0.18 seconds
C	2.7 seconds
Basic	10 seconds

Some Notable Features

- Comments (/* */, //): We use comments to document our programs and to enhance their readability. C compilers ignore all comments.
- Whitespace: We use whitespace to improve program readability and to display the structure of our program's logic. C compilers ignore all whitespace.
- Case Sensitivity: C language is case sensitive ('A' and 'a' are totally different).

Compiling C Programs

```c
/*
My first program
hello.c
Le The Anh
Aug 14 2021
*/

#include <stdio.h>

int main()
{
    printf("Hello World! I'm a C program.");
    return 0;
}
```

Summary

Exercises

!!! Exercises:

- Complete Workshop 1,
- Read pages 1-10 from Evan Weaver's subject notes, and
- Continue checking out the given offline web site.

- Assignment explanation

Q&A