Fundamentals of Computer Programming

[Lecture 1]

In natural science, Nature has given us a world and we're just to discover its laws.
In computers, we can stuff laws into it and create a world. [Alan Kay]

Saeed Reza Kheradpisheh

s_kheradpisheh@sbu.ac.ir

Department of Computer Science
Shahid Beheshti University

Fall 1398

About the Course

- Give you a better understanding of how computer applications work.
- Teach you how to write your own applications (in C++).
- You'll learn computational thinking!
- You'll learn how to solve problems using computers.

The Journey Ahead

- Course contents
  - Theoretical and Conceptual
  - **Practical**
- Types of sessions
  - Regular
  - Workshop
- Pay attention
  - You are not allowed to use any electronic devices during lectures.
  - Laptops are only allowed in workshop sessions.
  - Feel free to ask your questions during lectures.
  - Watch out deadlines (Exercises/Project).
  - Share your thoughts, not your code.
- **The Golden Key**: Practice, Practice, and Practice.

Grading

Exercises + Quizzes	$15+5$ points
Mid-term Exam (Paper + Computer)	$10+15$ points
Final Exam (Paper + Computer)	$10+15$ points
Final Project	30 points
Extra Points	up to 10 points

References

Textbook:

- Cay Horstmann, *Big C++: Late Objects*.

Good Resources:

- Deitel & Deitel, *C++ How to Program*.
- Cay Horstmann, *C++ for everyone*.

Introduction to Computers

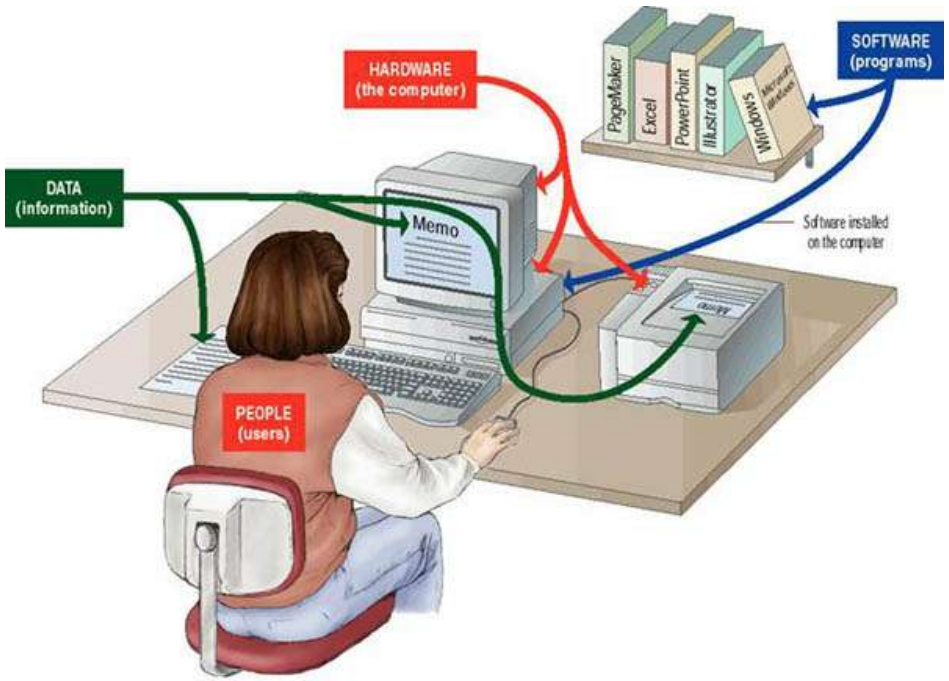
Definition

A **computer** is an electronic device that takes input such as raw data which can be numbers, text, sound, image, animations, video, etc., processes it, and converts it into meaningful information that could be understood, presenting the changed input (processed input) as output.

- The **data** consists of numbers, text, sound, images, animations, and video.
- The **process** converts numbers, text, sound, images, animations, and video (data) into **usable data**, which is called **information**.

Introduction to Computers

1. The data is inserted into **memory** using an **input device**.
2. The central processing unit (**CPU**) converts data to information.
3. The information is put on an **output device**.



Minimum Requirements for a Functional Computer System

1. A Keyboard
2. A Case containing a CPU and memory
3. A Monitor

Hardware

Definition

The term **Hardware** refers to the physical elements of a computer; the machinery or the electronics in a computer.

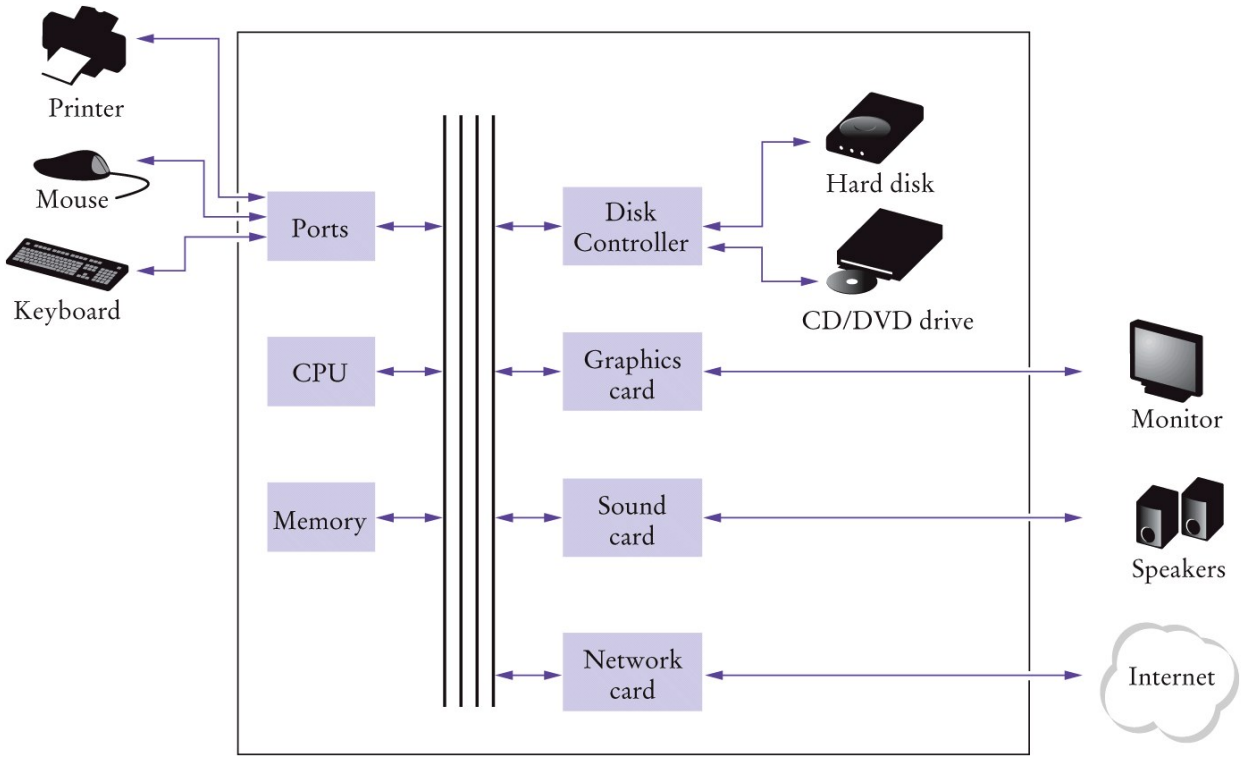
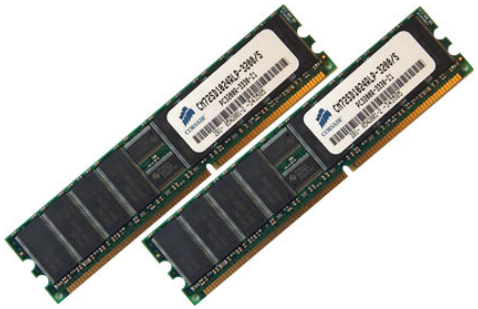


Figure 1.3

Hardware: examples

CPU	Memory
	
Motherboard	**Hard disk**
	

Numeral Systems

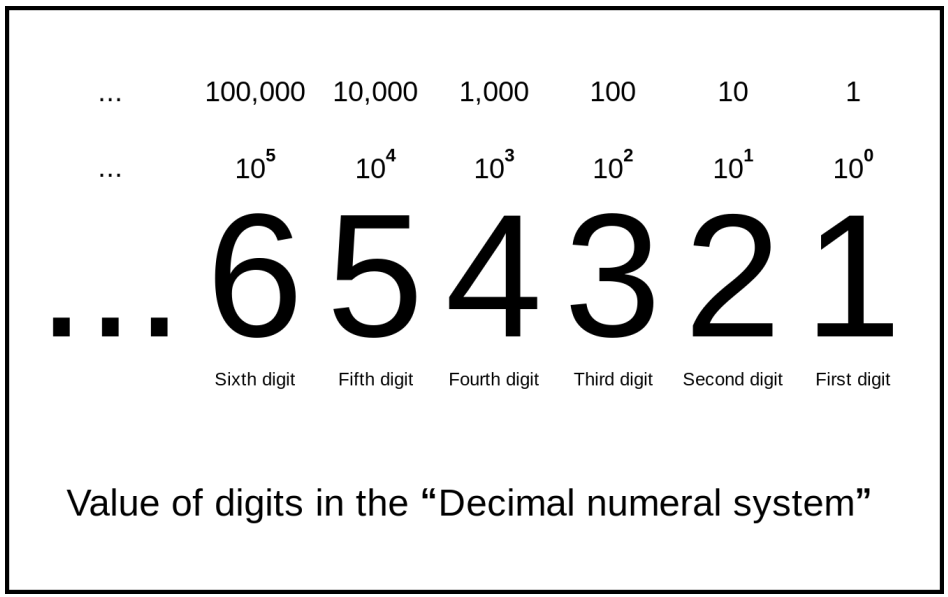
Definition

A writing method for expressing numbers is called a **numeral system**.

- The most common numeral systems are decimal (base-10), binary (base-2), hexadecimal (base-16) and octal (base-8).
- Because of its straightforward implementation in digital electronic circuitry using logic gates, the binary system is used internally by almost all modern computers.

Decimal Numeral Systems

The value of a digit is multiplied according to its placement in a numerical sequence (base-number ^ 0,1,2,3,...), from right to left.

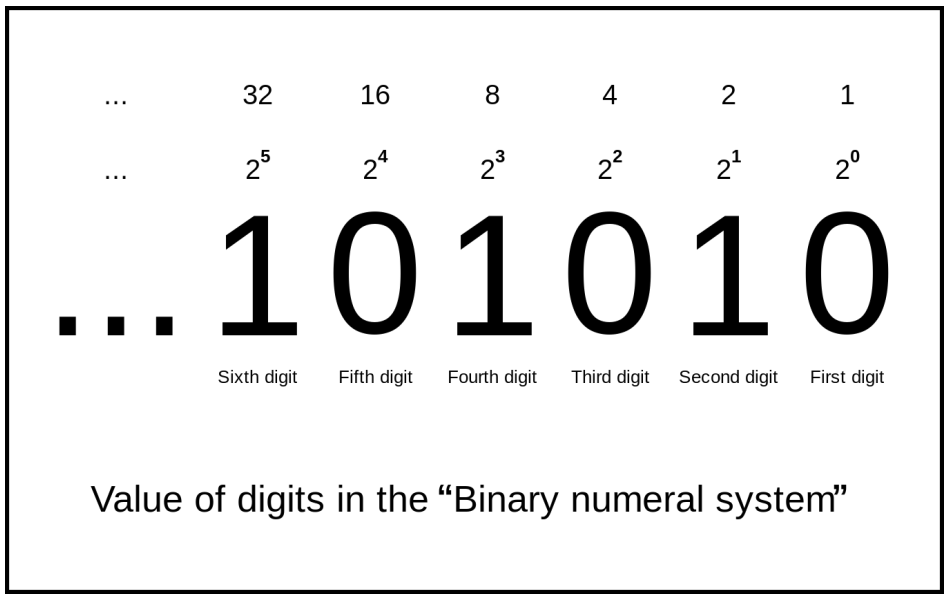


Value of digits in the “Decimal numeral system”

For example:

- 456 = (4 * 100) + (5 * 10) + (6 * 1) = 400 + 50 + 6
- 84568 = (8*10000) + (4 * 1000) + (5 * 100) + (6 * 10) + (8 * 1) = 80000 + 4000 + 500 + 60 + 8

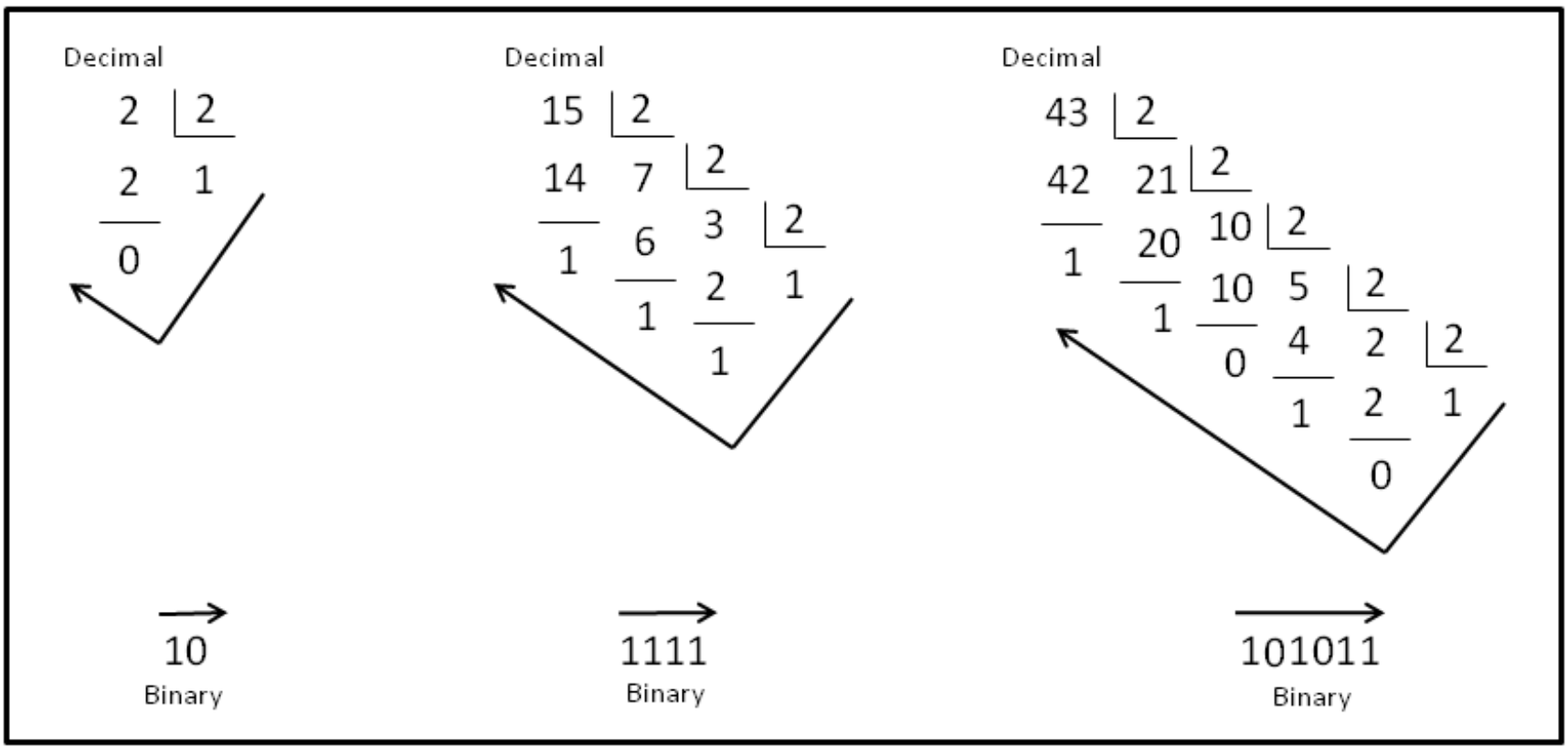
Binary Numeral Systems



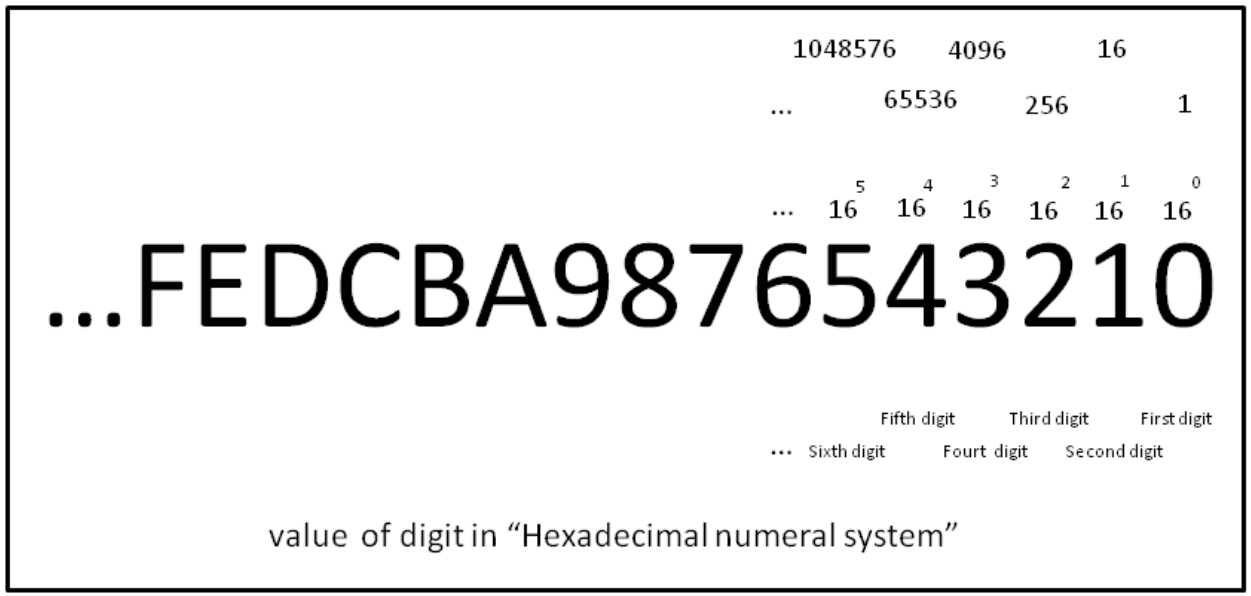
Converting Binary to Decimal:

- 11001 = (1*2^4) + (1*2^3) + (0*2^2) + (0*2^1) + (1*2^0) = 1*16 + 1*8 + 0*4 + 0*2 + 1*1 = 16 + 8 + 0 + 0 + 1 = 25

Converting Decimal to Binary



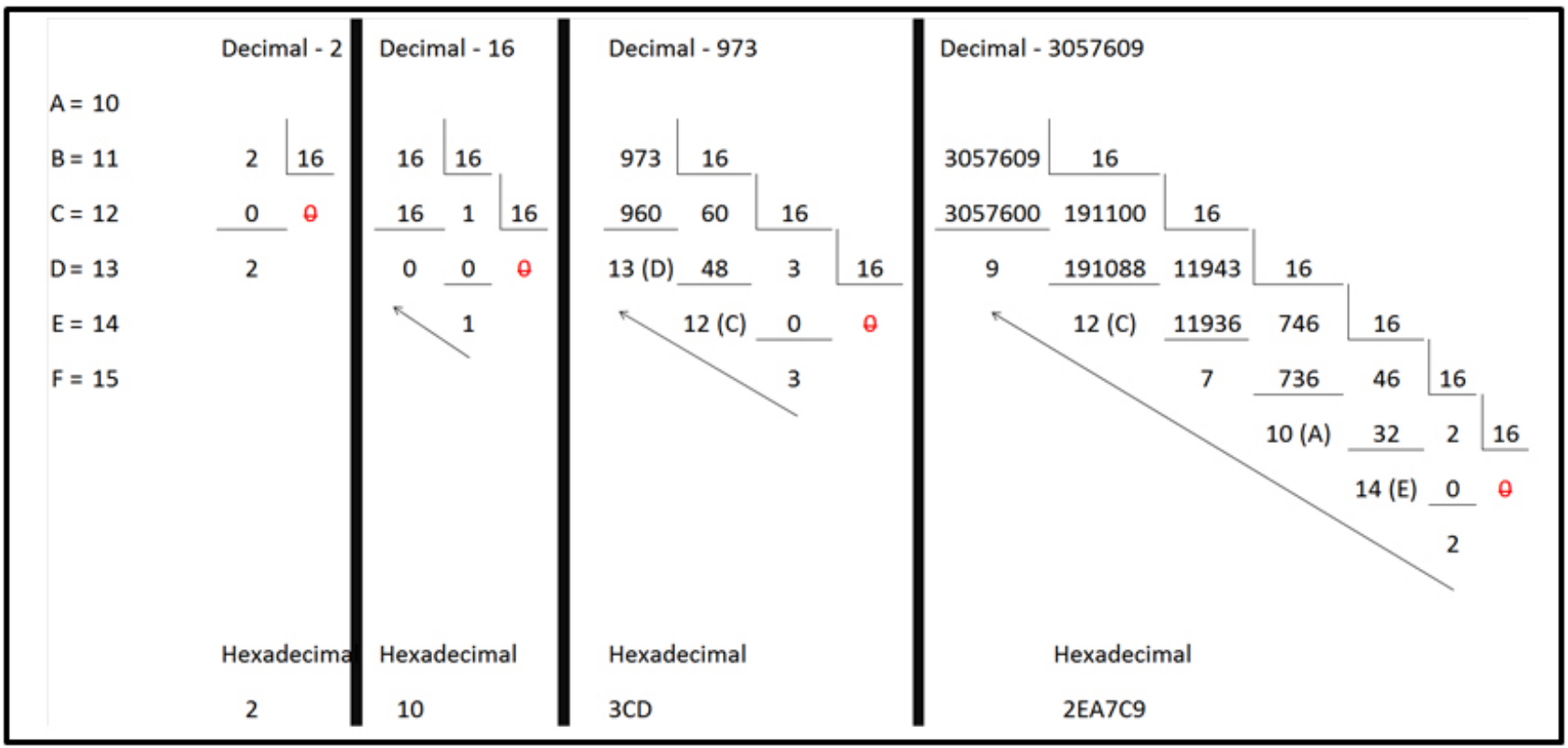
Hexadecimal Numeral Systems



Converting Hex to Decimal:

- AF85 = (10*16^3) + (15*16^2) + (8*16^1) + (5*16^0) = 10*4096 + 15*256 + 8*16 + 5*1 = 40960 + 3840 + 128 + 5 = 44933

Converting Decimal to Hex



Data Representation

Data in a computer is represented in a **series of bits**.

- **Bit**: A bit is a binary unit, simply a 1 or a 0. A true or a false. It is the most basic unit of data in a computer. **Bits are machine readable**.
- **Byte**: A byte equals 8 bits, and can be used to represent letters and numbers.
- **KiloByte (KB)**: A KB, is a unit of data that equals 1024 bytes (2^{10}).
- MegaByte (**MB**) = 2^{20}, GigaByte (**GB**) = 2^{30}, TeraByte (**TB**) = 2^{40}, PetaByte (**PB**) = 2^{50}

Characters in Computer

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Software

Definition

Software enables computer hardware to perform specific tasks. Software, commonly known as programs or applications, consists of all the electronic instructions that tell the hardware how to perform a task.

Software is capable of performing many different specific tasks, as opposed to hardware which only perform mechanical tasks that they are designed for. There are three major types of software:

- Operating System
- Application
- Programming

Operating System

Definition

An **operating system (OS)** is system software that manages computer hardware and software resources and provides common services for computer programs.

Examples:

- Microsoft Windows
- Linux
- OSX
- Android
- IOS

Computer Programs

Definition

A **Computer Program** is a sequence of instructions and decisions.

Definition

Programming is the act of designing and implementing computer programs.

When you '**run**' a program:

1. Program instructions and data are stored on the secondary memory.
2. When a program is started, it is brought into main memory.
3. The CPU runs the program one instruction at a time. The program may react to user input or perform some outputs.

Machine Code

Definition

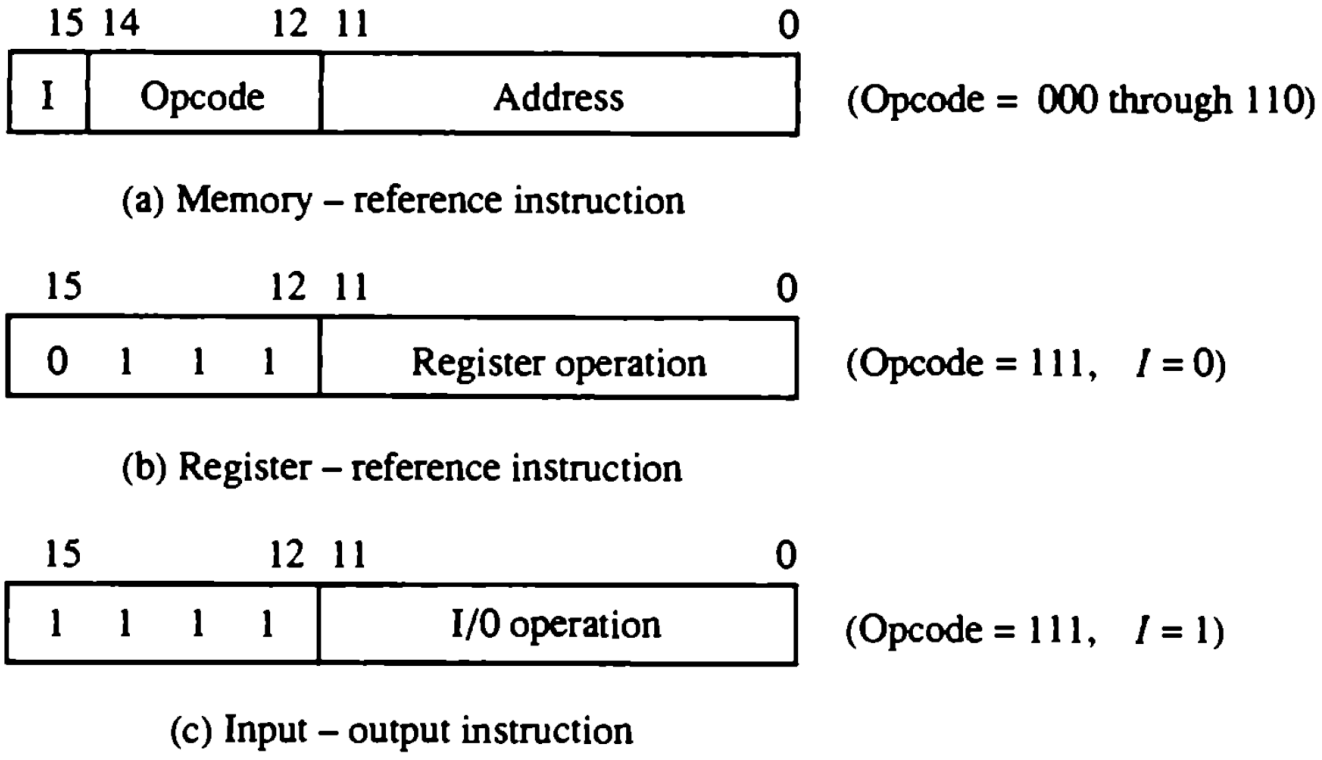
Machine code is the only language a computer can process directly without a previous transformation.
True machine code is a stream of raw, usually binary, data.

- Each CPU instruction is associated to a binary code.
- Writing programs in true Machine Code is too hard.
- A programmer coding in Machine Code normally codes instructions and data in a more readable form such as decimal, octal, or hexadecimal.

```
8B542408 83FA0077 06B80000 0000C383
FA027706 B8010000 00C353BB 01000000
C9010000 008D0419 83FA0376 078BD98B
B84AEBF1 5BC3
```

Basic Computer Instruction Format

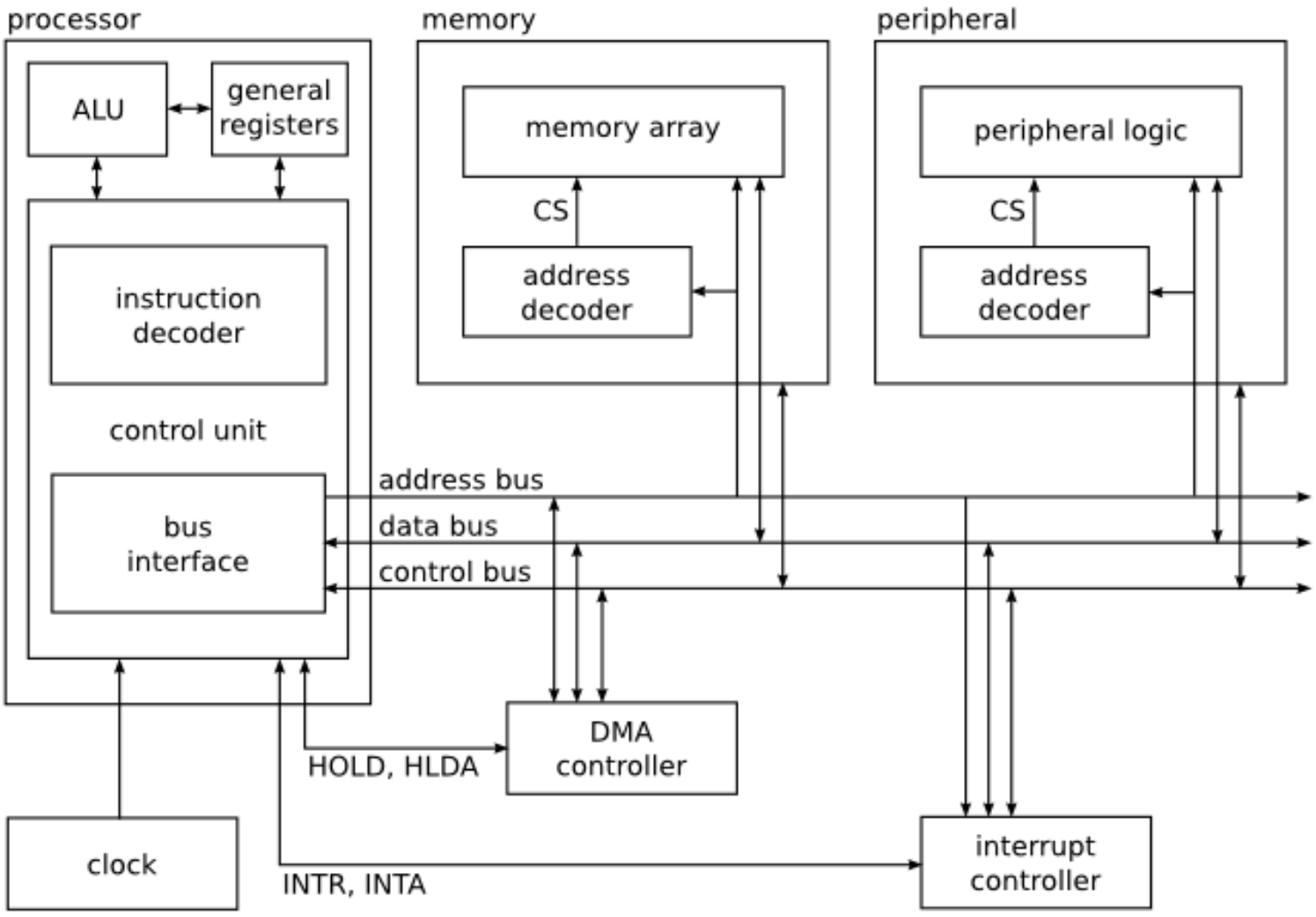
Figure 5-5 Basic computer instruction formats.



Basic Computer Instruction Codes

Symbol	Hexadecimal code $I = 0$	Hexadecimal code $I = 1$	Description
AND	0xxx	8xxx	AND memory word to *AC*
ADD	1xxx	9xxx	Add memory word to *AC*
LDA	2xxx	Axxx	Load memory word to *AC*
STA	3xxx	Bxxx	Store content of *AC* in memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero
CLA	7800		Clear *AC*
CLE	7400		Clear *E*
CMA	7200		Complement *AC*
CME	7100		Complement *E*
CIR	7080		Circulate right *AC* and *E*
CIL	7040		Circulate left *AC* and *E*
INC	7020		Increment *AC*
SPA	7010		Skip next instruction if *AC* positive
SNA	7008		Skip next instruction if *AC* negative
SZA	7004		Skip next instruction if *AC* zero
SZE	7002		Skip next instruction if *E* is 0
HLT	7001		Halt computer
INP	F800		Input character to *AC*
OUT	F400		Output character from *AC*
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Interrupt on
IOF	F040		Interrupt off

Computer Architecture



Computer Architecture

The CPU executes a program that is fetched more or less sequentially from the memory system. To execute each instruction a number of steps are taken:

1. Fetch the instruction from the location from memory system specified by the memory address register into the instruction register,
2. Decode the instruction, and increment the PC,
3. Fetch the operands,
4. Execute the desired operation using the ALU,
5. Optionally access memory,
6. Store the result in the desired location (could be a register or memory).

Assembly

- Second generation languages provide one abstraction level on top of the machine code.
- Assembly language has little semantics or formal specification, being only a mapping of human-readable symbols, including symbolic addresses, to opcodes, addresses, numeric constants, strings and so on.
- Typically, one machine instruction is represented as one line of assembly code.
- The **assembler** is in charge of translating your assembly code.

Assembly

```
fib:
    mov edx, [esp+8]
    cmp edx, 0
    ja @f
    mov eax, 0
    ret

    @@:
    cmp edx, 2
    ja @f
    mov eax, 1
    ret

    @@:
    push ebx
    mov ebx, 1
    mov ecx, 1

    @@:
        lea eax, [ebx+ecx]
        cmp edx, 3
        jbe @f
        mov ebx, ecx
        mov ecx, eax
        dec edx
    jmp @b

    @@:
    pop ebx
    ret
```

High-level Languages

Definition

In computer science, a **high-level programming language** is a programming language with strong abstraction from the details of the computer.

Low-level V.S. High-level

Rather than dealing with registers, memory addresses and call stacks, high-level languages deal with variables, arrays, objects, complex arithmetic or boolean expressions, subroutines and functions, loops, threads, locks, and other abstract computer science concepts.

High-level Languages: Execution Modes

- **Interpreted**: When code written in a language is interpreted, its syntax is read and then executed directly, with no compilation stage.
- **Compiled**: When code written in a language is compiled, its syntax is transformed into an executable form before running. There are two types of compilation:
  - Machine code generation
  - Intermediate representations
- **Source-to-Source Translated**: Code written in a language may be translated into terms of a lower-level programming language for which native code compilers are already widely available.

C++

Bjarne Stroustrup

- In this course we will learn C++ programming language, which was developed by Bjarne Stroustrup in the 1980s.
- Over the years, C++ has grown by the addition of many features.
  - A standardization process culminated in the publi cation of the international C++ standard in 1998.
  - Just as importantly, C++ is increasingly used for program ming ”embedded systems”, small computers that control devices such as automobile engines or cellular telephones.