

IT111 - Introduction to Information Technology Bachelor of Science in Information Technology

Eng. Eman Hamdy

Lecture 3 - Data Storage



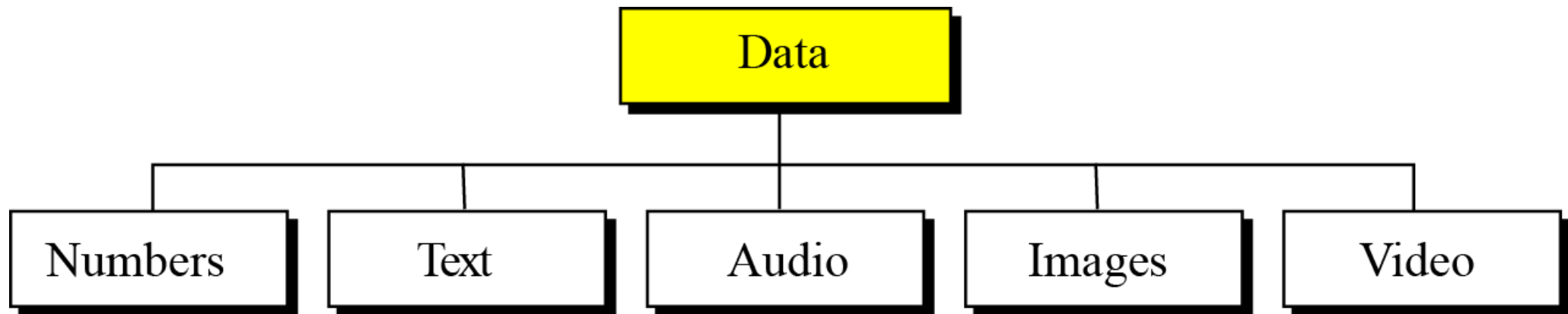
مجموعة جامعة
الجميع الذكية
للتعليم القابضة

Objectives

Upon successful completion of this course, students should be able to:

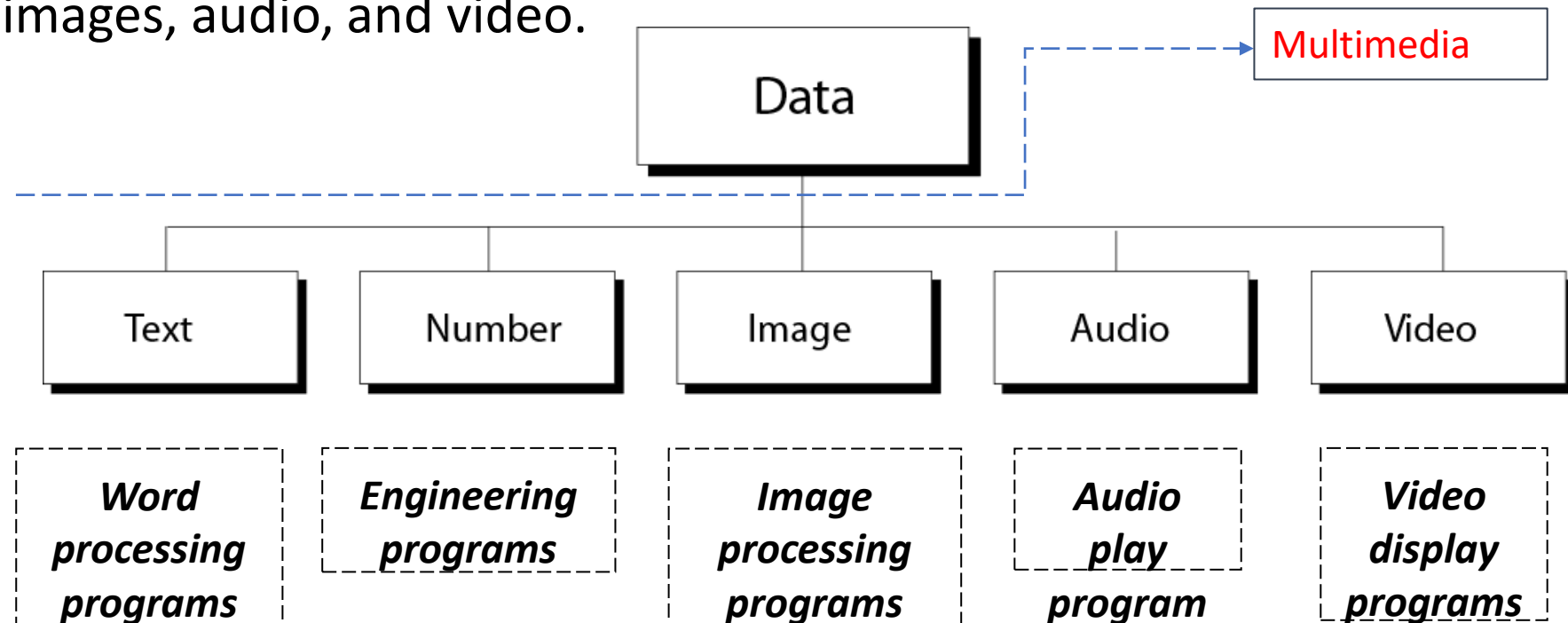
- ☐ List five different data types used in a computer.
- ☐ Describe different methods to store integers inside a computer.
- ☐ Describe how reals are stored in a computer.

- Data today come in different forms including numbers, text, audio, image, and video.



Different types of data

The computer industry uses the term “multimedia” to define information that contains numbers, text, images, audio, and video.



- A bit is the smallest unit of data that the computer deals with.
- a bit can take two values (0 or 1).
- A two-state electrical switch (transistor) is used to represent a bit (on state $\rightarrow$ 1, off state $\rightarrow$ 0).
- To store 16 bits you need 16 switches, to store million bits you need million switches.
- In computer memory data are stored as blocks of bits (bit-patterns), the length of bit-patterns is the number of bits in the bit-patterns.
- A bit-pattern of 8 bits length is called a byte

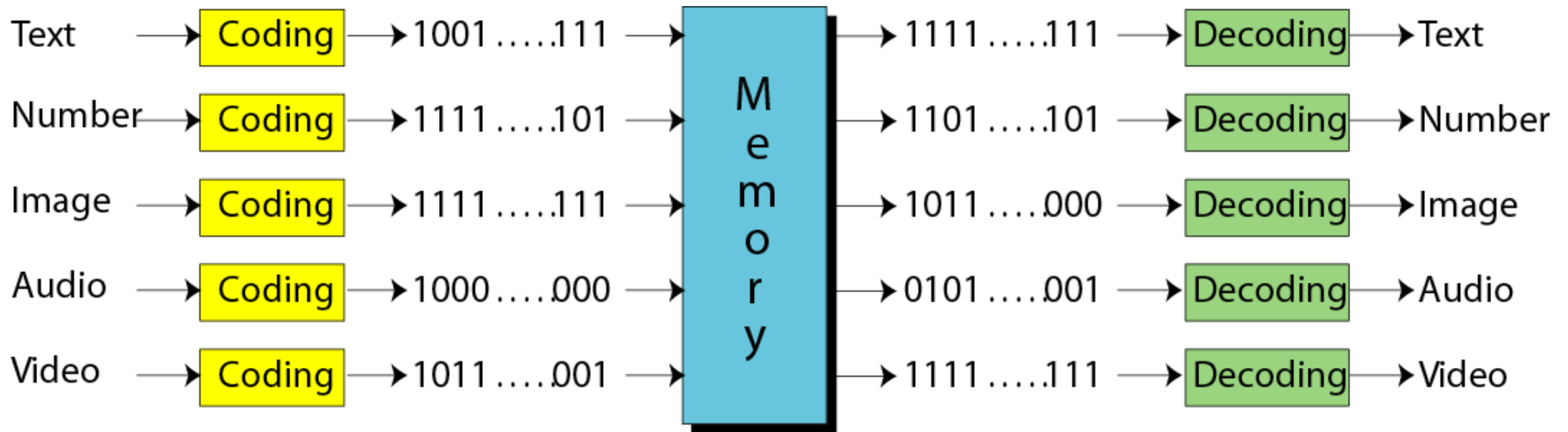
All data types are transformed into a uniform representation when they are stored in a computer and transformed back to their original form when retrieved. This universal representation is called a **bit pattern (string of bits)**.

1 0 0 0 1 0 1 0 1 1 1 1 1 1

A bit pattern

- Human senses deal with a variety of information (signals).
- Input devices of computer translates these information into electrical signals.
- Electrical signals are then translated into universal format (0s,1s), this is known as coding.
- After processing, output devices transform back data into their original form, this is known as decoding

Information coding and decoding



A **number** is changed to the binary system before being stored in the computer memory, as described in Chapter 2. However, there are still two issues that need to be handled:

1. How to store the sign of the number.
2. How to show the decimal point.

Computers use two different representation:

- fixed-point (store a number as an integer)
- floating-point (store a number as real)

An **unsigned integer** is an integer that can never be negative and can take only 0 or positive values. Its range is between 0 and positive infinity.

An input device stores an unsigned integer using the following steps:

1. The integer is changed to binary.
2. If the number of bits is:
 - less than $n \rightarrow$ 0s are added to the left.
 - Greater than $n \rightarrow$ the integer cannot be stored (overflow).

Store 7 in an 8-bit memory location using unsigned representation.

Solution

First change the integer to binary (Convert from Decimal to binary)

$$7/2 = 3.5 \quad \text{Remainder } 1$$

$$3/2 = 1.5 \quad \text{Remainder } 1$$

$$1/2 = 0.5 \quad \text{Remainder } 1$$

$$(7)_{10} = (111)_2$$

Add five 0s to make a total of eight bits, $(00000111)_2$.

The integer is stored in the memory location.

Example

Change 7 to binary

→

1 1 1

Add five bits at the left

→

0 0 0 0 0 1 1 1

Example

Store 258 in a 16-bit memory location using unsigned representation.

Solution

First change the integer to binary

$(258)_{10}$

$258/2 = 129$ Remainder 0

$129/2 = 64.5$ Remainder 1

$64/2 = 32$ Remainder 0

$32/2 = 16$ Remainder 0

$16/2 = 8$ Remainder 0

$8/2 = 4$ Remainder 0

$4/2 = 2$ Remainder 0

$2/2 = 1$ Remainder 0

$1/2 = 0.5$ Remainder 1

$(258)_{10} = (100000010)_2$

$(100000010)_2$.

Add seven 0s to make a total of sixteen bits, $(0000000100000010)_2$. The integer is stored in the memory location.

Change 258 to binary $\rightarrow$ 1 0 0 0 0 0 0 1 0

Add seven bits at the left $\rightarrow$ 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0

What is returned from an output device when it retrieves the bit string 00101011 stored in memory as an unsigned integer.

Solution

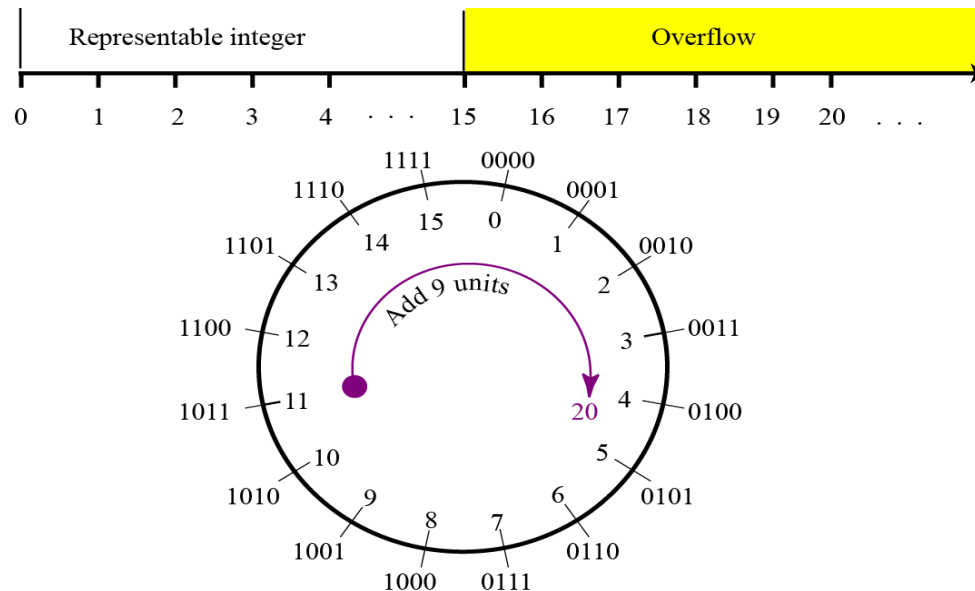
The binary integer is converted to the unsigned integer 43.

binary 0 0 1 0 1 0 1 1

Weight 128 64 32 16 8 4 2 1

$$= 32+8+2+1= 43$$

Figure shows what happens if we try to store an integer that is larger than $2^4 - 1 = 15$ in a memory location that can only hold four bits.



$$20/2 = 10 \rightarrow 0$$

$$10/2 = 5 \rightarrow 0$$

$$5/2 = 2 \rightarrow 1$$

$$2/2 = 1 \rightarrow 0$$

$$1/2 = 0 \rightarrow 1$$

$$(20)_{10} = (10100)_2$$

In this method, the available range for unsigned integers (0 to $2^n - 1$) is divided into two equal sub-ranges. The first half represents positive integers, the second half, negative integers.

In sign-and-magnitude representation, the leftmost bit defines the sign of the integer.

*If it is **0**, the integer is **positive**.*

*If it is **1**, the integer is **negative**.*

Store +28 in an 8-bit memory location using sign-and-magnitude representation.

Solution

The integer is changed to 7-bit binary. The leftmost bit is set to 0. The 8-bit number is stored.

$$28/2 = 14 \rightarrow 0$$

$$14/2 = 7 \rightarrow 0$$

$$7/2 = 3.5 \rightarrow 1$$

$$3/2 = 1.5 \rightarrow 1$$

$$1/2 = 0.5 \rightarrow 1$$

$$(28)_{10} = (11100)_2$$

Change 28 to 7-bit binary

0 0 1 1 1 0 0

Add the sign and store

0

0 0 1 1 1 0 0

Store -28 in an 8-bit memory location using sign-and-magnitude representation.

Solution

The integer is changed to 7-bit binary. The leftmost bit is set to 1. The 8-bit number is stored.

Change 28 to 7-bit binary

0 0 1 1 1 0 0

Add the sign and store

1

0 0 1 1 1 0 0

Retrieve the integer that is stored as 01001101 in sign-and-magnitude representation.

Solution

Since the leftmost bit is 0, the sign is positive. The rest of the bits (1001101) are changed to decimal

$$\begin{array}{r} \text{Binary (1 0 0 1 1 0 1)}_2 \\ \text{Weight (64 32 16 8 4 2 1)} \\ = 64 + 8 + 4 + 1 = (77)_{10} \end{array}$$

Retrieve the integer that is stored as 10100001 in sign-and-magnitude representation.

Solution

Since the leftmost bit is 1, the sign is negative. The rest of the bits (0100001) are changed to decimal as 33. After adding the sign, the integer is -33 .

Two's complement representation

- Almost all computers use two's complement representation to store a signed integer in an n-bit memory location.
- Two operations: One's complement and Two's complement.

- This operation can be applied to any integer, positive or negative.
- This operation simply reverses (flips) each bit.
- **A 0-bit is changed to a 1-bit, a 1-bit is changed to a 0-bit.**
- The following shows how take the one's complement of the integer 00110110.

Original pattern

0 0 1 1 0 1 1 0

After applying one's complement operation

1 1 0 0 1 0 0 1

- The second operation is called *two's completing* or taking the two's complement of an integer in binary.
- **This operation is done in two steps.**
- **First, we copy bits from the right until a 1 is copied;**
- **then, we flip the rest of the bits.**

The following shows how we take the two's complement of the integer 00110100.

Original integer

0 0 1 1 0 1 0 0

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

Two's complementing once

1 1 0 0 1 1 0 0

The following shows that we always get the original integer if we apply the two's complement operation twice.

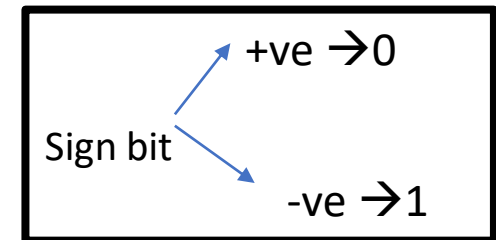
Original integer	0	0	1	1	0	1	0	0
	↓	↓	↓	↓	↓	↓	↓	↓
Two's complementing once	1	1	0	0	1	1	0	0
	↓	↓	↓	↓	↓	↓	↓	↓
Two's complementing twice	0	0	1	1	0	1	0	0

Find the 1's complement and 2's complement for each of the following binary numbers:

Binary number	1's complement (invert all bits)	2's complement
10101010	01010101	01010110
01110001	10001110	10001111

To store an integer in two's complement:-

1. the integer is changed to n-bits binary.
2. If the integer is Zero or positive, $\rightarrow$ it is stored as it is
3. If the integer is negative $\rightarrow$ take two's complement $\rightarrow$ store it.

**To retrieve an integer in two's complement:-**

- 1.If the leftmost bit is 1 $\rightarrow$ apply two's complement
- 2.If the leftmost bit is 0 $\rightarrow$ no operation applied

Store the integer 28 in an 8-bit memory location using two's complement representation.

Solution

The integer is positive (no sign means positive), so after decimal to binary transformation no more action is needed. Note that five extra 0s are added to the left of the integer to make it eight bits.

Change 28 to 8-bit binary

0 0 0 1 1 1 0 0

Store -28 in an 8-bit memory location using two's complement representation.

Solution

The integer is negative, so after changing to binary, the computer applies the two's complement operation on the integer.

Change 28 to 8-bit binary

0 0 0 1 1 1 0 0

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

Apply two's complement operation

1 1 1 0 0 1 0 0

Retrieve the integer that is stored as 00001101 in memory in two's complement format.

Solution

The leftmost bit is 0, so the sign is positive. The integer is changed to decimal, and the sign is added.

Leftmost bit is 0. The sign is positive

0 0 0 0 1 1 0 1

Integer changed to decimal

13

Sign is added (optional)

+13

Retrieve the integer that is stored as 11100110 in memory using two's complement format.

Solution

The leftmost bit is 1, so the integer is negative. The integer needs to be two's complemented before changing to decimal.

Leftmost bit is 1. The sign is negative

1	1	1	0	0	1	1	0
↓	↓	↓	↓	↓	↓	↓	↓

Apply two's complement operation

0	0	0	1	1	0	1	0
---	---	---	---	---	---	---	---

Integer changed to decimal

26

Sign is added

-26

Change the following decimal numbers to 8-bit two's complement integers
-12

Convert 12 to binary	0	0	0	0	1	1	0	0
Apply two's complement operation	1	1	1	1	0	1	0	0

Integer Representations

Table 3.1 Summary of integer representations

<i>Contents of memory</i>	<i>Unsigned</i>	<i>Two's complement</i>
0000	0	+0
0001	1	+1
0010	2	+2
0011	3	+3
0100	4	+4
0101	5	+5
0110	6	+6
0111	7	+7
1000	8	-8
1001	9	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1



Thank you