Information and Data

MODULE 3 / UNIT 9 - 2 / 1.1

MOISES M. MARTINEZ

FUNDAMENTALS OF COMPUTER ENGINEERING

2024/25

Floating-point numbers

01

Floating-point numbers

Real numbers are approximately represented in computers using the IEEE-754 standard, which employs a method similar to scientific notation. In this standard, a real number is represented by a fixed-precision integer, known as the **significand** (or mantissa), which is scaled by an **integer exponent** of a fixed **base**, typically base 2.

Example: If we want to represent 14.345 as floating point number in decimal base:

$$14.345 = \underbrace{\boxed{14345}}_{\text{significand}} \times \overset{\text{base}}{\boxed{10}}{}^{\overset{\text{exponent}}{\boxed{-3}}}$$

This allows the representation of a wide range of values, albeit with some limitations in precision due to the fixed number of bits available for both the significand and the exponent.

Floating-point numbers

In computer systems, floating-point numbers are expressed through a finite bit sequence that encompasses three distinct components:

- Sign-bit (1 single bit): This component indicates the sign of the number where a value of 1 to indicate a negative number and a value of 0 to indicate a positive number, analogous to how integers are represented.
- Significand, Mantissa or Fraction (23 bits in the case of single precision floating point): The significand represents the precision of the floating-point number and **is usually normalized**, meaning the leading digit is implicitly assumed to be 1 (for non-zero numbers).
- Exponent (8 bits in single precision floating point): The exponent determines the scale of the number by indicating the power of the base (usually 2) by which the significand should be multiplied. **It is stored in a biased form, allowing both positive and negative exponents**.

Floating-point numbers

Floating-point numbers are categorized into two distinct formats based on the aforementioned three components: single precision, which employs 32 bits, and double precision, which utilizes 64 bits.

32 bits

Sign | Exponent | Significand/Mantissa

1 bit | 8 bits | 23 bits

single precision

64 bits

Sign | Exponent | Significand/Mantissa

1 bit | 11 bits | 52 bits

double precision

Floating-point numbers

To convert a decimal number into its IEEE 754 Floating Point Representation, you must follow the next steps:

1. Select the desired precision: single-precision (32-bit) or double-precision (64-bit) representation.
2. Divide the number into its integral (whole) part and fractional (decimal) part.
3. Convert the integral (exponent) part of the number to its binary equivalent.
4. Convert the factional (decimal) part of the number to its binary equivalent.
5. Merge the binary representations of the integral and fractional parts to form a single binary number.
6. Normalize the binary number adjusting the binary number to the form 1.XXXXXXXXX x 2^n by shifting the decimal point and adjusting the exponent accordingly.
7. Determine the sign-bit: Assign 0 if the original number is positive, or 1 if it is negative.

Floating-point numbers

To convert a decimal number into its IEEE 754 Floating Point Representation, you must follow the next steps:

8. Normalize the binary number adjusting the binary number to the form 1.XXXXXXXXX x $2n$ (Scientific Notation) by shifting the decimal point and adjusting the exponent accordingly.
9. Compute the exponent by adding the bias (127 for single-precision, 1023 for double-precision) to the exponent from the normalization step.
10. Create the Final IEEE 754 Representation combining of the sign bit, the biased exponent, and the mantissa (**fractional part of the normalized binary number, excluding the implicit leading 1**).

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 88.125 from decimal to binary?

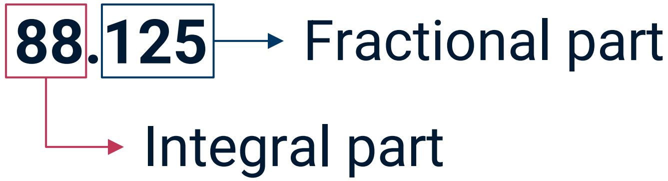
- **Which precision we want to use?** **single (32 bits)**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 88.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**



Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 88.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

88.125 → Fractional part

→ Integral part

88							/	2
0	44						/	2
	0	22					/	2
		0	11				/	2
			1	5			/	2
				1	2		/	2
					0	1	/	2
							/	2
0	0	0	1	1	0	1		

88 = 1011000

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 88.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

88.125 → Fractional part: .125

Integral part: 88

0.125 * 2 = 0.25

0 + 0.25 * 2 = 0.5

0 + 0.5 * 2 = 1.0

1 + 0.0 * 2 =

This process finish when we reach 1 + 0.0, this means we have normalized the binary number so that the mantissa (significand) is 1.0 followed by any remaining bits.

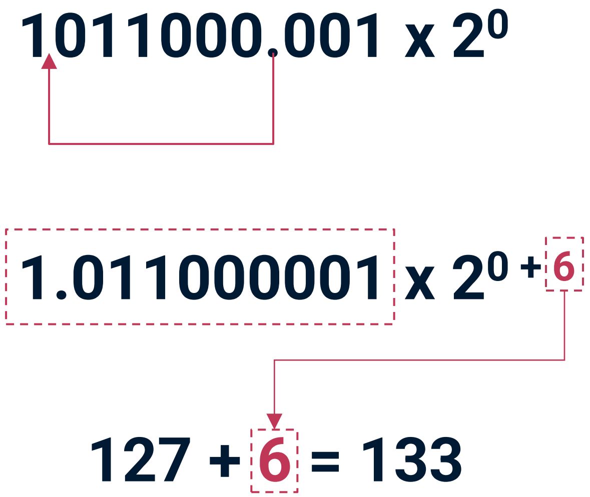
Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 88.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

88.125



$$1011000.001 \times 2^0$$

Shift the decimal point six positions to the left in order to position a single '1' in that location.

$$\boxed{1.011000001} \times 2^{0+\boxed{6}}$$

$$127 + \boxed{6} = 133$$

Specific biases are established for both single and double precision.

In the case of single precision, the exponent bias is set at 127. Consequently, the exponent derived earlier must be added to this bias value. Thus, the exponent to be utilized in this context is 133, calculated as 127 plus 6.

Why do you think we shift the decimal point?

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 88.125 from decimal to binary?

The shifting during the conversion process is based on the floating-point scientific notation.

For example, the numerical value 0.0015 can be expressed as 1.5 x 10−3, which is often written in scientific notation as 1.5e-3. In this notation, "e-3" signifies "10 raised to the power of minus three" indicating that the decimal point is shifted three places to the left. This exponentiated value is then multiplied by 1.5.

1.5e-3 is equivalent to the decimal value 0.0012.

3454.233434 = 3.454233434e+3

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 88.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

127 + 6 = 133

133								/	**2**
1	66							/	**2**
	0	33						/	**2**
		1	16					/	**2**
			0	8				/	**2**
				0	4			/	**2**
					0	2	2	/	**2**
						0	1	/	**2**
1	0	1	0	0	0	0	1		

133 = 10000101

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 88.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

+ → 0

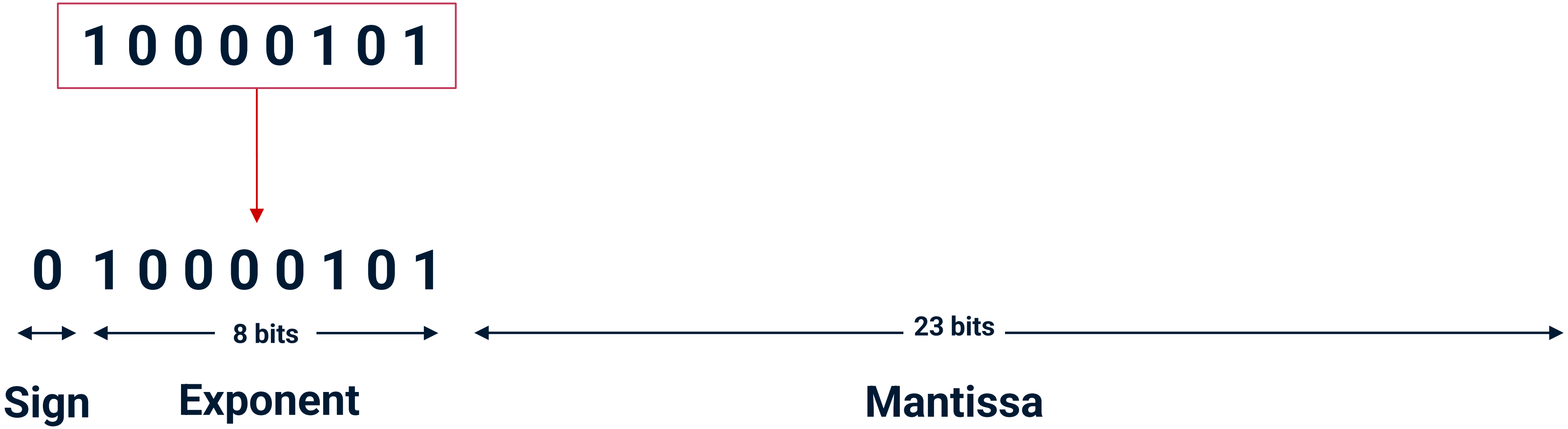
- → 1

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 88.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**



Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 88.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

1.01100001×2^6

Drop the leading 1 on the left and copy the decimal portion of the number that is being multiplied by 2.

0 10000101 01100001

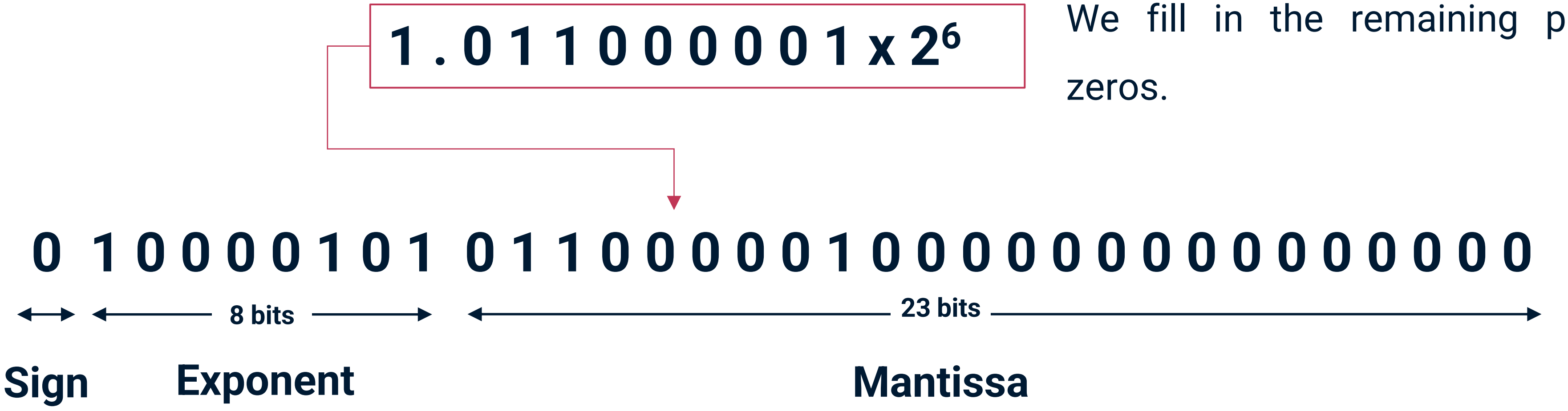
Sign	Exponent	Mantissa
0	10000101	01100001
	8 bits	23 bits

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 88.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**



$1 . 0 1 1 0 0 0 0 0 1 \times 2^6$

We fill in the remaining positions with zeros.

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 25.025 from decimal to binary?

- **Which precision we want to use?**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 25.025 from decimal to binary?

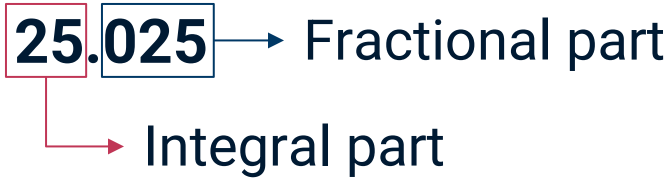
- **Which precision we want to use?** **single (32 bits)**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 25.025 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**



25.025

Fractional part

Integral part

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 25.025 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

25.025 → Fractional part

25 → Integral part

25					/	**2**
1	12				/	**2**
	0	6			/	**2**
		0	3		/	**2**
			1	1	/	**2**
1	0	0	1	1		

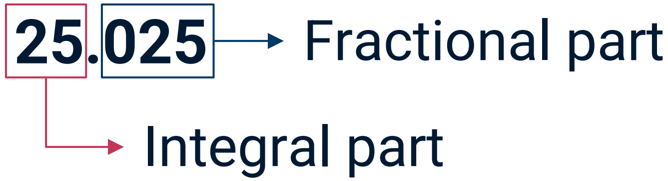
25 = 11001

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 25.025 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**



25.025

Fractional part

Integral part

0.025									*	2	=	0.05
0	0.05								*	2	=	0.1
	0	0.1							*	2	=	0.2
		0	0.2						*	2	=	0.4
			0	0.4					*	2	=	0.8
				0	0.8				*	2	=	1.6
					1	0.6			*	2	=	1.2
						1	0.2		*	2	=	0.4
							0	0.4	*	2	=	0.8

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 25.025 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

25.025 → Fractional part

25 → Integral part

0.025 * 2 = 0.05
0 0.05 * 2 = 0.1
0 0.1 * 2 = 0.2
0 0.2 * 2 = 0.4
0 0.4 * 2 = 0.8
0 0.8 * 2 = 1.6
1 0.6 * 2 = 1.2
1 0.2 * 2 = 0.4
0 0.4 * 2 = 0.8

There is a pattern.

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 25.025 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

25.025 → Integral part

→ Integral part

Stop this process once we have accumulated 23 bits in our mantissa.

	0.025	*	2	=	0.05
0	0.05	*	2	=	0.1
0	0.1	*	2	=	0.2
0	0.2	*	2	=	0.4
0	0.4	*	2	=	0.8
0	0.8	*	2	=	1.6
1	0.6	*	2	=	1.2
1	0.2	*	2	=	0.4
0	0.4	*	2	=	0.8

There is a pattern.

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 25.025 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

25.025

11001.000 0011 0011 0011 0011 0011 0 x 2^{0} **Move decimal point 4 places to left to let one 1 there.**

1.10010000 0110 0110 0110 0110 0110 x $2^{0+\boxed{4}}$

$$127 + \boxed{4} = 131$$

Specific biases are established for both single and double precision.

In the case of single precision, the exponent bias is set at **127**.

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 25.025 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

127 + 4 = 131

131								/	**2**
1	65							/	**2**
	1	32						/	**2**
		0	16					/	**2**
			0	8				/	**2**
				0	4			/	**2**
					0	2	2	/	**2**
						0	1	/	**2**
1	1	0	0	0	0	0	1		

131 = 10000011

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 25.025 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

+ → 0

- → 1

It is positive.

0

8 bits

23 bits

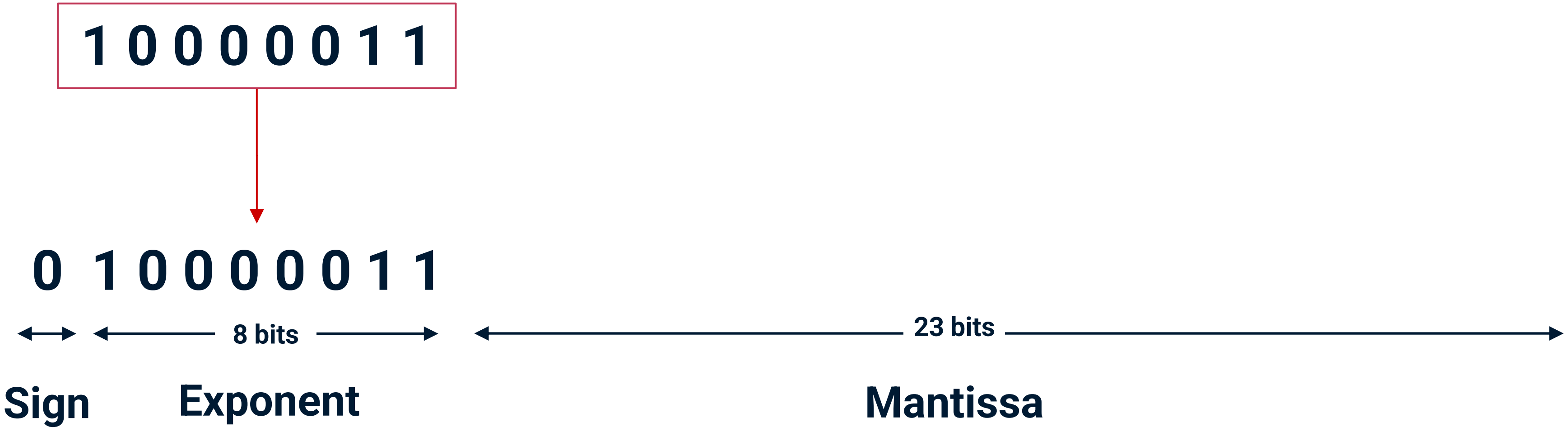
Sign **Exponent** **Mantissa**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 25.025 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**



Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 25.025 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

Drop the leading 1 on the left and copy the decimal portion of the number that is being multiplied by 2.

1. 1 0 0 1 0 0 0 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 x 2^4

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

8 bits

23 bits

Sign **Exponent** **Mantissa**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 0.5 from decimal to binary?

- **Which precision we want to use?**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 0.5 from decimal to binary?

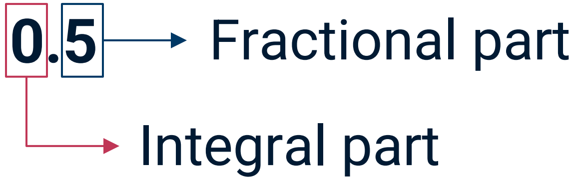
- **Which precision we want to use?** **single (32 bits)**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 0.5 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

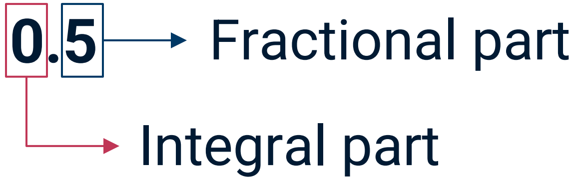


Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 0.5 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**



0 **/ 2**

0 0

0

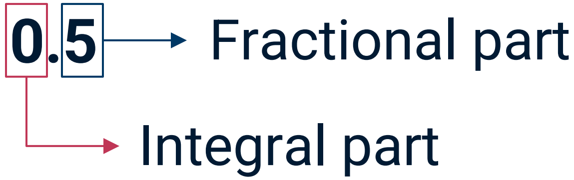
0 = 0

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 0.5 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**



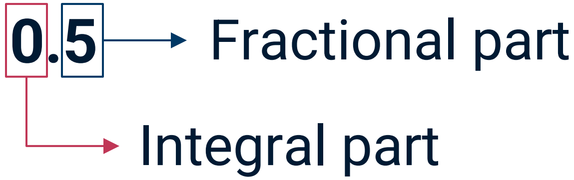
0.5 * 2 = 1

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 0.5 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**



Stop this process once we have 1.0.

0.5 * 2 = 1.0

1 0.0

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 0.5 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

0.5

$$0.100\ 0000\ 0000\ 0000\ 0000\ 000 \times 2^{0}$$

Move decimal point -1 places to right to let one 1 there.

$$01.00\ 0000\ 0000\ 0000\ 0000\ 000 \times 2^{0+\boxed{-1}}$$

$$127 + \boxed{-1} = 126$$

Specific biases are established for both single and double precision.

In the case of single precision, the exponent bias is set at **127**.

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 0.5 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

127 + -1 = 126

126								/	2
0	63							/	2
	1	31						/	2
		1	15					/	2
			1	7				/	2
				1	3			/	2
					1	1		/	2
						1	0		
0	1	1	1	1	1	1			

126 = 1111110

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 0.5 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

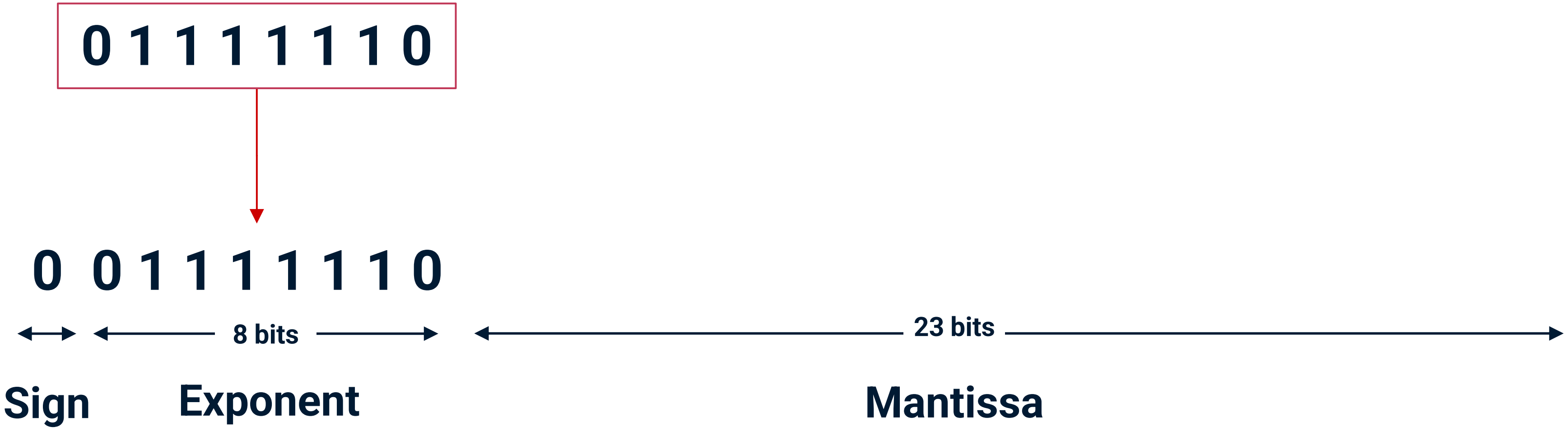
It is positive.

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 0.5 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**



0 1 1 1 1 1 1 0

0 0 1 1 1 1 1 1 0

8 bits

23 bits

Sign

Exponent

Mantissa

Single IEEE 754 Floating Point Representation

How do you convert 0.5 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

Drop the leading 1 on the left and copy the decimal portion of the number that is being multiplied by 2.

0 1. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 x 2^{-1}

0 0 1 1 1 1 1 1 0

8 bits

23 bits

Sign **Exponent** **Mantissa**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 0.5 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

Drop the leading 1 on the left and copy the decimal portion of the number that is being multiplied by 2.

0 1. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 x 2^{-1}

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

8 bits

23 bits

Sign **Exponent** **Mantissa**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert -0.125 from decimal to binary?

- **Which precision we want to use?**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert -0.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert -0.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

0.125 → Fractional part

0 → Integral part

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert -0.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

0.125 → Fractional part

Integral part

0 / **2**

0 0

0

0 = 0

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert -0.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

0.125 → Fractional part

Integral part

0.5 * 2 = 1

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert -0.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

$\boxed{0}.\boxed{125} \rightarrow$ Fractional part

Integral part

Stop this process once we have 1.0.

0.125				*	2	=	0.25
0	0.25			*	2	=	0.50
	0	0.50		*	2	=	1.0
		1	0.0				

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert -0.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

0.125

$$0.001\ 0000\ 0000\ 0000\ 0000\ 000 \times 2^{0}$$

Move decimal point -3 places to right to let one 1 there.

$$0001.\ 0000\ 0000\ 0000\ 0000\ 000 \times 2^{0+\boxed{-3}}$$

$$127 + \boxed{-3} = 124$$

Specific biases are established for both single and double precision.

In the case of single precision, the exponent bias is set at **127**.

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert -0.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

127 + -3 = 124

124								**/**	**2**
0	62							**/**	**2**
	0	31						**/**	**2**
		1	15					**/**	**2**
			1	7				**/**	**2**
				1	3			**/**	**2**
					1	1		**/**	**2**
						1	0		
0	0	1	1	1	1	1			

125 = 1111100

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert -0.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

+ → 0

\- → 1

It is positive.

1

8 bits

23 bits

Sign **Exponent** **Mantissa**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert -0.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

0 1 1 1 1 1 0 0

1 0 1 1 1 1 1 0 0

8 bits

23 bits

Sign

Exponent

Mantissa

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert -0.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

Drop the leading 1 on the left and copy the decimal portion of the number that is being multiplied by 2.

$0\ 0\ 0\ 1.\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \times 2^{-3}$

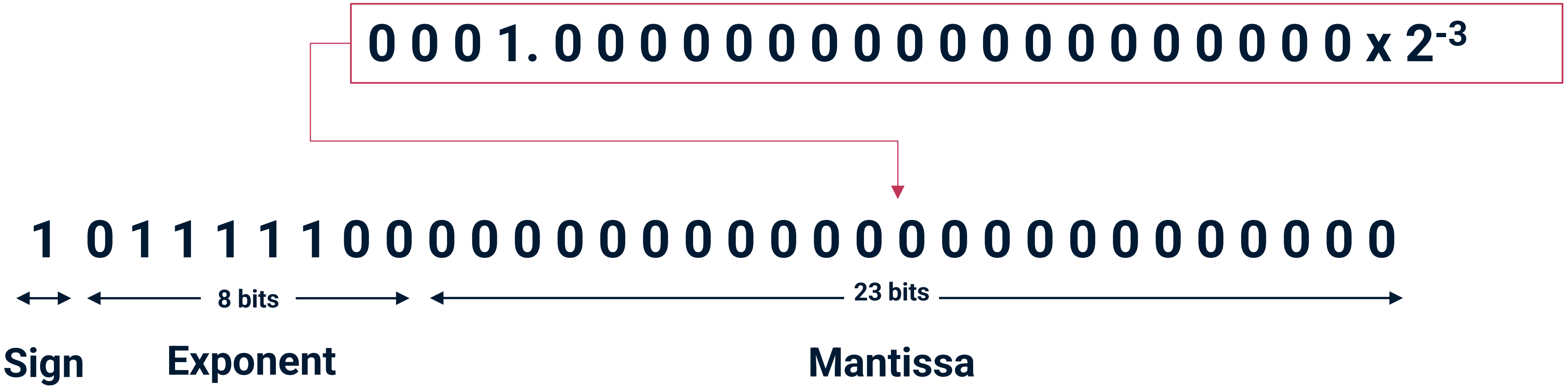
Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert -0.125 from decimal to binary?

- **Which precision we want to use?** **single (32 bits)**

Drop the leading 1 on the left and copy the decimal portion of the number that is being multiplied by 2.



$0\ 0\ 0\ 1.\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \times 2^{-3}$

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

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000010100010000000000000000000 from binary to decimal?

- **Which precision we must to use?**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000010100010000000000000000000 from binary to decimal?

- **Which precision we must to use?** **single (32 bits)**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000010100010000000000000000000 from binary to decimal?

- **Which precision we must to use?** **single (32 bits)**

We split the number in its parts (sign, exponent and mantissa).

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

8 bits

23 bits

Sign **Exponent** **Mantissa**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000010100010000000000000000000 from binary to decimal?

- **Which precision we must to use?** single (32 bits)

+ → 0

\- → 1

It is negative.

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

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000010100010000000000000000000 from binary to decimal?

- **Which precision we must to use?** **single (32 bits)**

We transform the exponent to decimal to find the normalization of the mantissa.

1 0 0 0 0 1 0 1 = 133 ➔ 133 – 127 = 6

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

↔ ← 8 bits → ← 23 bits →

Sign **Exponent** **Mantissa**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000010100010000000000000000000 from binary to decimal?

- **Which precision we must to use?** **single (32 bits)**

We denormalize the mantissa adding a 1 on the left.

1. 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

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

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000010100010000000000000000000 from binary to decimal?

- **Which precision we must to use?** **single (32 bits)**

We move decimal point 6 places to the right.

1 0 0 0 1 0 0 . 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

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

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000010100010000000000000000000 from binary to decimal?

- **Which precision we must to use?** **single (32 bits)**

We convert to decimal the part to the left of the decimal point.

68 0.0

1 0 0 0 1 0 0 . 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

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

↔ ← 8 bits → ← 23 bits →

Sign Exponent Mantissa

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000010100010000000000000000000 from binary to decimal?

- **Which precision we must to use?** **single (32 bits)**

(-1) x 68 + 0.0 = -68.0

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

↔ ← 8 bits → ← 23 bits →

Sign **Exponent** **Mantissa**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000011101000101100000000000000 from binary to decimal?

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000011101000101100000000000000 from binary to decimal?

- **Which precision we must to use?**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000011101000101100000000000000 from binary to decimal?

- **Which precision we must to use?** single (32 bits)

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000011101000101100000000000000 from binary to decimal?

- **Which precision we must to use?** **single (32 bits)**

We split the number in its parts (sign, exponent and mantissa).

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

8 bits

23 bits

Sign **Exponent** **Mantissa**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000011101000101100000000000000 from binary to decimal?

- **Which precision we must to use?** **single (32 bits)**

+ → 0

- → 1

It is negative.

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

8 bits

23 bits

Sign **Exponent** **Mantissa**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000011101000101100000000000000 from binary to decimal?

- **Which precision we must to use?** single (32 bits)

We transform the exponent to decimal to find the normalization of the mantissa.

1 0 0 0 0 1 1 1 = 135 ➔ 135 – 127 = 8

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

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000011101000101100000000000000 from binary to decimal?

- **Which precision we must to use?** **single (32 bits)**

We transform the exponent to decimal to find the normalization of the mantissa.

1 0 0 0 0 1 1 1 = 135 ➔ 135 – 127 = 8

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

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000011101000101100000000000000 from binary to decimal?

- **Which precision we must to use?** single (32 bits)

We denormalize the mantissa adding a 1 on the left.

1.01000101100000000000000

1 10000111 01000101100000000000000

8 bits

23 bits

Sign **Exponent** **Mantissa**

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000011101000101100000000000000 from binary to decimal?

- **Which precision we must to use?** **single (32 bits)**

We move decimal point 8 places to the right.

1 0 1 0 0 0 1 0 1 . 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

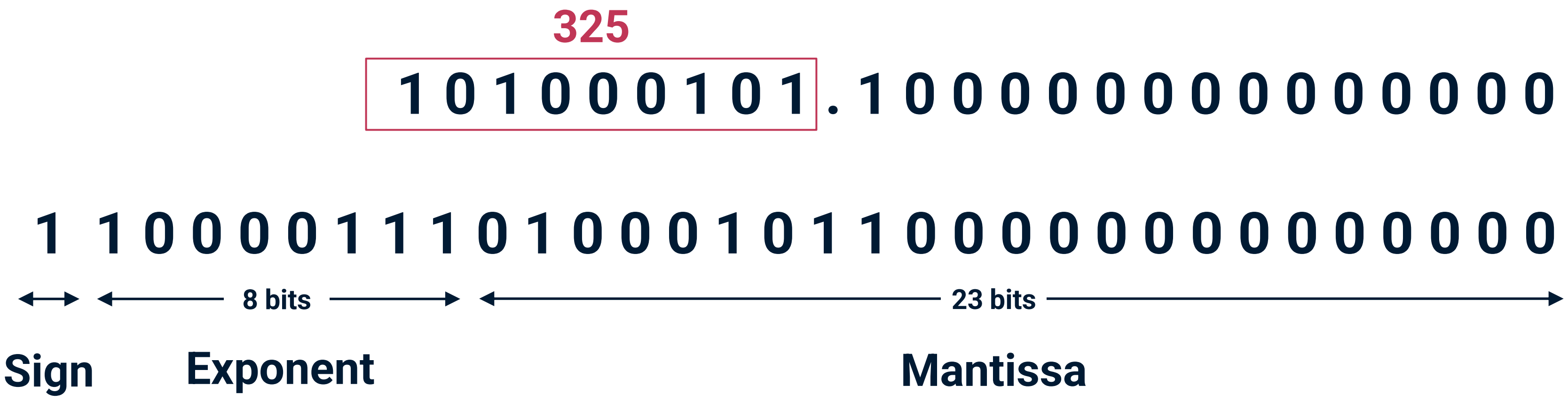
Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000011101000101100000000000000 from binary to decimal?

- **Which precision we must to use?** single (32 bits)

We convert to decimal the part to the left of the decimal point.



325

1 0 1 0 0 0 1 0 1 . 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

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

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000011101000101100000000000000 from binary to decimal?

- **Which precision we must to use?** single (32 bits)

We convert to decimal the part to the right of the decimal point.

$1 \times 2^{-1} + 0 \times 2^{-2} + \ldots + 1 \times 2^{-15} = 0.5$

$1\,0\,1\,0\,0\,0\,1\,0\,1\,.\boxed{1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0}$

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

↔ ← 8 bits → ← 23 bits →

Sign Exponent Mantissa

Floating-point numbers

Single IEEE 754 Floating Point Representation

How do you convert 11000011101000101100000000000000 from binary to decimal?

- **Which precision we must to use?** single (32 bits)

(-1) x 325 + 0.5 = -325.5

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

8 bits

23 bits

Sign

Exponent

Mantissa

Encoding representation

02

Encoding representation

An encoding refers to a collection of n-bit strings that follow a predefined convention, where each string represents either a numerical value or various types of information.

- Numeric encoding or code is used for representing numerical data.
- Alphanumeric encoding or code extends this representation to include numbers, letters, and punctuation marks.
- Error encoding or code is a technique used to modify information so that specific errors that may occur during the storage, retrieval, or transmission of data can be detected and corrected.

Encoding representation

Binary Coded Decimal

Binary Coded Decimal (BCD) is a method used for the binary representation of numbers in the decimal system. In BCD, each decimal digit (0-9) is represented by a combination of 4 bits.

$$734 = \boxed{0111}\,\boxed{0011}\,\boxed{0100}$$

$$22 = \boxed{0010}\,\boxed{0010}$$

Decimal	Binay (BCD) 8 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

Encoding representation

Alphanumeric Code

Alphanumeric codes are used for representing textual data, where each character is assigned a unique code in the form of a bit string. Typically, characters are categorized into five distinct groups:

- Alphanumeric characters: A, B, C, ….Z, a, b, c, …., z
- Numeric characters (Digits): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Special characters: ( ) + - , & > < Ñ ñ # Ç ç SP …
- Geometric and graphical characters: | ⌘¶
- Control characters: enter, space, …

Encoding representation

ASCII Code

The ASCII (American Standard Code for Information Interchange) code, established in 1968, is one of the earliest coding systems. Its primary purpose was to represent characters and symbols used in the English language. The basic ASCII code uses 7 bits to represent each character or symbol, allowing for 128 unique combinations.

C/Vega,7

C	/	V	e	g	a	,	7
1000011	0101111	1010110	1100101	1100111	1100001	0101100	0110111

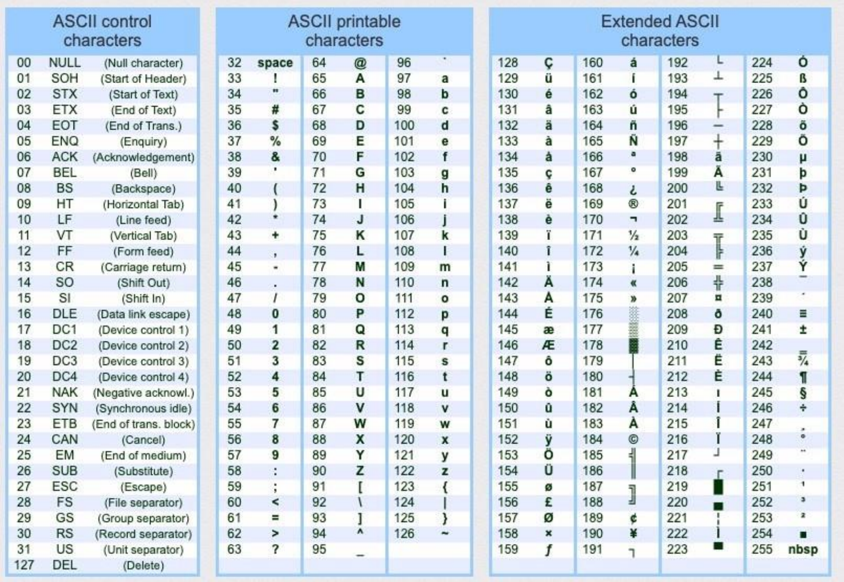
	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

It corresponds to the ANSI x 3.4 - 1968 or ISO 646 standardisation.

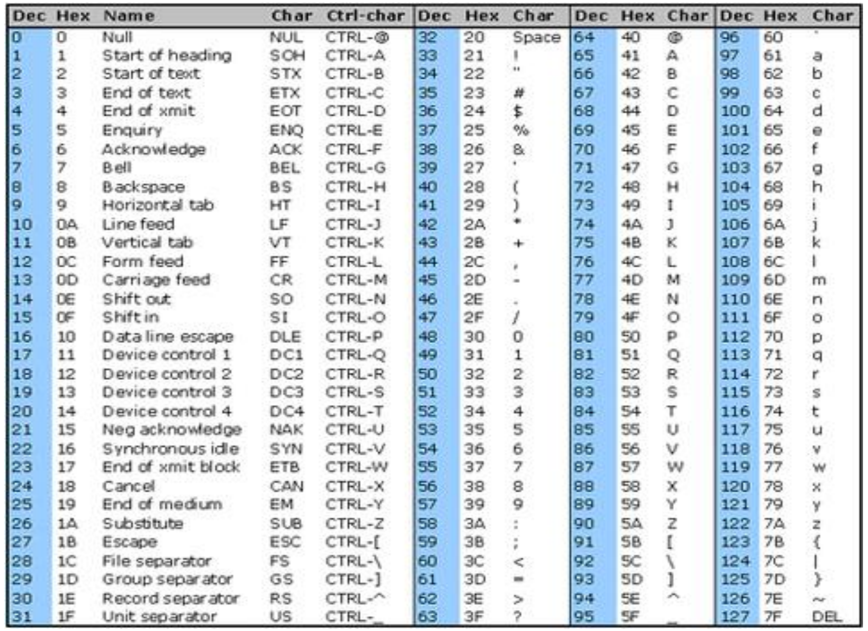
Encoding representation

Extended ASCII Code

The Extended ASCII employs an eight-bit character encoding scheme, retaining all the characters from the original seven-bit ASCII while also incorporating additional characters. This extension increases the total number of possible characters to 256.



ASCII control characters			ASCII printable characters						Extended ASCII characters							
00	NULL	(Null character)	32	space	64	@	96	`	128	Ç	160	á	192	└	224	Ó
01	SOH	(Start of Header)	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
02	STX	(Start of Text)	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
03	ETX	(End of Text)	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
04	EOT	(End of Trans.)	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
05	ENQ	(Enquiry)	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
06	ACK	(Acknowledgement)	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
07	BEL	(Bell)	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
08	BS	(Backspace)	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
09	HT	(Horizontal Tab)	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
10	LF	(Line feed)	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
11	VT	(Vertical Tab)	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
12	FF	(Form feed)	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
13	CR	(Carriage return)	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
14	SO	(Shift Out)	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
15	SI	(Shift In)	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
16	DLE	(Data link escape)	48	0	80	P	112	p	144	É	176	░	208	ð	240	≡
17	DC1	(Device control 1)	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
18	DC2	(Device control 2)	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
19	DC3	(Device control 3)	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
20	DC4	(Device control 4)	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
21	NAK	(Negative acknowl.)	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
22	SYN	(Synchronous idle)	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
23	ETB	(End of trans. block)	55	7	87	W	119	w	151	ù	183	À	215	Î	247	¸
24	CAN	(Cancel)	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
25	EM	(End of medium)	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	SUB	(Substitute)	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	ESC	(Escape)	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
28	FS	(File separator)	60	<	92	\	124	|	156	£	188	╝	220	▄	252	³
29	GS	(Group separator)	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
30	RS	(Record separator)	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
31	US	(Unit separator)	63	?	95	_			159	ƒ	191	┐	223	▀	255	nbsp
127	DEL	(Delete)														



Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

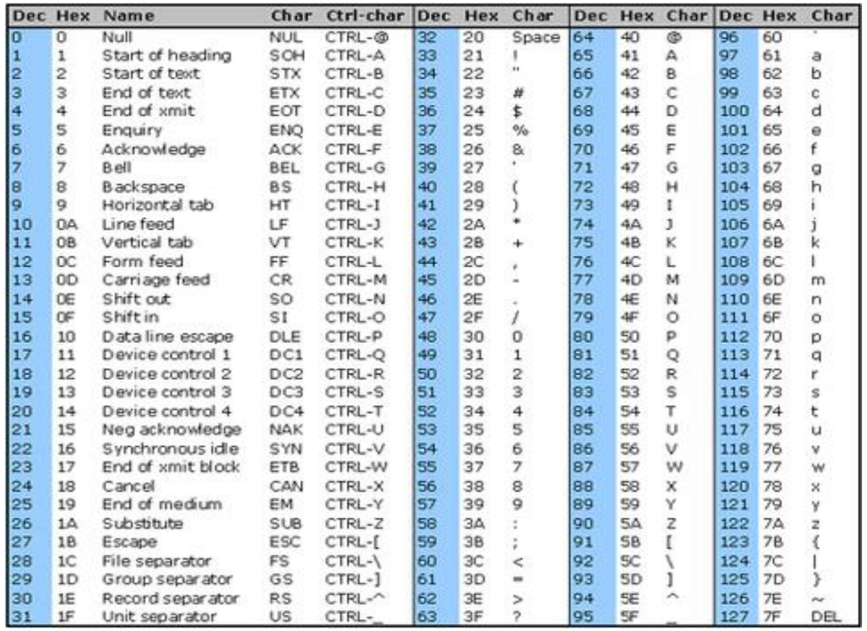
Encoding representation

Extended ASCII Code

H	e	l	l	o		W	o	r	l	d
01001000	01100101	01101100	01101100	01101111	00100000	01010111	01101111	01110010	01101100	01100100
48	65	6C	6C	6F	20	57	6F	72	6C	64

Converting characters to binary is simplified by first converting them to hexadecimal, and then transforming the hexadecimal values into binary.

This method is easier because each hexadecimal digit directly corresponds to a 4-bit binary sequence, making the conversion straightforward and reducing the chance of errors compared to converting directly from characters to binary.



Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Encoding representation

Extended ASCII Code

Several extended versions of the ASCII code exist, each employing 8 bits for character representation.

Name	ISO family	Geographical area
Latin-1	ISO 8859-1	Western and Eastern Europe
Latin-2	ISO 8859-2	Central and Eastern Europe
Latin-3	ISO 8859-3	Southern Europe, Maltese and Esperanto
Latin-4	ISO 8859-4	North europe
Latin/cyrillic	ISO 8859-5	Slavic languages
Latin/arabic	ISO 8859-6	Arabic languages
Latin/greek	ISO 8859-7	Modern greek

ISO/IEC 8859 is a joint ISO and IEC series of standards for 8-bit character encodings.

Encoding representation

Unicode Code

The Unicode standard is an information standard meticulously designed to ensure the consistent encoding, representation, and management of text across a wide array of the world's writing systems.

- Universality: It covers the vast majority of existing written languages.
- Uniqueness: Each symbol is assigned a unique code, ensuring that every character is distinctly represented.
- Uniformity: Each character is represented by a consistent number of bits, with modern versions of Unicode typically using 16 bits, although earlier versions and certain encoding forms can use 8, 16, or 32 bits, depending on the specific encoding scheme (e.g., UTF-8, UTF-16, UTF-32).

Encoding representation

Unicode Code

The codes in the Unicode standard (Basic Multilingual Plane) are categorized into four distinct groups or zones, which are typically outlined as follows:

Zone	Codes (HEX)	Symbols	Characters
A	0000 - 3FFF	Basic Latin (ASCII), Latin-1 and other Latin characters, Greek, Cyrillic, Armenian, Hebrew, Arabic, Syrian, Chinese, Japanese and Korean phonetic characters	8192
I	4000 - 9FFF	Chinese, Japanese and Korean ideograms	24576
O	A000 - DFFF	Not assigned	16384
R	E000 - FFFF	Local and user-specific characters	8192

Encoding representation

Extra information

- Base change: https://youtu.be/5WtLFbriEEE
- Integer numbers: https://youtu.be/B7SpmkW0lTs
- Two’s complement: https://youtu.be/UTVuROxztuQ
- Floating-point numbers https://youtu.be/HcjXH9WGmAU
- Some tips: https://youtu.be/5TlUWLxOWzU