Theory Exercises

Exerise 1

$$\boxed{123|_{10} = X|_2}$$

Solution:

$$\begin{array}{r|l} 123 & 1 \\ 61 & 1 \\ 30 & 0 \\ 15 & 1 \\ 7 & 1 \\ 3 & 1 \\ 1 & 1 \\ 0 & \\ & \uparrow \end{array}$$

$$\boxed{\mathbf{X = 1111011|_2}}$$

Correctness Check:

6	5	4	3	2	1	0
1	1	1	1	0	1	1

$$1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4 + 1 \cdot 2^5 + 1 \cdot 2^6 =$$
$$= 1 + 2 + 8 + 16 + 32 + 64 = 123|_{10}$$

Exercise 2

$$\boxed{011101|_2 = X|_{10}}$$

Solution:

5	4	3	2	1	0
0	1	1	1	0	1

$$X = 1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4 + 0 \cdot 2^5 = 1 + 4 + 8 + 16 = 29|_{10}$$

$$\boxed{\mathbf{X = 29|_{10}}}$$

Exercise 3

$$\boxed{23|_{10} = X|_5}$$

Solution:
The method is the same as for base 2. We iteratively divide by 5 and take note of the remainder. Then, we read the number from bottom to top.

$$\begin{array}{r|l} 23 & 3 \\ 4 & 4 \\ 0 & \\ & \uparrow \end{array}$$

$$\boxed{\mathbf{X = 43|_5}}$$

Correctness Check:

$$\begin{array}{cc} 1 & 0 \\ \hline 4 & 3 \end{array}$$

$$3 \cdot 5^0 + 4 \cdot 5^1 = 23|_{10}$$

Exercise 4

$$\boxed{123|_{10} = X|_H}$$

Solution:
Base "H" means *Hexadecimal*.

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

General Method:
Iteratively divide by 16 and take note of the remainder. Then, read the number from bottom to top.

$$\begin{array}{r|l} 123 & \text{B (11)} \\ 7 & 7 \\ 0 & \\ & \uparrow \end{array}$$

$$\boxed{\mathbf{X = 7B|_H}}$$

Alternative Method:
We know from Exercise 1 that $123|_{10} = 1111011|_2$. Because $16 = 2^4$, each hexadecimal digit corresponds exactly to 4 bits. Therefore, the bits can be grouped in groups of 4 (adding zeros to the left if necessary, since we are dealing with unsigned numbers):

$$\boxed{0}\ \ 1\ \ 1\ \ 1 \mid 1\ \ 0\ \ 1\ \ 1$$

At this point, each group of 4 bits can be converted independently:

- $0111|_2 = 7_H$
- $1011|_2 = B_H$

$$\boxed{\mathbf{X = 7B|_H}}$$

Note:
The conversion from hexadecimal to binary follows the same principle (in reverse). Each hexadecimal digit can be converted to 4 bits independently of the others.

- $7_H = 0111|_2$
- $B_H = 1011|_2$

This type of procedure can be followed only for bases that are powers of 2 (octal, hexadecimal, base 32, etc.).

Exercise 5

$$\boxed{A1|_H = X|_{10}}$$

Solution:

1	0
A	1

$$X = 1 \cdot 16^0 + A \cdot 16^1 = 1 \cdot 16^0 + 10 \cdot 16^1 = 1 + 16 \cdot 10$$

$$\boxed{\mathbf{X = 161|_{10}}}$$

Exercise 6

$$\boxed{91|_H = X|_8}$$

Solution:
Since both bases are powers of 2, it is convenient to pass through the binary representation.

Step 1: conversion from hexadecimal to binary, digit by digit:

- $9|_H = 1001|_2$
- $1|_H = 0001|_2$

Step 2: conversion from binary to octal.
In octal, each digit corresponds to 3 bits. Therefore, the bits must be grouped in groups of 3, adding a zero to the left. After that, the various groups can be converted independently.

		(9)			(1)		
0	**1**	**0**	**0**	**1**	0	0	0	1

- $010|_2 = 2|_8$
- $010|_2 = 2|_8$
- $001|_2 = 1|_8$

$$\boxed{\mathbf{91|_H = 221|_8}}$$

Exercise 7

$$\boxed{141|_4 = X|_{10}}$$

Solution:
Impossible. The digit 4 is not a valid digit in base 4.

Exercise 8

$$\boxed{-13|_{10} = X|_{S\&M-6bit}}$$

Solution:

Step 1: Convert the absolute value to pure binary:

13	1
6	0
3	1
1	1
0	
	↑

$$13|_{10} = 1101|_2$$

Step 2: Add zeros to the left of the module until you reach 5 bits (6 - 1).

$$13|_{10} = 01101|_2$$

Step 3: Add the sign bit to the left of the module (1 = negative).

$$\boxed{\mathbf{-13|_{10} = 101101|_{S\&M-6bit}}}$$

Exercise 9

$$\boxed{-32|_{10} = X|_{S\&M-6bit}}$$

Solution:
Impossible. To represent the module, 6 bits are needed, while only 5 are available:

$$32|_{10} = 100000|_2$$

You must remember that with the Sign and Magnitude format on 6 bits, only numbers in the range can be represented:

$$[-2^{6-1} + 1 : 2^{6-1} - 1] = [-31 : 31]$$

Exercise 10

$$\boxed{-13|_{10} = X|_{2C-6bit}}$$

Solution:

Step 1: Convert the absolute value to pure binary:

$$13|_{10} = 1101|_2$$

Step 2: Add zeros to the left of the module until you reach 6 bits.

$$13|_{10} = 001101|_2$$

Step 3: Invert 0 and 1, bit by bit.

$$001101 \rightarrow 110010$$

Step 4: Add +1.

$$\begin{array}{rl} 110010 & + \\ 000001 & = \\ \hline 110011 & \end{array}$$

$$\boxed{\mathbf{-13|_{10} = 110011|_{2C-6bit}}}$$

Alternative Method:

Steps 1 and 2: As in the previous exercise.

$$13|_{10} = 001101|_2$$

Step 3: Copy the bits from right to left, up to the first 1 (inclusive). After that, invert 0 and 1 in the remaining bits.

0	0	1	1	0	1
1	1	0	0	1	**1**

The first 1 from the right is highlighted in bold.

$$\boxed{\mathbf{-13|_{10} = 110011|_{2C-6bit}}}$$

Other Examples of the Alternative Method (1):

$$\boxed{-10|_{10} = X|_{2C-8bit}}$$

Steps 1 and 2:

$$10|_{10} = 00001010|_2$$

Step 3:

0	0	0	0	1	0	1	0
1	1	1	1	0	1	**1**	0

The first 1 from the right is highlighted in bold.

$$\boxed{\mathbf{-10|_{10} = 11110110|_{2C-8bit}}}$$

Other Examples of the Alternative Method (2):

$$\boxed{-24|_{10} = X|_{2C-8bit}}$$

Steps 1 and 2:

$$24|_{10} = 00011000|_2$$

Step 3:

0	0	0	1	1	0	0	0
1	1	1	0	**1**	0	0	0

The first 1 from the right is highlighted in bold.

$$\boxed{\mathbf{-24|_{H} = 11101000|_{2C-8bit}}}$$

Exercise 11

$$\boxed{1000001|_{2C} = X|_{10}}$$

Solution:

6	5	4	3	2	1	0
1	0	0	0	0	0	1

$$X = 1 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 0 \cdot 2^3 + 0 \cdot 2^4 + 0 \cdot 2^5 \ominus 1 \cdot 2^6 = 1 - 64$$

$$\boxed{\mathbf{X = -63|_{10}}}$$

Exercise 12

Add and subtract the following pairs of numbers in pure binary.

1. 10001000, 00001110
2. 01000110, 10001100
3. 10011000, 11101111

Solution:

1)

			1					carry
1	0	0	0	1	0	0	0	+
0	0	0	0	1	1	1	0	=
1	0	0	1	0	1	1	0	

Result: 10010110

1	0	0	0	1	0	0	0	-
-1	-1	-1	-1	-1	-1			borrow
0	0	0	0	1	1	1	0	=
0	1	1	1	1	0	1	0	

Result: 01111010

2)

			1	1				carry
0	1	0	0	0	1	1	0	+
1	0	0	0	1	1	0	0	=
1	1	0	1	0	0	1	0	

Result: 11010010

	0	1	0	0	0	1	1	0	-
-1		-1	-1	-1					borrow
	1	0	0	0	1	1	0	0	=
	0	0	1	1	1	0	1	0	

Result: The borrow on the most significant bit indicates that the result does not exist in pure binary (as it would be negative).

3)

1	1	1	1	1					carry
	1	0	0	1	1	0	0	0	+
	1	1	1	0	1	1	1	1	=
	1	0	0	0	0	1	1	1	

Result: The carry on the most significant bit indicates the presence of overflow in pure binary.

	1	0	0	1	1	0	0	0	-
-1	-1	-1		-1	-1	-1	-1		borrow
	1	1	1	0	1	1	1	1	=
	1	0	1	0	1	0	0	1	

Result: The borrow on the most significant bit indicates that the result does not exist in pure binary (as it would be negative).

Exercise 13

Perform the following additions between 6-bit two's complement numbers.

1. 100010 + 111011
2. 000001 + 111111

Solution:

1)

1				1			carry
	1	0	0	0	1	0	+
	1	1	1	0	1	1	=
	0	1	1	1	0	1	

Result: The most significant bit of the result is discordant with that of the two addends, so there is an overflow situation.

Note: In two's complement, the carry on the most significant bit is not considered in order to verify the correctness of the result.

2)

1	1	1	1	1	1		carry
	0	0	0	0	0	1	+
	1	1	1	1	1	1	=
	0	0	0	0	0	0	

Result: 000000. The result is correct since the two operands have discordant signs, a condition in which there can never be overflow in two's complement.

Exercise 14

Perform the following subtraction between 6-bit two's complement numbers: 100010 - 111011.

Solution:

To perform a subtraction between two numbers in two's complement, it is more convenient to change the sign of the subtrahend (i.e. calculate its two's complement), and then perform the sum: $a - b = a + (-b)$
The number 111011, when its sign is changed, i.e. complemented, becomes 000101 (see previous exercise), so we have:

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

Result: 100111.

Exercise 15

What is the range of numbers that can be represented in pure binary, sign and magnitude, and two's complement, using 6 bits?
Solution:

- Pure binary, 6 bits: $[0 : 63]$
- Sign and Magnitude, 6 bits: $[-31 : 31]$
- Two's Complement, 6 bits: $[-32 : 31]$

Exercise 16

We want to save a text of 100 pages, each composed of 50 lines of 69 characters, in an ASCII file. Each line is terminated by the LF (line feed) character and each page is terminated by the FF (form feed) character. Calculate the size of the file in kB.

Solution:
Each ASCII character requires 8 bits, i.e. 1 byte, whether it is a text or control character (e.g. LF and FF mentioned in the problem).
So, each line of the file occupies:

- 69 bytes for the text
- 1 byte for the line feed character

Each page contains 50 lines, each of which occupies the space calculated above. In addition, at the end of each page the FF character is written, which occupies an additional 1 byte. Since the file is composed of 100 pages, the total number of bytes occupied is:

$$100 \cdot (50 \cdot (69 + 1) + 1) = 350100B$$

One kB corresponds to 2^{10} B, so the solution is:

$$350100/(2^{10}) = 341.9kB$$

.

Exercise 17

In a mass storage with 1.44 MB of available space, the data of the students of a school must be written. Each student corresponds to a file containing the following data:

- surname (32 ASCII characters)
- name (32 ASCII characters)
- date of birth (DDMMYYYY encoded in ASCII)
- student ID (6-digit decimal number, integer and non-negative, encoded in pure binary)

Knowing that the data of each student should occupy an integer number of bytes, calculate the maximum number of students whose data can be written in the available space.

Solution:
An ASCII character is encoded using a byte. Therefore, the following number of bytes are needed:

- surname: 32 Byte
- name: 32 Byte
- date of birth: 8 Byte

For the student ID, we first calculate the number of **bits** needed to store it, from which we then obtain the corresponding Bytes.
We want to find the minimum number of bits that allows us to represent all possible decimal numbers of 6 digits, integers and non-negative. Therefore, all numbers from 0 to 999999. Using the pure binary representation, with N bits we can represent numbers in the interval:

$$[0 : 2^N - 1]$$

We therefore want to find the smallest N such that:

$$2^N - 1 \geq 999999 \rightarrow 2^N \geq 1 \cdot 10^6$$

The "greater than or equal" is necessary because the number of bits must be an integer.

Since the logarithm is a monotonic function, the inequality also holds after taking the base 2 logarithm of both sides.

$$N \geq \log_2(1 \cdot 10^6)$$

Since we want the smallest possible integer solution, this inequality can be transformed into an equality using the "ceiling" operator:

$$N = \lceil \log_2(1 \cdot 10^6) \rceil$$

$$N = \lceil 19.931 \rceil = 20$$

We therefore need 20 bits for each student ID. We know from the text of the problem that an integer number of bytes is needed for each student. We therefore calculate the number of bytes needed to store a student ID:

$$\left\lceil \frac{20}{8} \right\rceil = 3 \ Byte$$

In total, for each student we need:

$$32 + 32 + 8 + 3 = 75 \ Byte$$

The available space on the disk is:

$$1.44 \ MB = 1.44 \cdot 2^{20} \ Byte$$

So, the maximum number of students whose data can be stored on the disk is obtained as:

$$\left\lfloor \frac{1.44 \cdot 2^{20}}{75} \right\rfloor = \lfloor 20132.65 \rfloor$$

$$\boxed{\textbf{N. of Students} = \textbf{20132}}$$

In this case, the "floor" operator must be used because it is not possible to store the data of a student if only a fraction of the necessary space is available.

Exercise 18

A processing system has a central memory of 2^{30} cells of 16 bits, with an access time of 50 ns. This memory is used by a microprocessor (μP) whose ALU can perform 20 million operations per second. Estimating that for reading the data on which the operations are carried out, on average, one main memory access is required every two operations, evaluate whether it is necessary to provide a cache memory, to obtain the maximum performance from the system.‘

Solution:
Memory accesses required:

- $\frac{20 \cdot 10^6}{2} = 10 \cdot 10^6 Accesses/s$ to read and write the data necessary for the program (one every two operations, as indicated in the text)
- $20 \cdot 10^6 Accesses/s$ to fetch the program instructions. **Each instruction must be read from memory before it can be executed**.

Total:

$$30 \cdot 10^6 Accesses/s$$

Number of accesses per second that can be made in the main memory:

$$\frac{1s}{50ns} = \frac{1}{50 \cdot 10^{-9}} = 20 \cdot 10^6 Accesses/s$$

So the cache memory is necessary. Otherwise, the μP will not be able to read from memory all the data necessary to keep the ALU always active, and therefore to maximize performance.

Note:
The data on the number of cells and the number of bits per cell are not necessary for the solution of the problem.

Exercise 19

Calculate the number of memory bits in a bank with an 8-wire Address Bus and a 16-wire Data Bus.

Solution:

With an 8-wire Address Bus, $2^8 = 256$ memory cells can be addressed.
Each memory cell contains a number of bits equal to the number of wires in the Data Bus, i.e. 16.
Therefore, the total number of bits in memory is:

$$2^8 \cdot 16 = 256 \cdot 16 = 4096\ bit = 4\ Kbit$$

Exercise 20

A memory bank of 512 Kbit is equipped with a 16-wire Address Bus. How many Data Bus wires are needed to address all the available memory, i.e. how long is a memory word?

Solution:

The memory is made up of:

$$512 \cdot 1024 = 524288 \; bit$$

With a 16-wire Address Bus, $2^{16} = 65536$ cells can be addressed. Therefore, in order to address all the available memory, each word must have a length of:

$$\frac{524288}{65536} = 8 \; bit$$

Exercise 21

Expain which system buses are involved in the fetch phase of a machine instruction, and what information is transmitted during this phase.

Solution:

In the fetch phase, the content of the memory cell whose address is contained in the Program Counter (PC) must be transferred to the Instruction Register (IR). Therefore, all system buses are involved, in the following way:

1. The CPU requests the memory cell pointed to by the PC register, i.e.:
   - 1.A The CPU writes the value contained in the PC to the Address Bus (ABUS)
   - 1.B The CPU writes the code corresponding to the read command on the Control Bus (CBUS)
2. The memory provides the CPU with the content of the requested cell, i.e.:
   - 2.A The memory writes the content of the cell corresponding to the address present on the ABUS to the Data Bus (DBUS).

Graphically:

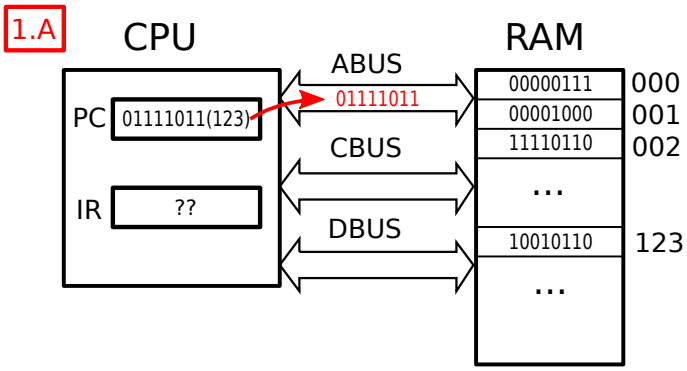

Fig. 1: Phase 1.A: The CPU writes the value contained in the PC to the Address Bus (ABUS).

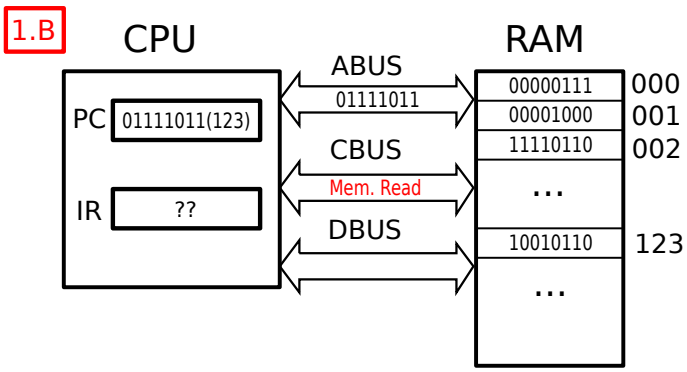

Fig. 2: Phase 1.B: The CPU writes the code corresponding to the read command on the Control Bus (CBUS). **Note:** The command will obviously be encoded as 0 and 1 on the CBUS wires, but this encoding is not relevant for the exercise.

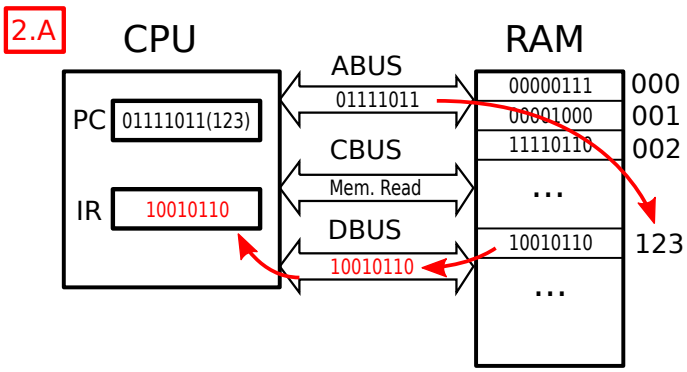

Fig. 3: Phase 2.A: The memory writes the content of the cell corresponding to the address present on the ABUS to the Data Bus (DBUS). This content will then be saved by the CPU in the IR.