Fundamentals of Digital Systems Design

Course website:
http://www.liacs.leidenuniv.nl/~stefanovtp/courses/DITE/index.html

Main Lecturer: Todor Stefanov (**t.p.stefanov@liacs.leidenuniv.nl**)

Assistants: 8 BSc/MSc/PhD students (dite.liacs@gmail.com)

Leiden Embedded Research Center (LERC)
Leiden Institute of Advanced Computer Science (LIACS)
Leiden University, The Netherlands

Course Organization: Structure and Rules

- **On-campus** Lectures
  - your attendance is strongly recommended
- **On-campus** Hands-on tutorials with practical assignments
  - your attendance is strongly encouraged
- **On-campus and At-home** Design Project
  - you must make and turn-in the design project
- Homeworks
  - doing homeworks is strongly recommended
  - excellent material to practice for the exams!

Important:

- All info needed for the course will be on the course website
- Check the website regularly for instructions
- Check regularly Brightspace and your email for announcements

Course Organization: On-campus Lectures

- Goal
  - <u>To learn the fundamentals of digital systems design</u>
  - Theoretical Foundations
  - Basic Digital Circuits and Building Blocks
  - Design Algorithms and Procedures
  - The basics of Memory Design
  - The basics of Micro-Processor Design
- Format
  - **On-campus presentations** supported by PowerPoint slides
  - Slides available after each lecture on the course website
- Literature
  - M. Morris Mano and Charles R. Kime,
    "Logic and Computer Design Fundamentals", 3rd or 4th edition

<u>Important:</u>

- <u>Dedicated 2 hours for questions after each lecture on Mondays!</u>

Lectures Schedule and Location

Date	Time	Location
02 Sept (Monday)	15:15 - 17:00	Lecture Hall - C3
04 Sept (Wednesday)	15:15 - 17:00	Lecture Hall - C3
09 Sept (Monday)	15:15 - 17:00	Lecture Hall - C3
11 Sept (Wednesday)	15:15 - 17:00	Lecture Hall - C3
16 Sept (Monday)	15:15 - 17:00	Lecture Hall - C3
18 Sept (Wednesday)	15:15 - 17:00	Lecture Hall - C3
23 Sept (Monday)	15:15 - 17:00	Lecture Hall - C3
30 Sept (Monday)	15:15 - 17:00	Lecture Hall - C3
07 Oct (Monday)	15:15 - 17:00	Lecture Hall - C3
28 Oct (Monday)	15:15 - 17:00	Lecture Hall - C3
04 Nov (Monday)	15:15 - 17:00	Lecture Hall - C3
11 Nov (Monday)	15:15 - 17:00	Lecture Hall - C3
18 Nov (Monday)	15:15 - 17:00	Lecture Hall - C3
25 Nov (Monday)	15:15 - 17:00	Lecture Hall - C3
02 Dec (Monday)	15:15 - 17:00	Lecture Hall - C3

Course Organization: On-campus Tutorials

- Goal
  - To learn how to solve simple digital design problems in practice
  - To learn how to use software tools to design digital circuits
- Format
  - Pen-and-paper practical assignments (simple digital circuit designs)
  - Design and simulation of digital circuits using software tools
- Tools
  - Xilinx ISE for digital circuit design and simulation
  - <u>Go to the website of the course to see options to access Xilinx ISE!</u>
- Timetable and Location

Period	Day	Time	Place
02 Oct – 06 Nov	Every Wednesday	15:15 - 17:00	Rooms DM.0.09-PC to DM.0.21-PC in Gorlaeus

Course Organization: Design Project

- Goal
  - To apply the knowledge gained at lectures and hands-on tutorials
  - To design a simple 4-bit Micro-Processor
- Format
  - Project specification will be given in beginning of November
  - Project tutorials with the tutors on Wednesdays
    - <u>to work on the project on-campus</u>
    - to get advice on how to design the micro-processor
    - to ask questions
  - <u>You have to spend extra time at home or in our computer labs if</u>
    - project tutorials time is not sufficient to accomplish the project
- Timetable and Location

Period	Day	Time	Place
13 Nov - 11 Dec	Every Wednesday	15:15 - 17:00	Rooms DM.0.09-PC to DM.0.21-PC in Gorlaeus

Course Organization: Homeworks

- Goals
  - To understand better the material given at lectures
  - To study alone additional material not given at lectures
  - To prepare and practice for exams!
- Format
  - Several homeworks will be given
  - Homework instructions posted on website every Wednesday
  - Try to do homework alone
    - within 7 days after the instructions are posted
  - Answers to homeworks will be posted on the course website
    - 7 days after the instructions

<u>Important:</u>

- <u>Dedicated 2 hours for questions after each lecture on Mondays!</u>

Course Organization: Exams and Grading

- Exams
  - One *Midterm exam* and one *Final exam* at the end of the course
  - You must be present at these exams
  - In case you fail at the final exam, there will be one Re-take exam
- Timetable and Location

Exams	Date	Hour	Place
Midterm Exam	October 23, 2024	13:15 – 16:15	Lecture Hall - C1&C2
Final Exam	January 14, 2025	13:15 – 16:15	Lecture Hall - C3 Gorlaeus - BE.0.17&0.18

- Grading
  - You will receive 3 Grades: one for Final exam (**G^{Fexam}**), one for Project (**$G^{Project}$**), and one for Midterm exam (**G^{Mexam}**),
  - All grades are important and will form your ***Final Grade*** as follows:

If ($G^{Fexam} \geq$ **6.0** and $G^{Project} \geq$ **6.0**) Then

Final Grade = $0.4 \cdot G^{Fexam} + 0.4 \cdot G^{Project} + 0.2 \cdot G^{Mexam}$

Else

Final Grade ≤ 5

Digital Systems and Information
Part I

Overview

- Introduction to Digital Systems
  - Digital Systems (examples and general remarks)
  - Basic Digital System Structure (basic components and their function)
- Number Systems
  - Positional Number Systems (decimal, binary, octal, and hexadecimal)
  - Number Conversions (r-to-decimal, decimal-to-r, other conversions)
- Representations of Numbers in Digital Systems
  - Integer Numbers (unsigned and signed representations)
  - Real Numbers (floating-point representation)

Digital Systems: Example

- Modern Digital Personal Computers (PC)
  - The best known example of a digital system
  - Most striking properties are: generality and flexibility

Digital Systems: More Examples



Digital Systems: General Remarks

- Digital Systems manipulate discrete elements/quantities of information
- Discrete quantities of information emerge from:
  - the nature of the data being processed
  - the data may be discretized from continues values
- Early computer systems used mainly for numeric computations
  - the discrete elements used were the digits
  - hence the term digital computer/system emerged
- In general, any system uses an alphabet (set of symbols) to represent information
  - The English language system uses an alphabet of 26 symbols (letters)
  - The decimal number system uses an alphabet of 10 symbols (digits)
- What about the alphabet of the Digital Systems?

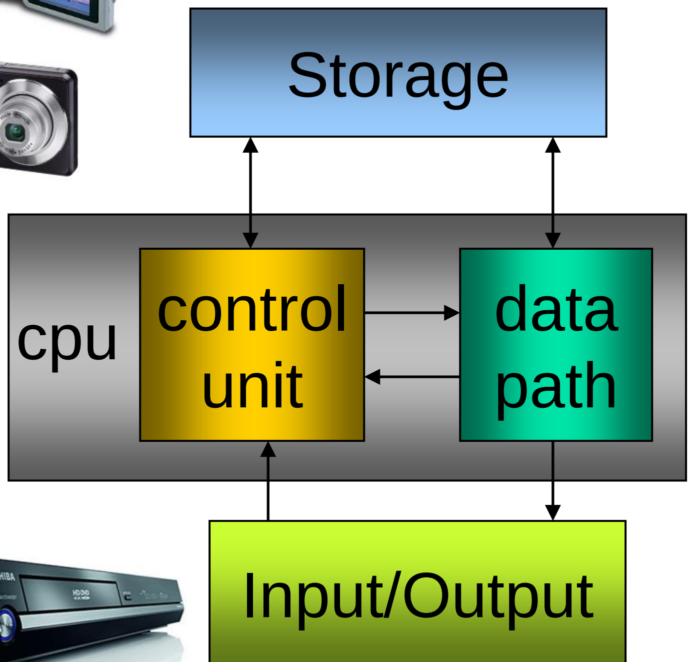
The Digital Systems' Alphabet is Binary

- Digital Systems use only one alphabet with two symbols (digits) '0' and '1' (hence binary alphabet)
  - Binary digit is called a bit
  - Information is represented by groups of bits
- Why is a binary alphabet used?
  - Digital systems have a basic building block called **switch**
  - The switch can only be **"on"** or **"off"**
  - That is, two discrete values **'0'** and **'1'** can be distinguished
- Electric device, transistor, physically implements the switch
  - **'0'** and **'1'** are physically represented by voltage values called **LOW** and **HIGH**
    - "on" (closed) switch corresponds to **'0'** and is represented by **LOW** voltage value (between 0.0 and 1.0 Volt)
    - "off" (open) switch corresponds to **'1'** and is represented by **HIGH** voltage value (between 3.0 and 5.0 Volts)
  - More information will be given later in another lecture

Basic Digital System Structure









- **CPU:** Central Processing Unit
- **Data Path:** arithmetic and logic operations
- **Control Unit:** make sure that the sequence of data path operations is correct
- **Storage:** no memory = no system
- **Input/Output:** allow the system to interact with the outside world

Information Representation

- All information in Digital Systems is represented in binary form
- Information that is not binary is converted to binary before processed by Digital Systems
- Decimal numbers are expressed
  - in the binary number system or
  - by means of a binary code
- That is not too difficult because
  - All number systems have a similar formal representation
  - Thus, one number system can be converted into another

- Let us look into number systems and conversions

Number Systems

Number Systems are used in arithmetic to represent numbers by strings of digits. There are two types of systems:

- Positional number systems
  - The meaning of each digit depends on its position in the number
  - Example: Decimal number system
    - **585.5** is a decimal number in positional code
    - "**5** hundreds plus **8** tens plus **5** units plus **5** tenths"
    - The hundreds, tens, units, and tenths are powers of 10 implied by the position of the digits

$$\mathbf{585.5} = \mathbf{5}x10^{2} + \mathbf{8}x10^{1} + \mathbf{5}x10^{0} + \mathbf{5}x10^{-1}$$

    - Decimal number system is said to be of *base* or *radix* 10 because
      - it uses 10 distinct digits (0 – 9)
      - the digits are multiplied by powers of 10
- Non-positional number systems
    - Old Roman numbers: for example, ***XIX*** equals to 19

Positional Number Systems

We can represent numbers in any number system with base r

- Number in positional code
  - $(A_{n-1}A_{n-2}\ldots A_1A_0\mathbf{.}A_{-1}A_{-2}\ldots A_{-m+1}A_{-m})_r$
  - r is the base (radix) of the system, $r \in \{2, 3, \ldots, \text{I}\}$
  - every digit $A_i \in \{0, 1, 2, \ldots, r\text{-}1\}$, where $\{0, 1, 2, \ldots, r\text{-}1\}$ is the digit set
  - "**.**" is called the radix point
  - A_{n-1} is referred to as the most significant digit
  - A_{-m} is referred to as the least significant digit
- Number in base r expressed as power series of r
  - $A_{n-1}\,r^{\,n-1} + A_{n-2}\,r^{\,n-2} + \ldots + A_1\,r^{\,1} + A_0\,r^{\,0} + A_{-1}\,r^{\,-1} + A_{-2}\,r^{\,-2} + \ldots + A_{-m+1}\,r^{\,-m+1} + A_{-m}\,r^{\,-m}$
- Example: a number in number system with base 5
  - $(132.4)_5 = 1x5^2 + 3x5^1 + 2x5^0 + 4x5^{-1} = 25 + 15 + 2 + 0.8 = (42.8)_{10}$

Binary Number System

This is the system used for arithmetic in all digital computers

- Number in positional code
  - $(b_{n-1}b_{n-2}\ldots b_1b_0\mathbf{.}b_{-1}b_{-2}\ldots b_{-m+1}b_{-m})_r$
  - $r = 2$ is the base of the binary system
  - every digit $b_i \in \{0, 1\}$
  - the digits b_i in a binary number are called bits
  - b_{n-1} is referred to as the most significant bit (MSB)
  - b_{-m} is referred to as the least significant bit (LSB)
- Number in base *2* expressed as power series of *2*
  - $b_{n-1}\,2^{\,n-1} + b_{n-2}\,2^{\,n-2} + \ldots + b_1\,2^{\,1} + b_0\,2^{\,0} + b_{-1}\,2^{\,-1} + b_{-2}\,2^{\,-2} + \ldots + b_{-m+1}\,2^{\,-m+1} + b_{-m}\,2^{\,-m}$
- Example: a number in the binary number system
  - $(1011.01)_2 = 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 + 0x2^{-1} + 1x2^{-2} = 8 + 2 + 1 + 0.25 = (11.25)_{10}$

Burn this table into your memory

Power of Two

n	2^n	n	2^n	n	2^n
0	1	5	32	10	1024
1	2	6	64	11	2048
2	4	7	128	12	4096
3	8	8	256	13	8192
4	16	9	512	14	16384

$2^{10} = 1024$ **is 1K (kilo)**

2^{20} **is 1M (mega)**

2^{30} **is 1G (giga)**

Other Useful Number Systems

Octal (base-8) and Hexadecimal (base-16) number systems

-- useful for representing binary quantities indirectly because

-- their bases are powers of two

-- have more compact representations of binary quantities

- Octal number system
  - $(o_{n-1}o_{n-2}\ldots o_1o_0.o_{-1}o_{-2}\ldots o_{-m+1}o_{-m})_8$
  - every digit $o_i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$.
  - $o_{n-1}\,8^{\,n-1} + o_{n-2}\,8^{\,n-2} + \ldots + o_1\,8^{\,1} + o_0\,8^{\,0} + o_{-1}\,8^{\,-1} + o_{-2}\,8^{\,-2} + \ldots + o_{-m+1}\,8^{\,-m+1} + o_{-m}\,8^{\,-m}$
  - $(127.4)_8 = 1x8^2 + 2x8^1 + 7x8^0 + 4x8^{-1} = (87.5)_{10} = (001\ 010\ 111.100)_2$
- Hexadecimal number system
  - $(h_{n-1}h_{n-2}\ldots h_1h_0.h_{-1}h_{-2}\ldots h_{-m+1}h_{-m})_{16}$
  - every digit $h_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$.
  - $h_{n-1}\,16^{\,n-1} + h_{n-2}\,16^{\,n-2} + \ldots + h_1\,16^{\,1} + h_0\,16^{\,0} + h_{-1}\,16^{\,-1} + h_{-2}\,16^{\,-2} + \ldots + h_{-m+1}\,16^{\,-m+1} + h_{-m}\,16^{\,-m}$
  - $(B6F.4)_{16} = 11x16^2 + 6x16^1 + 15x16^0 + 4x16^{-1} = (2927.25)_{10} = (1011\ 0110\ 1111.0100)_2$

Another Important Table

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hex (base 16)
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Conversion from base *r* to Decimal

To convert a number in base ***r*** to decimal number (base 10) do:

-- expand the number in power series of ***r*** and

-- add all the terms as shown below

$(A_{n-1}A_{n-2}\ldots A_1A_0.A_{-1}A_{-2}\ldots A_{-m+1}A_{-m})_r =$

$A_{n-1}r^{n-1} + A_{n-2}r^{n-2} + \ldots + A_1 r^1 + A_0 r^0 + A_{-1} r^{-1} + A_{-2} r^{-2} + \ldots + A_{-m+1} r^{-m+1} + A_{-m} r^{-m}$

- Example of converting Binary (base 2) to Decimal (base 10):

$\mathbf{(1011.01)_2} = 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 + 0x2^{-1} + 1x2^{-2} = 8 + 2 + 1 + 0.25 = \mathbf{(11.25)_{10}}$

- Example of converting number in base 5 to Decimal (base 10):

$\mathbf{(132.4)_5} = 1x5^2 + 3x5^1 + 2x5^0 + 4x5^{-1} = 25 + 15 + 2 + 0.8 = \mathbf{(42.8)_{10}}$

- Example of converting Octal (base 8) to Decimal (base 10):

$\mathbf{(127.4)_8} = 1x8^2 + 2x8^1 + 7x8^0 + 4x8^{-1} = \mathbf{(87.5)_{10}}$

- Example of converting Hexadecimal (base 16) to Decimal (base 10):

$\mathbf{(B6F.4)_{16}} = 11x16^2 + 6x16^1 + 15x16^0 + 4x16^{-1} = \mathbf{(2927.25)_{10}}$

Conversion from Decimal to base *r*

The conversion is done as follows:

1) Separate the number into integer part and fraction part
   -- the two parts must be converted differently
2) Converting the integer part
   -- divide the integer part and all successive quotients by *r* and
   -- accumulate the remainders
3) Converting the fraction part
   -- multiply the fractional parts by r and accumulate integers

- Example of converting Decimal (base 10) to Binary (base 2): **$(41.6875)_{10}$**

Converting the integer part $(41)_{10}$:

41/2 = 20 + 1/2	Remainder = 1	LSB
20/2 = 10 + 0/2	0	
10/2 = 5 + 0/2	0	
5/2 = 2 + 1/2	1	
2/2 = 1 + 0/2	0	
1/2 = 0 + 1/2	1	MSB

$(41)_{10} = (101001)_2$

Converting the fraction part $(0.6875)_{10}$:

0.6875 x 2 = 1.3750	Integer = 1	MSB
0.3750 x 2 = 0.7500	0	
0.7500 x 2 = 1.5000	1	
0.5000 x 2 = 1.0000	1	LSB

$(0.6875)_{10} = (0.1011)_2$

$(41.6875)_{10} = (101001.1011)_2$

Other Conversions

- Binary to Octal or Hexadecimal: <u>grouping bits starting from the radix point</u>
  - $\mathbf{(1101010.01)_2}$ to Octal (groups of 3): $(001|101|010.010|)_2$ = $\mathbf{(152.2)_8}$
  - $\mathbf{(1101010.01)_2}$ to Hex (groups of 4): $(0110|1010.0100|)_2$ = $\mathbf{(6A.4)_{16}}$
- Octal to Binary: <u>convert each digit to binary using 3 bits</u>
  - $\mathbf{(475.2)_8}$ = $\mathbf{(100\ 111\ 101.\ 010)_2}$
- Hexadecimal to Binary: <u>convert each digit to binary using 4 bits</u>
  - $\mathbf{(7A5F.C)_{16}}$ = $(0111\ 1010\ 0101\ 1111.\ 1100)_2$ = $\mathbf{(111101001011111.11)_2}$
- Hexadecimal to Octal
  - Hexadecimal → Binary → Octal
- Octal to Hexadecimal
  - Octal → Binary →Hexadecimal

Range of Numbers in base r

- Assume numbers represented by *n* digits for the integer part and *m digits* for the fraction part -- $(A_{n-1}A_{n-2}\ldots A_1A_0.A_{-1}A_{-2}\ldots A_{-m+1}A_{-m})_r$
- What is the smallest and the largest number that can be represented?
- The smallest integer number is **0** and the larges is $(r-1)\,r^{n-1} + (r-1)\,r^{n-2} + \ldots + (r-1)r^{1} + (r-1)\,r^{0} = \mathbf{r^n-1}$,i.e., the range is from **0** to $\boldsymbol{r^n-1}$
- The smallest fraction number is **0.0** and the largest is $(r-1)\,r^{-1} + (r-1)\,r^{-2} + \ldots + (r-1)\,r^{-m+1} + (r-1)\,r^{-m} = \mathbf{1- r^{-m}}$,i.e., the range is from **0.0** to $\boldsymbol{1- r^{-m}}$
- The range of numbers is from **0.0** to $\boldsymbol{r^n - r^{-m}}$
  - The range of numbers in base *r* depends on the number of digits used to represent the numbers
- Examples:
  - Largest 3-digit integer decimal (base 10) number is 10^3-1 = 1000 - 1 = 999
  - Largest 8-digit integer binary (base 2) number is $(11111111)_2$,i.e., 2^8-1 = 255
  - Largest 5-digit decimal (base 10) fraction is 1-10^{-5} = 1 – 0.00001 = 0.99999
  - Largest 16-digit binary (base 2) fraction is 1-2^{-16} = 0.9999847412
- What about the range of negative numbers?

Representations of Numbers in Digital Systems (1)

- Numbers are represented in binary format as strings of bits
  - Bit is the smallest binary quantity with a value of 0 or 1
  - Byte is a string (sequence) of 8 bits
  - Word is a string (sequence) of n bits (n > 8)
  - Examples → bit: 1 byte: 01101111 16-bit word: 11110100 10001010

- Positive Integer Numbers
  - Can be represented as **Unsigned Binary Numbers** using a string of **n** bits
  - Magnitude representation – number **X** in binary having **n** bits
    - Example: **00001001** (represents integer number 9 using 8 bits)
  - *2's Complement representation* – Given number **X** in binary having **n** bits, the 2's complement of **X** is defined as $2^{\mathbf{n}} - \mathbf{X}$.
    - Example: **11110111** (2's complement of integer number 9)
  - *1's Complement representation* - Given number **X** in binary having **n** bits, the 1's complement of **X** is defined as $(2^{\mathbf{n}} - 1) - \mathbf{X}$.
    - Example: **11110110** (1's complement of integer number 9).

Representations of Numbers in Digital Systems (2)

- Positive and Negative Integer Numbers
  - Can be represented as **Signed Binary Numbers** using a string of **n** bits
    - The most significant bit is interpreted as a sign bit
    - For positive numbers the sign bit is 0
    - For negative numbers the sign bit is 1

  - Signed-Magnitude representation
    - Example: **0|0001001** (represents integer number +9 using 8 bits)
    - Example: **1|0001001** (represents integer number -9 using 8 bits)
  - Signed-*2's Complement representation*
    - Example: **0|0001001** (signed-2's complement of integer number +9)
    - Example: **1|1110111** (signed-2's complement of integer number -9)
  - Signed-*1's Complement representation*
    - Example: **0|0001001** (signed-1's complement of integer number +9)
    - Example: **1|1110110** (signed-1's complement of integer number -9)

- How do we get the signed complements?

Representations of Numbers in Digital Systems (3)

- Real Numbers
  - Can be represented as fixed-point or floating-point numbers
- We will look closer to floating-point numbers
  - Floating-point representation is similar to scientific notation
  - It consists of two parts – mantissa M and exponent E
  - Floating-point number F represented by the pair (M,E) has the value

$$F = M x \beta^{E} \quad (\beta \text{ - base of exponent})$$

- Example: the decimal number **-179.75** as a floating-point number is **$- 0.17975 \times 10^{+3}$**
  - Mantissa **M** = -0.17975 – represents the number as a fraction
  - Exponent **E** = +3 – designates the position of the radix point
  - Base **β** = 10 – the number is decimal
- Floating-point representation is not unique
  - $-179.75 = -0.17975 \times 10^{+3} = -0.0017975 \times 10^{+5}$.
  - $-0.17975 \times 10^{+3}$ is called normalized form because the most significant digit of the fraction is nonzero.

Representations of Numbers in Digital Systems (4)

- IEEE standard single-precision floating-point representation
  - Using 32-bit word with the following format:

1	8	23
S	Exponent E	Unsigned Significand M

  - The value of the number is: $(-1)^S \times (1.M)_2 \times 2^{E-127}$
  - The larges and smallest positive numbers are:
    - $(-1)^0 \times (1.\mathbf{11111111111111111111111})_2 \times 2^{\mathbf{254}-127} = +(1+1-2^{-23})\times 2^{127}$
    - $(-1)^0 \times (1.\mathbf{00000000000000000000000})_2 \times 2^{\mathbf{1}-127} = +\ 2^{-126}$
  - Special cases:
    - E = 255 and M = 0, the number represents plus or minus infinity.
    - E = 255 and M ≠ 0, not a valid number (signify invalid operations).
    - E = 0 and M = 0, the number denotes plus or minus zero.
    - E = 0 and M ≠ 0, the number is said to be de-normalized.
- Example: $(\mathbf{-179.75})_{10} = (-10110011.11)_2 = (-1.\mathbf{011001111} \times 2^{\mathbf{7}})_2$

1	**10000110**	**011001111** 00000000000000