Week 01: Assignment 1.

COMPUTING BASISES (NUMBER SYSTEMS, BINARY ARITHMETIC, BOOLEAN ALGEBRA)

1. PURPOSE OF WORK

1.1. Get experience with the conversion of numbers between different number systems used in electronics.

1.2. Learn how computers use various forms of the binary system to perform calculations.

1.3. In this work, it is important that you think about how number systems work, so HOW the answer is more important than the answer itself. Don’t use calculator - you need to learn the method, not just the answer!

2. TASKS FOR WORK

2.1. Read three topics with the theoretical minimum about Number Systems, Binary Arithmetic, Boolean Algebra.

2.2. Make self-training.

- Complete some number converting tasks on the Learn Binary Practice website. (https://subnettingpractice.com/binary.html).
- Complete some number Boolean Algebra Tasks on the website in Course Repository: https://github.com/Arailym-ray/AITU-Operating-system-concepts/blob/e36152455d767a90b611b3ce375aa58e28cd2aa4/week%201/Boolean%20Algebra%20Tasks%20Generator.html

2.3. For every Tasks select Your variant Nr generated from Your Name Surname.
Make Task, don’t use calculator - you need to learn the method, not just the answer!

3. REPORT

Fill in Table with Detailed Answers and send Report to send to Moodle. Download the Report Blank Form from the Moodle or GitHub.

3.1. GRADUATION.

Grade =correctly formed initial numbers and correctly made of all 5 tasks variants.

VARIANT OF TASKS AND EXAMPLES.

Nr	Assignment, Instruction, Variant of Task	Detailed Answer Example
3.1	Convert Decimal integer to a) Binary, b) Octal, c) Hexadecimal, d) Check your answer by converting Bin→Dec • Choose your variant x = 1st letter of your Name in the English alphabet. • Use your date of birth to number generation from date template DdMmYYyy. • In [Square Brackets] Your see examples for 23 Apr 1987 Year= 23041987= DdMmYYyy. • Example for **Y**URIY LI, born 23/04/1987, task variant = “Y” with template yydD are selected and number = 8732_{10} are generated. A) Your date of birth in DdMm format [example for 23/04/1987, 2304]; B) Your date of birth in MmDd format [example for 23/04/1987, 0423]; C) Your year of birth in YYyy format [example for 23/04/1987, 1987]; D) Your year of birth in YyyD format [example for 23/04/1987, 9870]; E) Your year of birth in YYDd format [example for 23/04/1987, 1923]; F) Your year of birth in YYyD format [example for 23/04/1987, 9872]; G) Your date of birth in YYMm format [example for 23/04/1987, 1904]; H) Your date of birth in Mmyy format [example for 23/04/1987, 0487]; I) Your date of birth in Ddyy format [example for 23/04/1987, 2387]; J) Your date of birth in DdYy format [example for 23/04/1987, 2398]; K) Your date of birth in DdYY format [example for 23/04/1987, 2319]; L) Your date of birth in dDyy format [example for 23/04/1987, 3287]; M) Your date of birth in dDYy format [example for 23/04/1987, 3298]; N) Your date of birth in dDYY format [example for 23/04/1987, 3219]; O) Your date of birth in mMDd format [example for 23/04/1987, 4023]; P) Your date of birth in mMdD format [example for 23/04/1987, 4032]; Q) Your date of birth in mMYY format [example for 23/04/1987, 4019]; R) Your date of birth in mMYy format [example for 23/04/1987, 4098]; S) Your date of birth in mMyy format [example for 23/04/1987, 4087]; T) Your date of birth in dDMm format [example for 23/04/1987, 3204]; U) Your date of birth in dDmM format [example for 23/04/1987, 3240]; V) Your date of birth in DdmM format [example for 23/04/1987, 2340]; W) Your date of birth in yyYY format [example for 23/04/1987, 8719]; X) Your date of birth in YYyd format [example for 23/04/1987, 1983]; Y) Your date of birth in yydD format [example for 23/04/1987, 8732]; Z) Your date of birth in MmYy format [example for 23/04/1987, 0498].	Example for yydD=8732_{10} a) $8732_{10} \rightarrow 10001000011100_2$ Remainder 8732 /2=4366 0 (↑) LSB* 4366 /2=2183 0 (↑) 2183 /2=1091 1 (↑) 1091 /2=545 1 (↑) 545 /2=272 1 (↑) 272 /2=136 0 (↑) 136 /2=68 0 (↑) 68 /2=34 0 (↑) 34 /2=17 0 (↑) 17 /2=8 1 (↑) 8 /2=4 0 (↑) 4 /2=2 0 (↑) 2 /2=1 0 (↑) 1 /2=0 1 (↑) MSB** * LSB - Least Significant Bit ** MSB - Most Significant Bit b) $010\ 001\ 000\ 011\ 100_2 \rightarrow 21034_8$ c) $0010\ 0010\ 0001\ 1100_2 \rightarrow 221C_{16}$ d) $10001000011100_2 \rightarrow SUM(n_i*b^n{}_i) \rightarrow$ $1*2^{14}+0*2^{13}+0*2^{12}+\ldots+1*2^2+0*2^1+0*2^0 =$ $8192+0+0+0+512+0+0+0+0+16+8+4+0+0=$ 8732_{10}

Nr	Assignment, Instruction, Variant of Task	Detailed Answer Example
3.2	Convert a Decimal real number to a) Binary integer & b) Binary fraction with an accuracy of 8 digits after RADIX point. • Choose your variant x = 1st letter of your Surname in the English alphabet. • Use your date of birth to number generation from date template DdMmYYyy. • In [Square Brackets] Your see examples for 23 Apr 1987 Year = 23041987 = DdMmYYyy. • Example for YURIY **L**I, born 23/04/1987, task variant = “L” with template dD.yy are selected and number = 32.87_{10} are generated. A) Your date of birth in Dd.Mm format [example for 23/04/1987, 23.04]; B) Your date of birth in Mm.Dd format [example for 23/04/1987, 04.23]; C) Your year of birth in YY.yy format [example for 23/04/1987, 19.87]; D) Your year of birth in Yy.yD format [example for 23/04/1987, 98.70]; E) Your year of birth in YY.Dd format [example for 23/04/1987, 19.23]; F) Your year of birth in YY.yD format [example for 23/04/1987, 98.72]; G) Your date of birth in YY.Mm format [example for 23/04/1987, 19.04]; H) Your date of birth in Mm.yy format [example, 04.87]; I) Your date of birth in Dd.yy format [example, 23.87]; J) Your date of birth in Dd.Yy format [example, 23.98]; K) Your date of birth in Dd.YY format [example, 23.19]; L) Your date of birth in dD.yy format [example, 32.87]; M) Your date of birth in dD.Yy format [example, 32.98]; N) Your date of birth in dD.YY format [example, 32.19]; O) Your date of birth in mM.Dd format [example, 40.23]; P) Your date of birth in mM.dD format [example, 40.32]; Q) Your date of birth in mM.YY format [example, 40.19]; R) Your date of birth in mM.Yy format [example, 40.98]; S) Your date of birth in mM.yy format [example, 40.87]; T) Your date of birth in dD.Mm format [example, 32.04]; U) Your date of birth in dD.mM format [example, 32.40]; V) Your date of birth in Dd.mM format [example, 23.40]; W) Your date of birth in yy.YY format [example, 87.19]; X) Your date of birth in YY.yd format [example, 19.83]; Y) Your date of birth in yy.dD format [example, 87.32]; Z) Your date of birth in Mm.Yy format [example, 04.98].	Example for Dd.Mm=32.87_{10} a) $32_{10} \rightarrow 100000_2$ Remainder 32 /2=16 **0** (↑) LSB* 16 /2=8 0 (↑) 8 /2=4 0 (↑) 4 /2=2 0 (↑) 2 /2=1 0 (↑) 1 /2=0 1 (↑) MSB** b) $0.87_{10} \rightarrow 0.11011110_2$ Remainder Carry 0.87×2 =**1**.74 =0.74 **1** (↓) MSB** 0.74×2 =**1**.48 =0.48 **1** (↓) 0.48×2 =**0**.96 =0.96 **0** (↓) 0.96×2 =**1**.92 =0.92 **1** (↓) 0.92×2 =**1**.84 =0.84 **1** (↓) 0.84×2 =**1**.68 =0.68 **1** (↓) 0.68×2 =**1**.36 =0.36 **1** (↓) 0.36×2 =**0**.72 =0.72 **0** (↓) LSB* $32.87_{10} \rightarrow 100000.11011110_2$ * LSB - Least Significant Bit ** MSB - Most Significant Bit

Nr	Assignment, Instruction, Variant of Task	Detailed Answer Example
3.3	Convert Binary integer number to a) Decimal, b) Octal, c) Hexadecimal. • Choose your variant x = 2nd letter of your Name in the English alphabet. • Example for YURIY LI, task variant = “U” (change to ?) are selected. A) 1101011_2 N) 1101100_2 ?) 1100101_2 B) 1011001_2 O) 1110101_2 C) 1111010_2 P) 1000111_2 D) 1001100_2 Q) 1011100_2 E) 1100111_2 R) 1110111_2 F) 1011101_2 S) 1101101_2 G) 1111011_2 T) 1111010_2 H) 1011100_2 U) 1101100_2 I) 1010011_2 V) 1011011_2 J) 1111100_2 W) 1111101_2 K) 1100011_2 X) 1011111_2 L) 1000001_2 Y) 1010000_2 M) 1100010_2 Z) 1100100_2	Example for 1100101_2 a) $1100101_2 \rightarrow 1\times2^6+ 1\times2^5+ 0\times2^4+ +0\times2^3+ 1\times2^2+ 0\times2^1+ 1\times2^0 = =64+32+4+1=101_{10}$ b) $001\ 100\ 101_2 \rightarrow 145_8$ c) $0110\ 0101_2 \rightarrow 65_{16}$
Nr	Assignment, Instruction, Variant of Task	Detailed Answer Example
3.4	You need a) add two binary numbers; b) check your answer by converting Bin→Dec. • Choose your variant x = 2nd letter of your Surname in the English alphabet. • Example for YURIY LI, task variant = “I” (change to ?) are selected. A) $110101_2 + 110001_2$ N) $101110_2 + 110000_2$?) $111011_2 + 101011_2$ B) $101101_2 + 100001_2$ O) $111011_2 + 111100_2$ C) $111100_2 + 110000_2$ P) $110111_2 + 111011_2$ D) $100110_2 + 101110_2$ Q) $111100_2 + 101101_2$ E) $110011_2 + 111011_2$ R) $110110_2 + 111111_2$ F) $101111_2 + 110111_2$ S) $101101_2 + 110110_2$ G) $111101_2 + 111100_2$ T) $111111_2 + 100001_2$ H) $101110_2 + 110110_2$ U) $111011_2 + 101101_2$ I) $101001_2 + 101101_2$ V) $111011_2 + 101001_2$ J) $111110_2 + 111111_2$ W) $111111_2 + 100001_2$ K) $110001_2 + 101110_2$ X) $111100_2 + 101101_2$ L) $100001_2 + 111100_2$ Y) $100111_2 + 101111_2$ M) $110000_2 + 110001_2$ Z) $101101_2 + 101111_2$	Example for $111011_2 + 101011_2$ a) Bin 0111011_2 0101011_2 + 1110110_2 Carry Up 1100110_2 Sum b) Dec $0111011_2 \rightarrow 32+16+8+2+1=59_{10}$ $0101011_2 \rightarrow 32+8+2+1=43_{10}$ $1100110_2 \rightarrow 64+32+4+2=102_{10}$ 059_{10} 043_{10} + 110_{10} Carry Up 102_{10} Sum

Nr	Assignment, Instruction, Variant of Task	Detailed Answer Example
3.5	Find value for a Boolean expression. • Choose your variant x = 3rd letter of your Name in the English alphabet or 3rd letter of your Surname in the English alphabet (if you Name is short). • Example for YU**R**IY LI, task variant = "R" (change to ?) are selected. A) NOT((NOT(s) NOR NOT(h)) NAND (c NOR NOT(h))) for s=false; h=false; c=true. B) NOT((NOT(t) XOR m) NAND (j AND NOT(m))) for t=true; m=true; j=false. C) NOT((NOT(r) XOR NOT(b)) NOR (NOT(g) OR b)) for r=false; b=false; g=false. D) NOT((NOT(m) NOR h) OR (b XOR h)) for m=true; h=true; b=true. E) NOT((NOT(d) XOR b) NAND (k AND b)) for d=false; b=false; k=false. F) NOT((NOT(e) NAND NOT(p)) XOR (v NOR p)) for e=true; p=true; v=false. G) NOT((NOT(m) NOR NOT(g)) NOR (NOT(i) NOR NOT(g))) for m=false; g=true; i=false H) (r OR NOT(e)) NAND (NOT(x) AND NOT(e)) for r=true; e=true; x=true. I) (NOT(e) OR g) XOR (NOT(w) XOR g) for e=true; g=true; w=false. J) (z XOR NOT(w)) NAND (NOT(y) OR w) for z=true; w=false; y=true. K) (j AND NOT(k)) OR (q NOR k) for j=true; k=true; q=true. L) (k XOR t) AND (NOT(e) AND NOT(t)) for k=true; t=false; e=false. M) (x OR NOT(q)) NAND (b NOR q) for x=true; q=false; b=true. N) (n OR i) AND (NOT(p) NOR i) for n=true; i=true; p=true. O) NOT((NOT(q) OR NOT(i)) AND (v NOR NOT(i))) for q=true; i=false; v=true. P) (k OR NOT(c)) OR (g XOR NOT(c)) for k=true; c=true; g=true. Q) (NOT(n) NOR NOT(y)) XOR (s AND y) for n=false; y=false; s=true. R) (NOT(p) XOR t) XOR (NOT(d) OR t) for p=true; t=false; d=true. S) NOT((NOT(k) NOR c) OR (NOT(s) XOR c)) for k=true; c=false; s=false. T) (NOT(g) OR u) NAND (p NAND NOT(u)) for g=true; u=false; p=false. U) NOT((NOT(a) NAND y) AND (i AND NOT(y))) for a=true; y=false; i=false. V) (m NAND NOT(q)) NOR (NOT(n) XOR NOT(q)) for m=true; q=false; n=true. W) NOT((NOT(p) OR NOT(s)) NOR (h XOR s)) for p=false; s=true; h=false. X) NOT((y NOR NOT(j)) OR (NOT(k) XOR NOT(j))) for y=true; j=true; k=false. Y) (a AND NOT(f)) AND (NOT(h) AND NOT(f)) for a=true; f=false; h=true. Z) NOT((s NOR NOT(x)) NAND (NOT(r) NOR NOT(x))) for s=true; x=true; r=true. ?) NOT((c AND k) NAND (NOT(r) NAND k)) for c=true; k=false; r=false	Example for NOT((c AND k) NAND (NOT(r) NAND k)) for c=true; k=false; r=false NOT((1 AND 0) NAND (NOT(0) NAND 0))= = NOT(0 NAND (NOT(0) NAND 0))= = NOT(0 NAND (1 NAND 0))= = NOT(0 NAND 1)= = NOT(1)= 0 = false Boolean Algebra Laws: true = 1 1 XOR 1 = 0 false = 0 1 XOR 0 = 1 NOT(1) = 0 0 XOR 1 = 1 NOT(0) = 1 0 XOR 0 = 0 1 AND 1 = 1 1 NAND 1 = 0 1 AND 0 = 0 1 NAND 0 = 1 0 AND 1 = 0 0 NAND 1 = 1 0 AND 0 = 0 0 NAND 0 = 1 1 OR 1 = 1 1 NOR 1 = 0 1 OR 0 = 1 1 NOR 1 = 0 0 OR 1 = 1 1 NOR 1 = 0 0 OR 0 = 0 1 NOR 1 = 1 Boolean Algebra operations order: 1. From left to right 2. Brackets 3. NOT 4. AND, NAND, XOR 5. OR, NOR

4. Guidelines

4.1. Introduction

4.1.1. Positional Number Systems.

Many number counting systems are used today. For example clocks and compasses use the ancient **Babylonian number system** based on 60. Why? Because 60 is easier to divide into equal segments, it can be evenly divided by 1,2,3,4,5,6,10,12,15, 20 and 30. This is much better for applications such as time, or degrees of angle than a base of 10, which can only be divided into equal parts by 1, 2 and 5.

Commonly used numeral systems include:

Base/radix	Name	Description
2	Binary numeral system	Used internally by nearly all computers, is base 2. The two digits are "0" and "1", expressed from switches displaying OFF and ON respectively. Used in most electric counters.
8	Octal system	Used occasionally in computing. The eight digits are "0–7" and represent 3 bits (2^3).
10	Decimal system	The most used system of numbers in the world, is used in arithmetic. Its ten digits are "0–9". Used in most mechanical counters.
12	Duodecimal (dozenal) system	Sometimes advocated due to divisibility by 2, 3, 4, and 6. It was traditionally used as part of quantities expressed in dozens and grosses.
16	Hexadecimal system	Often used in computing as a more compact representation of binary (1 hex digit per 4 bits). The sixteen digits are "0–9" followed by "A–F" or "a–f".
20	Vigesimal	Traditional numeral system in several cultures, still used by some for counting.
60	Sexagesimal system	Originated in ancient Sumer and passed to the Babylonians.[3] Used today as the basis of modern circular coordinate system (degrees, minutes, and seconds) and time measuring (minutes, and seconds) by analogy to the rotation of the Earth.

For a larger list, see list of numeral systems.
https://en.wikipedia.org/wiki/List_of_numeral_systems

Radix or **base** is the number of unique digits, including the digit zero, used to represent numbers in a positional numeral system.

In any standard positional numeral system, a number is conventionally written as $(x)_y$ with x as the string of digits and y as its base.

For example, $(100)_{10}$, $(100)_2$, $(100)_8$.

For decimal system used simple form without $()_{10}$

Radix Point. When writing a number, the digits used give its value, but the number is 'scaled' by its RADIX POINT.

For example, 456.2_{10} is ten times bigger than 45.62_{10} although the digits are the same.

Exponents numbers write. A decimal number can be considered as the sum of the values of its individual digits, where each digit has a value dependent on its position within the number (the value of the column). Each digit in the number is multiplied by the system radix raised to a power depending on its position relative to the radix point. Decimal value of any system number find from Exponents write:

$$DEC = SUM(n_i*b^i) = n_{m-1}*b^{m-1} + n_{m-1}*b^{m-1} +\ldots+ n_2*b^2 + n_1*b^1 + n_0*b^0 +\ldots+ n_{-1}*b^{-1} + n_{-2}*b^{-2} +\ldots+ n_k*b^k$$

For example,

- Decimal exponents $(409.28)_{10}$ $= 4*10^2 + 0*10^1 + 9*10^0 + 2*10^{-1} + 8*10^{-2}$
- Hexadecimal exponents $(98.2)_{16}$ $= (9 \times 16^1) + (8 \times 16^0) + (2 \times 16^{-1})$
- Octal exponents $(56.2)_8$ $= (5 \times 8^1) + (6 \times 8^0) + (2 \times 8^{-1})$
- Binary Exponents $(10.1)_2$ $= (1 \times 2^1) + (0 \times 2^0) + (1 \times 2^{-1})$

Some column values of different number systems				
Decimal	1000	100	10	1
Binary	8	4	2	1
Octal	512	64	8	1
Hexadecimal	4096	256	16	1

Floating Point Notation. Move radix point to the left/right by increasing/decreasing the exponent, without altering the value of the number. Example:

$$102.6_{10} = 102.6 \times 10^0 = 10.26 \times 10^1 = 1.026 \times 10^2 = .1026 \times 10^3$$

Normalized Form. By putting the radix point at the front of the number, and keeping it there by changing the exponent, calculations become easier to do electronically, in any radix. For example $11011_2 \times 2^3$ is the normalized form of the binary number 110.11_2.

4.1.2. Electronic storage of numbers.

Because numbers in electronic systems are stored as binary digits, and a binary digit can only be 1 or 0, it is not possible to store the radix point within the number. Therefore the number is stored in its normalized form and the exponent is stored separately. The exponent is then reused to restore the radix point to its correct position when the number is displayed.

In electronics systems a single binary digit is called a bit (short for **B**inary Dig**IT**), but as using a single digit would seriously limit the math's that could be performed, binary bits are normally used in groups:

- 1 nibble = 4 bits
- 1 byte = 8 bits
- Multiple bytes, such as 16 bits, 32 bits, 64 bits are usually called 'words', e.g. a 32 bit word. The length of the word depends on how many bits can be physically handled or stored by the system at one time.

4 Bit Binary Representation

Decimal	MSB	4 Bit Binary		LSB
	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
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
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

When a number is stored in an electronic system, it is stored in a memory location having a fixed number of binary bits.

Some of these memory locations are used for general storage whilst others, having some special function, are called registers.

Therefore a decimal number such as 13, which can be expressed in four binary bits as 1101_2 becomes 00001101_2 when stored in an eight-bit register. This is achieved by adding four NON SIGNIFICANT ZEROS to the left of the most significant '1' digit.

Using this system, a binary register that is n bits wide can hold 2^n values. Therefore:

- 8 bit register can hold 2^8 values = 256 values (0 to 255)
- 4 bit register can hold 2^4 values = 16 values (0 to 15)

Larger numbers problem

Numbers containing more digits than the register can hold are broken up into register sized groups and stored in multiple locations.

Special Flag Register, MSB (Most Significant Bit), LSB (Least Significant Bit) used for different Binary Arithmetic: Signed Binary, Ones Complement, Twos Complement for Overflow Problems solution:

1. When adding large positive numbers.
2. When adding large negative numbers.
3. When subtracting a large negative number from a large positive number.
4. When subtracting a large positive number from a large negative number.

4.2. CONVERTING BETWEEN POSITIONAL NUMERAL SYSTEMS

4.2.1. Between BIN, OCT, HEX Number Converting.

BIN → OCT	OCT → BIN
Group binary digits into sets of three, starting with the least significant (rightmost) digits. Then, look up each group in a table and write octal result:	Simply look up each octal digit to obtain the equivalent group of three binary digits and write binary result:

Binary:	000	001	010	011	100	101	110	111
Octal:	0	1	2	3	4	5	6	7

```
BIN: 11100101 = 011 100 101
OCT:             3   4   5 = 345₈
```

```
OCT: 345₈ = 3   4   5
BIN:       011 100 101 = 11100101₂
```

BIN → HEX	HEX → BIN
Group binary digits into sets of four, starting with the least significant (rightmost) digits. Then, look up each group in a table and write hexadecimal result:	Simply look up each hexadecimal digit to obtain the equivalent group of four binary digits and write binary result:

Binary:	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Hexadecimal:	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

```
BIN: 11100101₂ = 1110 0101
HEX:               E    5 = E5₁₆
```

```
HEX: E5₁₆ = E    5
BIN:       1110 0101 = 11100101₂
```

OCT → BIN → HEX	HEX → BIN → OCT

When converting from octal to hexadecimal, it is often easier to first convert the octal number into binary and then from binary into hexadecimal.	When converting from hexadecimal to octal, it is often easier to first convert the hexadecimal number into binary and then from binary into octal.

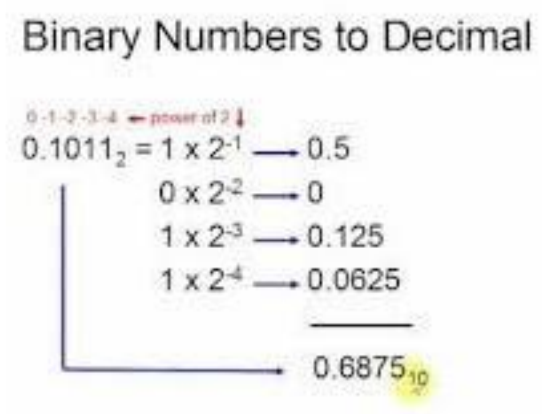
4.2.2. Binary to Decimal Number Converting.

Have more methods. One method involves addition and multiplication.

1. Start the decimal result at 0.
2. Remove the most significant binary digit (leftmost) and add it to the result.
3. If all binary digits have been removed, you’re done. Stop.
4. Otherwise, multiply the result by 2.
5. Go to step 2.

Here is an example of converting 11100000001 binary to decimal:

Binary Digits	Operation	Decimal Result	Operation	Decimal Result
11100000001	+1	1	× 2	2
1100000001	+1	3	× 2	6
100000001	+1	7	× 2	14
00000001	+0	14	× 2	28
0000001	+0	28	× 2	56
000001	+0	56	× 2	112
00001	+0	112	× 2	224
0001	+0	224	× 2	448
001	+0	448	× 2	896
01	+0	896	× 2	1792
1	+1	1793	done.	



4.2.2.1. Quick Converting BIN to DEC

The most commonly encountered number systems are binary and hexadecimal, and a quick method for converting to decimal is to use a simple table showing the column weights, as shown in Table:

Bit	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	**radix**	2^{-1}	2^{-2}	2^{-3}	2^{-4}
Value	1024	512	256	128	64	32	16	8	4	2	1	.	0.5	0.25	0.125	0.0625
BIN:	0	1	0	0	1	0	0	0	0	1	1	.	0	1	0	1
DEC:	512+64+2+1=579.3125															

4.2.3. Converting from Decimal to any Radix

To convert a decimal integer number (a decimal number in which any fractional part is ignored) to any other radix, all that is needed is to continually divide the number by its radix, and with each division, write down the remainder.

When read from bottom to top, the remainder will be the converted result.

DEC → BIN	DEC → OCT	DEC → HEX	Fraction Party DEC → BIN
$13_{10} \rightarrow 1101_2$	$86_{10} \rightarrow 126_8$	$2861_{10} = B2D_{16}$	$0.8329_{10} = 11010101_2$
2)13 Remainder 2) 6 1 2) 3 0 2) 1 1 0 1	8)86 Remainder 8)10 6 8) 1 2 0 1	Remainder 16)2861 Dec. Hex. 16) 178 13 D 16) 11 2 2 0 11 B	0.8329×2 =**1**.6650 =0.6658 carry **1** (MSB) 0.6258×2 =**1**.3300 =0.3316 carry **1** (↓) 0.3316×2 =**0**.6632 =0.6632 carry **0** (↓) 0.6632×2 =**1**.3264 =0.3264 carry **1** (↓) 0.3264×2 =**0**.6528 =0.6528 carry **0** (↓) 0.6528×2 =**1**.3056 =0.3056 carry **1** (↓) 0.3056×2 =**0**.6112 =0.6112 carry **0** (↓) 0.6112×2 =**1**.2224 =0.2224 carry **1** (LSB)

4.2.3.1. Converting Decimal Numbers with Fractions to Binary.

In electronics this is not normally done, as binary does not work well with fractions. The method used is to get rid of the radix (decimal) point by NORMALISING the decimal fraction using FLOATING POINT arithmetic and restored to its correct position when the result get.

However, for the sake of completeness, here is a method for converting decimal fractions to binary fractions.

1. For Integer Party any method described above is used to covert the integer party.
2. For Fraction Party.

The fractional part of the number is found by successively multiplying the given fractional part of the decimal number repeatedly by 2 (×2), noting the carries in forward order, until the value becomes “0” producing the binary equivalent. So if the multiplication process produces a product greater than 1, the carry is a “1” and if the multiplication process produces a product less than “1”, the carry is a “0”.

Note also that if the successive multiplication processes does not seem to be heading towards a final zero, the fractional number will have an infinite length or until the equivalent number of bits have been obtained, for example 8-bits. or 16-bits, etc. depending on the degree of accuracy required.

4.3. Binary Arithmetic Systems

4.3.1. Binary Addition Rules

Arithmetic rules for binary numbers are quite straightforward, and similar to those used in decimal arithmetic. The rules for addition of binary numbers are: Notice that in Fig., 1+1 = (1)0 requires a 'carry' of 1 to the next column. Remember that binary $10_2 = 2_{10}$ decimal	0 + 0 = 0 0 + 1 = 1 1 + 0 = 1 1 + 1 = (1)0 Rules for Binary Addition
Example:	Decimal Binary 2 10 1 + 01 + Answer 3 11 Simple Binary Addition
Binary addition is carried out just like decimal, by adding up the columns, starting at the right and working column by column towards the left Just as in decimal addition, it is sometimes necessary to use a 'carry', and the carry is added to the next column. For example, in Fig. right when two ones in the right-most column are added, the result is 2_{10} or 10_2, the least significant bit of the answer is therefore 0 and the 1 becomes the carry bit to be added to the 1 in the next column..	Decimal Binary 3 0011 1 + 0001 + Carry 0110 4 0100 Binary Addition with Carry

More theory about Binary Arithmetic read on site article (p.p. 1.3, 1.4, 1.5):https://github.com/Arailym-ray/AITU-Operating-system-concepts/blob/e36152455d767a90b611b3ce375aa58e28cd2aa4/week%201/Binary%20Arithmetic.pdf

4.3. BOOLEAN ALGEBRA

Read theory and make study examples on site:

https://github.com/Arailym-ray/AITU-Operating-system-concepts/blob/796a0ab83e2fbb9e5daa9a6a57dd1409bf6a875f/week%201/Boolean%20Algebra%20Tasks%20Generator.html