Numeral Systems

Subjects:

- **Numeral System**
- **Positional systems**
- **Decimal**
- **Binary**
- **Octal**

Introduction

A **numeral system** (or **system of numeration**) is a writing system for expressing numbers, that is a mathematical notation for representing numbers of a given set, using digits or other symbols in a consistent manner.

It can be seen as the context that allows the symbols "**11**" to be interpreted as the **binary** symbol for ***three***, the **decimal** symbol for ***eleven***, or a symbol for other numbers in different **bases**.

Ideally, a **numeral system will**:

- ❑Represent a useful set of numbers (e.g. all integers, or rational numbers);
- ❑Give every number represented a unique representation (or at least a standard representation);
- ❑Reflect the algebraic and arithmetic structure of the numbers.

- ✓ A **number** is a mathematical object used to count, label, and measure.
- ✓ A **digit** is a symbol (a numeral symbol such as "3" or "7") used in combinations (such as "37") to represent numbers in positional numeral systems.

0123456789

The ten digits of the Western Arabic numerals, in order of value

Positional notation

Positional notation or **place-value notation** is a method of representing or encoding numbers. Positional notation is distinguished from other notations (such as Roman numerals) for its use of the same symbol for the different orders of magnitude (for example, the "ones place", "tens place", "hundreds place"). This greatly simplified arithmetic and led to the quick spread of the notation across the world.

With the use of a radix point, the notation can be extended to include fractions and the numeric expansions of real numbers. ***The Hindu-Arabic numeral system is an example for a positional notation, based on the number 10****.*

In a positional **base-*b*** numeral system (with ***b*** a natural number greater than **1** known as the **radix**), ***b*** basic symbols (or digits) corresponding to the first ***b*** natural numbers including zero are used. To generate the rest of the numerals, the position of the symbol in the figure is used. The symbol in the last position has its own value, and as it moves to the left its value is multiplied by ***b***.

Positional notation

For example, in the **decimal system** (**base 10**), the numeral **4327** means

$$(4\times10^3) + (3\times10^2) + (2\times10^1) + (7\times10^0),$$

noting that $10^0 = 1$.

In general, if b is the base, one writes a number in the numeral system of base b by expressing it in the form

$$a_n b^n + a_{n-1} b^{n-1} + a_{n-2} b^{n-2} + \ldots + a_0 b^0$$

and writing the enumerated digits $a_n a_{n-1} a_{n-2} \ldots a_0$ in descending order. The digits are natural numbers between **0** and $b - 1$, inclusive.

If a text (such as this one) discusses multiple bases, and if ambiguity exists, the base (itself represented in base 10) is added in subscript to the right of the number, like this: $\mathbf{number}_{base}$*. Unless specified by context, numbers without subscript are considered to be decimal.*

Positional notation

By using a dot to divide the digits into two groups, one can also write fractions in the positional system. For example, the **base-2** numeral **10.11** denotes

$$\mathbf{1\times2^{1} + 0\times2^{0} + 1\times2^{-1} + 1\times2^{-2} = 2.75}.$$

In general, numbers in the base ***b*** system are of the form:

$$(a_n a_{n-1} \cdots a_1 a_0 . c_1 c_2 c_3 \cdots)_b = \sum_{k=0}^{n} a_k b^k + \sum_{k=1}^{\infty} c_k b^{-k}.$$

The numbers b^k and b^{-k} are the weights of the corresponding digits. The ***position k*** is the logarithm of the corresponding *weight* **w**, that is

$$k = \log_b w = \log_b b^k$$

The number of tally marks required in the unary numeral system for ***describing the weight*** would have been **w**. In the positional system, the number of digits required to describe it is only

$$k + 1 = \log_b w + 1$$

for $k \geq 0$.

Positional notation

E.g. to describe the weight **1000** then four digits are needed since

$$\log_{10} 1000 + 1 = 3 + 1$$

The number of digits required to ***describe the position*** is

$$\log_b k + 1 = \log_b \log_b w + 1$$

(in positions 1, 10, 100,... only for simplicity in the decimal example).

Position	3	2	1	0	−1	−2	. . .
Weight	b^3	b^2	b^1	b^0	b^{-1}	b^{-2}	. . .
Digit	a_3	a_2	a_1	a_0	c_1	c_2	. . .
Decimal example weight	1000	100	10	1	0.1	0.01	. . .
Decimal example digit	4	3	2	7	0	0	. . .

Note that a number has a terminating or repeating expansion if and only if it is rational; this does not depend on the base. A number that terminates in one base may repeat in another (thus $\mathbf{0.3_{10} = 0.0100110011001\ldots_2}$). An irrational number stays aperiodic (with an infinite number of non-repeating digits) in all integral bases. Thus, for example in base 2, $\mathbf{\pi = 3.1415926\ldots_{10}}$ can be written as the aperiodic $\mathbf{11.001001000011111\ldots_2}$.

Computer Number Systems

Representing Information in Computers

All the different types of information in computers can be represented using binary code:

- Numbers;
- Letters of the alphabet and punctuation marks;
- Microprocessor instruction;
- Graphics/Video;
- Sound.

Why binary?

- The original computers were designed to be high-speed calculators.
- The designers needed to use the electronic components available at the time.
- The designers realized they could use a **simple coding system** - ***the binary system*** - to represent their numbers.

Bits and Bytes

- A **binary digit** is a single numeral in a binary number.
- Each **1** and **0** in the number below is a **binary digit**:

1 0 0 1 0 1 0 1

- The term "**binary digit**" is commonly called a "**bit**."
- **Eight bits** grouped together **is** called **a** "**byte**."

Computer Number Systems

Computer Number Systems are:

- ❑**Decimal Numbers**
- ❑**Binary Numbers**
- ❑**Hexadecimal Numbers**

The study of number systems is useful due to the fact that number systems other than the familiar **decimal** (**base 10**) **number system** are used in the computer field.

Digital computers internally use the **binary** (**base 2**) **number system** to represent data and perform arithmetic calculations. The binary number system is very efficient for computers, but not for humans. Representing even relatively small numbers with the binary system requires working with long strings of ones and zeroes.

The **hexadecimal** (**base 16**) **number system** (often called "**hex**" for short) provides us with a shorthand method of working with binary numbers. **One digit in hex corresponds to four binary digits** (bits), so the **internal representation of one byte can be represented either by eight binary digits or two hexadecimal digits**.

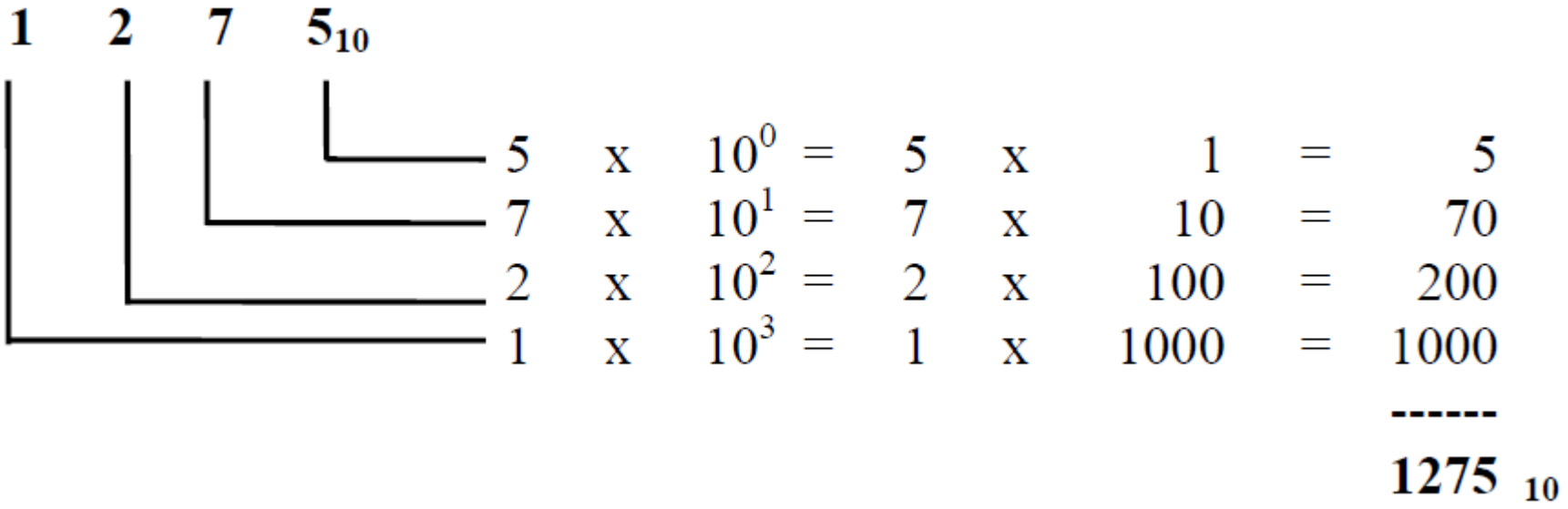
Less commonly used is the **octal** (**base 8**) **number system**, where one digit in octal corresponds to three binary digits (bits).

Decimal Number System

- The prefix "***deci***-" stands for 10.
- The decimal number system is a **Base 10 number system**:
  - ✓ There are **10 symbols** that represent quantities:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

- ✓ Each place value in a decimal number is a power of **10**.



Binary Numbers

The Binary Number System

- The prefix "**bi**-" stands for **2**.
- The binary number system is a **Base 2** number system:
  - ✓ There are **2 symbols** that represent quantities:

0, 1

- ✓ **Each place value in a binary number is a power of 2.**

From right to left, the successive positions of the binary number are weighted **1**, **2**, **4**, **8**, **16**, **32**, **64**, etc. A list of the first several powers of **2** follows:

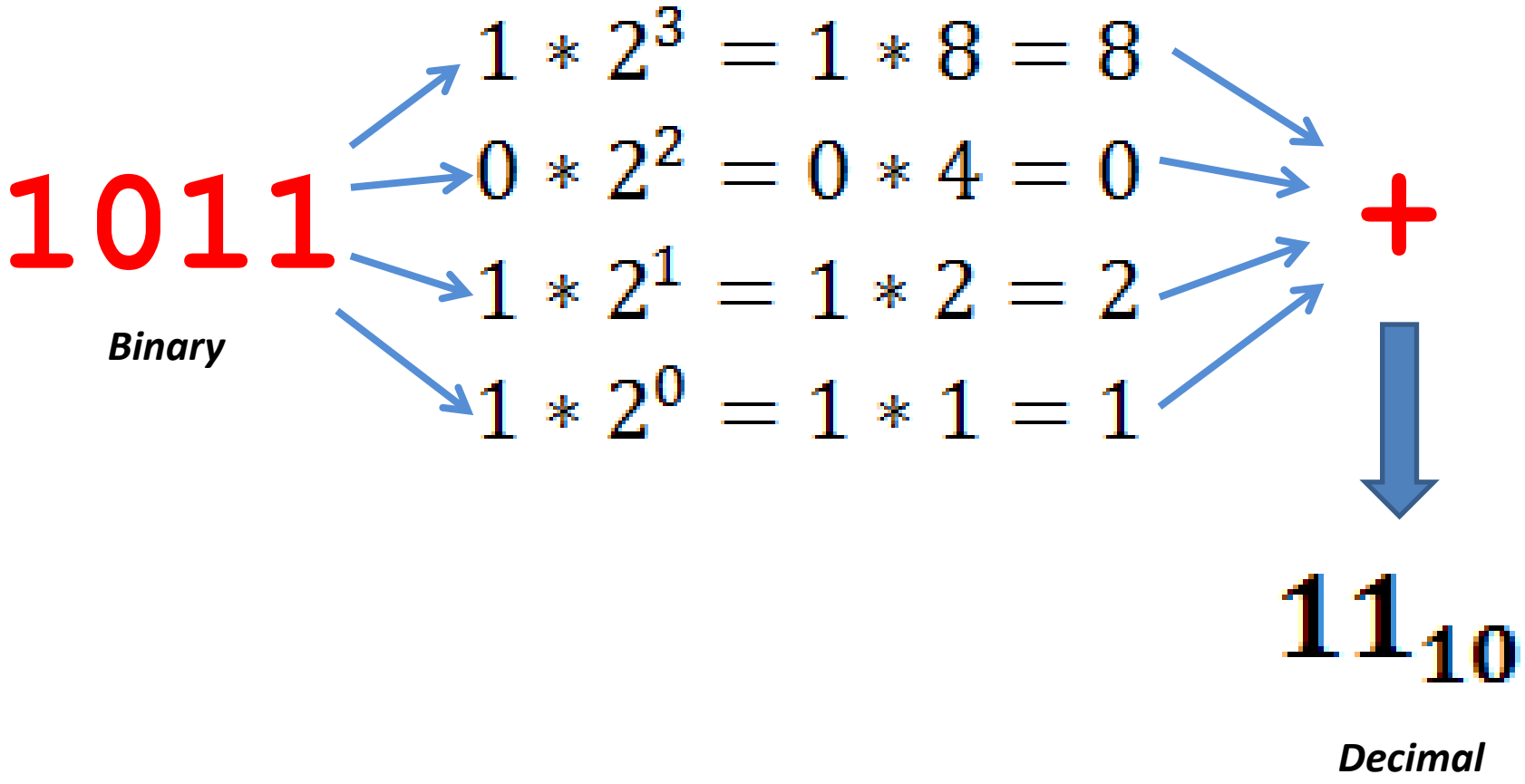
2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}
1	2	4	8	16	32	64	128	256	512	1024	2048

Binary Numbers

For reference, the following table shows the decimal numbers **0** through **31** with their binary equivalents:

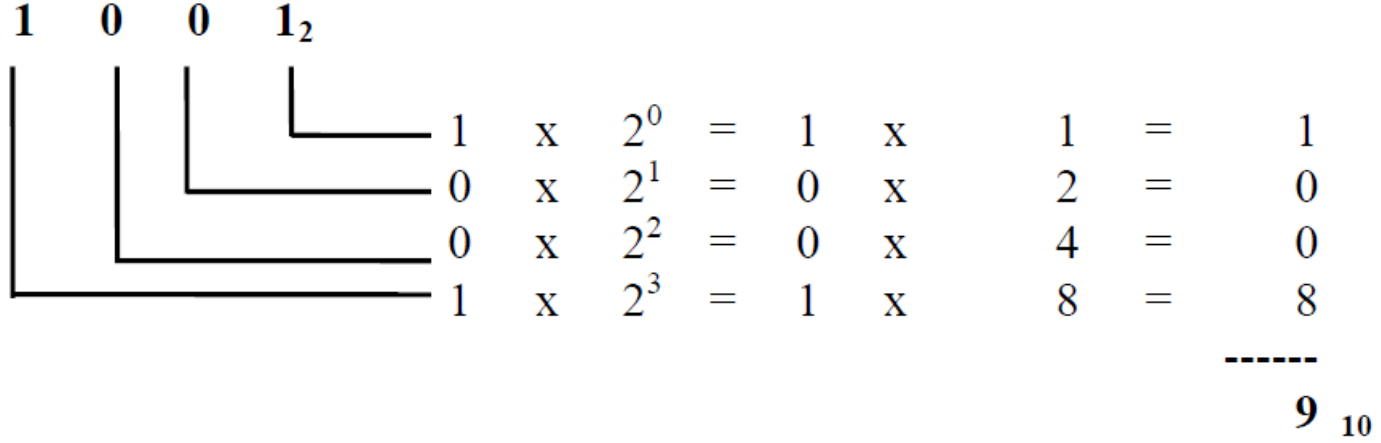
Decimal	Binary	Decimal	Binary
0	0	16	10000
1	1	17	10001
2	10	18	10010
3	11	19	10011
4	100	20	10100
5	101	21	10101
6	110	22	10110
7	111	23	10111
8	1000	24	11000
9	1001	25	11001
10	1010	26	11010
11	1011	27	11011
12	1100	28	11100
13	1101	29	11101
14	1110	30	11110
15	1111	31	11111

Binary Numbers

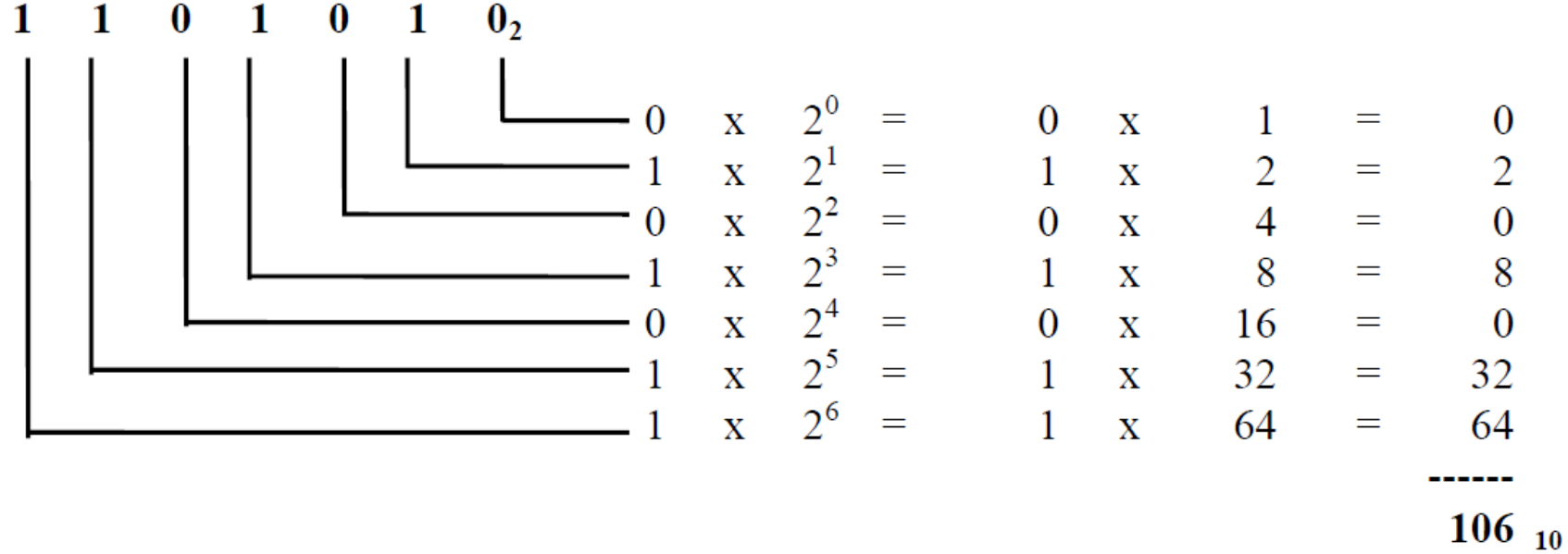


Converting a Binary Number to a Decimal Number

To determine the value of a binary number (**1001**, for example), **we can expand the number using the positional weights** as follows:



Here's another example to determine the value of the binary number **1101010**:



Converting a Decimal Number to a Binary Number

To convert a decimal number to its binary equivalent, **the remainder method** can be used. (This method can be used to convert a decimal number into any other base.)

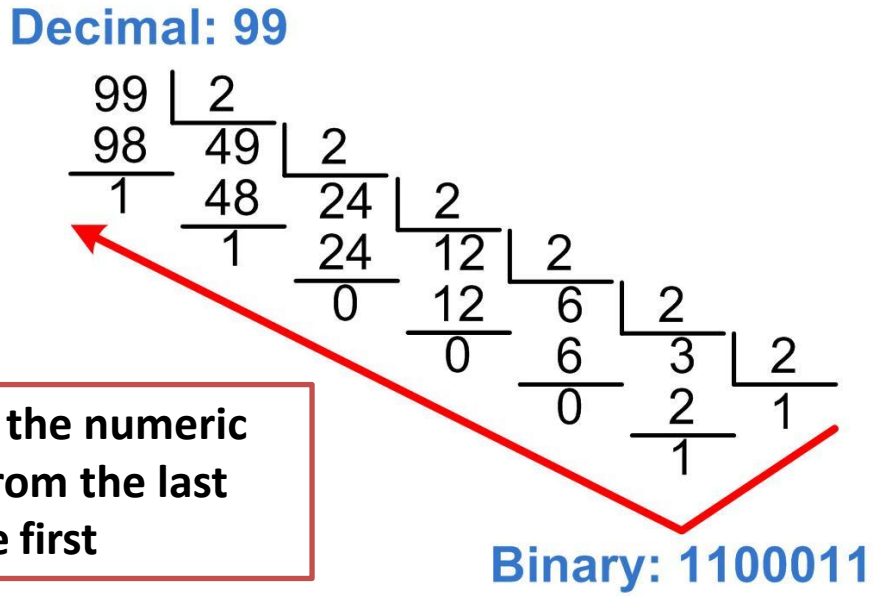
The remainder method involves the following **four steps**:

1) **Divide** the decimal number by the base (in the case of binary, divide by 2).
2) **Indicate the remainder** to the right.
3) **Continue dividing** into each quotient (and indicating the remainder) **until** the divide operation produces **a zero quotient**.
4) **The base 2 number is the numeric remainder reading from the last division to the first** (if you start at the bottom, the answer will read from top to bottom).

Converting a Decimal Number to a Binary Number

Example 1: Convert the decimal number **99** to its binary equivalent:

1) Divide 2 into 99. The quotient is 49 with a remainder of 1; indicate the 1 on the right.
2) Divide 2 into 49 (the quotient from the previous division). The quotient is 24 with a remainder of 1, indicated on the right.
3) Divide 2 into 24. The quotient is 12 with a remainder of 0, as indicated .
4) Divide 2 into 12. The quotient is 6 with a remainder of 0, as indicated.
5) Divide 2 into 6. The quotient is 3 with a remainder of 0, as indicated.
6) Divide 2 into 3. The quotient is 1 with a remainder of 1, as indicated.
7) Divide 2 into 1. The quotient is 0 with a remainder of 1, as indicated. Since the quotient is 0, stop here.



The base 2 number is the numeric remainder reading from the last division to the first

Binary Addition

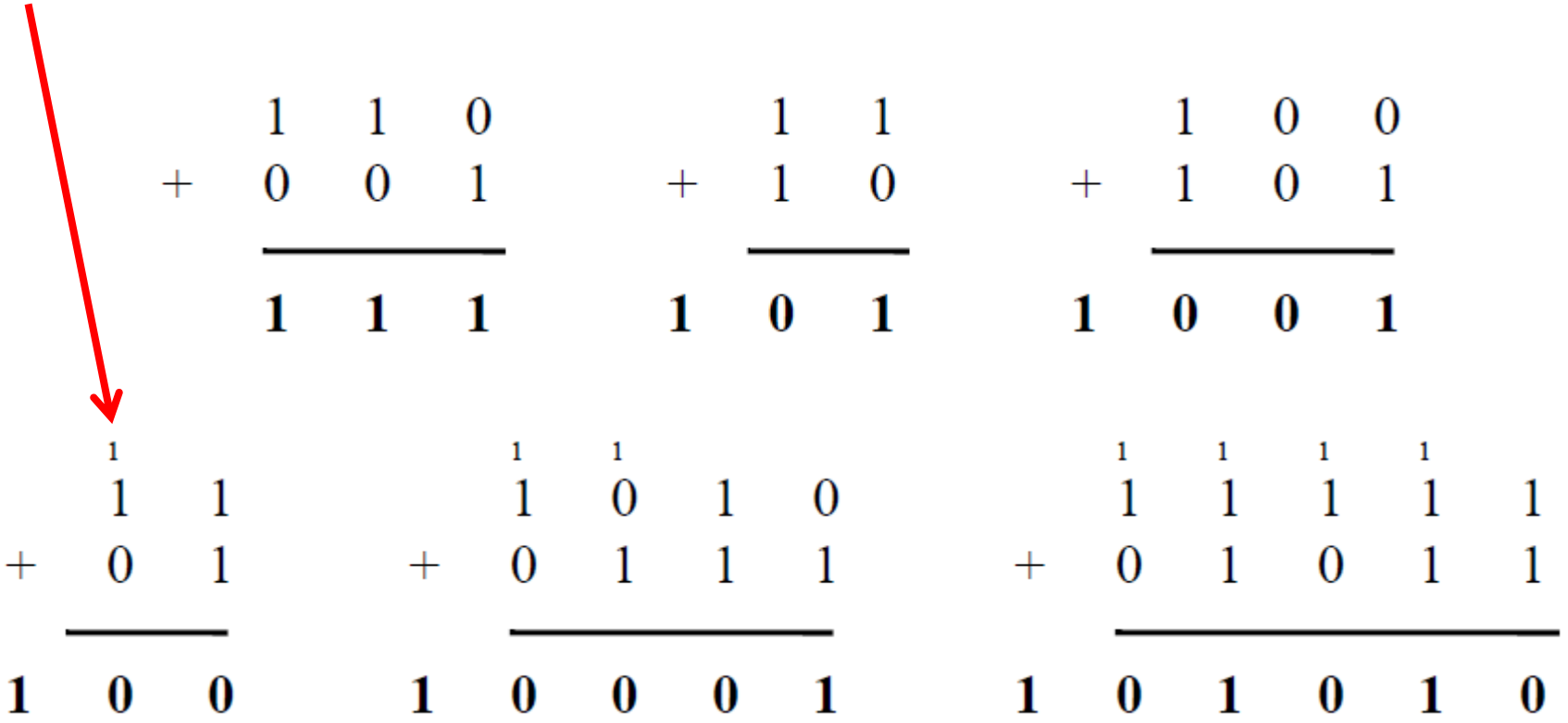
Adding two binary numbers together is easy, keeping in mind the following **four addition rules**:

(1)	**0**	+	**0**	=	**0**
(2)	**0**	+	**1**	=	**1**
(3)	**1**	+	**0**	=	**1**
(4)	**1**	+	**1**	=	**10**

Note in the last example that it was necessary to "*carry the 1*". After the first two binary counting numbers, **0** and **1**, all of the binary digits are used up. In the decimal system, we used up all the digits after the tenth counting number, 9. The same method is used in both systems to come up with the next number: place a zero in the "ones" position and start over again with one in the next position on the left. In the decimal system, this gives ten, or **10**. In binary, it gives 10_2, which is read "**one-zero, base two**."

Binary Addition

Consider the following binary addition problems and ***note where it is necessary to carry the 1***:



Subtraction Using Complements

Subtraction in any number system **can be accomplished through the use of complements**.

A complement is a number that is used to represent the negative of a given number.

When two numbers are to be subtracted, the ***subtrahend*** can either be subtracted directly from the ***minuend*** (as we are used to doing in decimal subtraction) **or**, **the complement of the subtrahend can be added to the minuend to obtain the difference**. When the latter method is used, the addition will produce a high-order (leftmost) one in the result (a "carry"), which must be dropped.

*This is how the computer performs subtraction: it is very efficient for the computer to use the same "**add**" circuitry to do both addition and subtraction; thus, when the computer "subtracts", it is really adding the complement of the subtrahend to the minuend.*

To understand complements, consider a mechanical register, such as a car mileage indicator, being rotated backwards. A five-digit register approaching and passing through zero would read as follows:

00005
00004
00003
00002
00001
00000
99999
99998
99997
etc.

Subtraction Using Complements

It should be clear that the number **99998** corresponds to **-2**. Furthermore, if we add

$$\begin{array}{r} 00005 \\ +\ 99998 \\ \hline 1\ 00003 \end{array}$$

and ignore the carry to the left, we have effectively formed the operation of subtraction: **5 - 2 = 3**.

The number **99998** is called the ***ten's complement of 2***. **The ten's complement of any decimal number may be formed by subtracting each digit of the number from 9, then adding 1 to the least significant digit of the number formed**.

In the example above, subtraction with the use of complements was accomplished as follows:

(1) We were dealing with a five-digit subtrahend that had a value of **00002**. First, each digit of the subtrahend was subtracted from 9 (this preliminary value is called the ***nine's complement of the subtrahend***):

$$\begin{array}{ccccccccc} 9 & & 9 & & 9 & & 9 & & 9 \\ -\ 0 & - & 0 & - & 0 & - & 0 & - & 2 \\ \hline 9 & & 9 & & 9 & & 9 & & 7 \end{array}$$

Subtraction Using Complements

(2) Next, **1** was added to the nine's complement of the subtrahend (**99997**) giving the ten's complement of subtrahend (**99998**):

$$\begin{array}{r} 9\ 9\ 9\ 9\ 7 \\ +\ \ \ \ \ 1 \\ \hline 9\ 9\ 9\ 9\ 8 \end{array}$$

(3) The ten's complement of the subtrahend was added to the minuend giving **100003**. **<u>The leading (carried) 1 was dropped</u>**, effectively **performing the subtraction of 00005 - 00002 = 00003**.

$$\begin{array}{r} 0\ 0\ 0\ 0\ 5 \\ +\ 9\ 9\ 9\ 9\ 8 \\ \hline \cancel{1}\ 0\ 0\ 0\ 0\ 3 \end{array}$$

The answer can be checked by making sure that 2 + 3 = 5.

Binary Subtraction

We will use the ***complement method*** to perform subtraction in binary and in the sections on octal and hexadecimal that follow. As mentioned above, the use of complemented binary numbers makes it possible for the computer to add or subtract numbers using only circuitry for addition - the computer performs the subtraction of **A - B** by adding **A + (two's complement of B) and then dropping the carried 1**.

The steps for **subtracting two binary numbers** are as follows:

(1) Compute the one's complement of the subtrahend by subtracting each digit of the subtrahend by 1. A shortcut for doing this is to simply reverse each digit of the subtrahend - the **1's** become **0's** and the **0's** become **1's**.

(2) Add 1 to the one's complement of the subtrahend to get the two's complement of the subtrahend.

(3) Add the two's complement of the subtrahend to the minuend and drop the high-order 1. This is your difference.

Binary Subtraction

Example 1: Compute 11010101 – 1001011

(1) Compute the one's complement of **1001011** by subtracting each digit from **1** (note that a leading zero was added to the 7-digit subtrahend to make it the same size as the 8-digit minuend):

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

(Note that the one's complement of the subtrahend causes each of the original digits to be reversed.)

(2) Add 1 to the one's complement of the subtrahend, giving the two's complement of the subtrahend:

1	0	1	1	0	1	0	0
						+	1
1	0	1	1	0	1	0	1

Binary Subtraction

(3) Add the two's complement of the subtrahend to the minuend and drop the high-order 1, giving the difference:

$$
\begin{array}{cccccccccc}
 & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & & {\scriptstyle 1} & & {\scriptstyle 1} & & \\
 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \\
+ & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & \\
\hline
\cancel{1} & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 &
\end{array}
$$

So **11010101 – 1001011 = 10001010**.

The answer can be checked by making sure that **1001011 + 10001010 = 11010101**.

The Octal Number System

The same principles of positional number systems we applied to the decimal and binary number systems can be applied to the **octal number system**.

However, **the base of the octal number system is *eight***, so **each position of the octal number represents a successive power of eight**. From right to left, the successive positions of the octal number are weighted **1**, **8**, **64**, **512**, etc.

A list of the first several powers of **8** follows:

8^0	8^1	8^2	8^3	8^4	8^5
1	8	64	512	4096	32768

The Octal Number System

For reference, the following table shows the decimal numbers 0 through 31 with their octal equivalents:

Decimal	Octal	Decimal	Octal
0	0	16	20
1	1	17	21
2	2	18	22
3	3	19	23
4	4	20	24
5	5	21	25
6	6	22	26
7	7	23	27
8	10	24	30
9	11	25	31
10	12	26	32
11	13	27	33
12	14	28	34
13	15	29	35
14	16	30	36
15	17	31	37

Converting an Octal Number to a Decimal Number

To determine the value of an octal number (**367_8**, for example), we can expand the number using the positional weights as follows:

$3\ 6\ 7_8$

$$
\begin{array}{rrrrrrrrr}
7 & \times & 8^0 & = & 7 & \times & 1 & = & 7 \\
6 & \times & 8^1 & = & 6 & \times & 8 & = & 48 \\
3 & \times & 8^2 & = & 3 & \times & 64 & = & 192 \\
 & & & & & & & & \text{------} \\
 & & & & & & & & \mathbf{247}_{10}
\end{array}
$$

Here's another example to determine the value of the octal number **1601_8**:

$1\ 6\ 0\ 1_8$

$$
\begin{array}{rrrrrrrrr}
1 & \times & 8^0 & = & 1 & \times & 1 & = & 1 \\
0 & \times & 8^1 & = & 0 & \times & 8 & = & 0 \\
6 & \times & 8^2 & = & 6 & \times & 64 & = & 384 \\
1 & \times & 8^3 & = & 1 & \times & 512 & = & 512 \\
 & & & & & & & & \text{------} \\
 & & & & & & & & \mathbf{897}_{10}
\end{array}
$$

Converting a Decimal Number to an Octal Number

To convert a decimal number to its octal equivalent, the remainder method (the same method used in converting a decimal number to its binary equivalent) can be used.

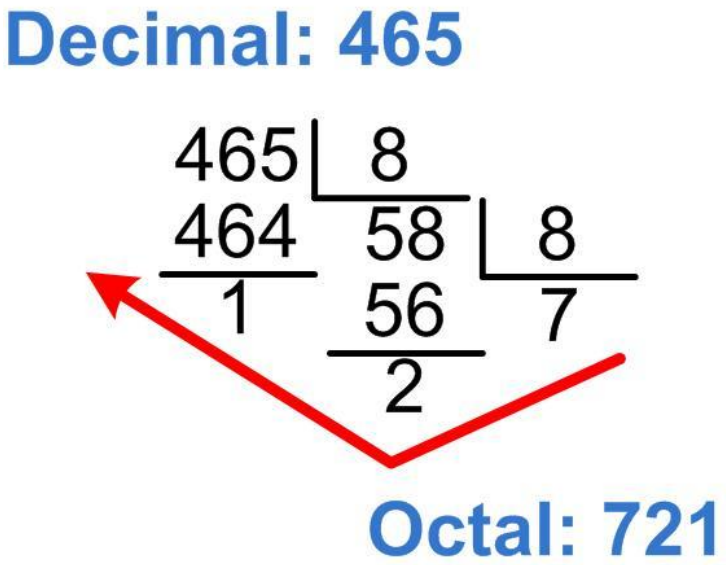
To review, the **remainder method** involves the following **four steps**:

(1) Divide the decimal number by the base (*in the case of **octal**, divide by **8***).
(2) Indicate the remainder to the right.
(3) Continue dividing into each quotient (and indicating the remainder) until the divide operation produces a zero quotient.
(4) The base 8 number is the numeric remainder reading from the last division to the first (if you start at the bottom, the answer will read from top to bottom).

Converting a Decimal Number to an Octal Number

Example 1: Convert the decimal number 465_{10} to its octal equivalent:

1) Divide 8 into 465. The quotient is 58 with a remainder of 1; indicate the 1 on the right.
2) Divide 8 into 58 (the quotient from the previous division). The quotient is 7 with a remainder of 2, indicated on the right.
3) Divide 8 into 7. The quotient is 0 with a remainder of 7, as indicated. Since the quotient is 0, stop here.



The answer, reading the remainders from top to bottom, is **721,** so **465_{10} = 721_8.**

Octal Addition

Octal addition is performed just like decimal addition, except that **if a column of two addends produces a sum greater than 7, you must subtract 8 from the result, put down that result, and carry the 1**. Remember that there are no such digits as "**8**" and "**9**" in the octal system, and that $8_{10} = 10_8$, $9_{10} = 11_8$, etc.

Example 1: Add $543_8 + 121_8$ (*no carry required*):

$$\begin{array}{rrrr} & 5 & 4 & 3 \\ + & 1 & 2 & 1 \\ \hline & 6 & 6 & 4 \end{array}$$

Example 2: Add $7652_8 + 4574_8$ (*carries required*):

$$\begin{array}{rrrrr} & {\scriptstyle 1} & {\scriptstyle 1} & & \\ & 7 & 6 & 5 & 2 \\ + & 4 & 5 & 7 & 4 \\ \hline & 12-8=4 & 12-8=4 & 12-8=4 & \\ \mathbf{1} & \mathbf{4} & \mathbf{4} & \mathbf{4} & \mathbf{6} \end{array}$$

Octal Subtraction

We will use the complement method to perform octal subtraction. **The steps for subtracting two octal numbers** are as follows:

(1) Compute the seven's complement of the subtrahend by subtracting each digit of the subtrahend by 7.
(2) Add 1 to the seven's complement of the subtrahend to get the eight's complement of the subtrahend.
(3) Add the eight's complement of the subtrahend to the minuend and drop the high-order 1. This is your difference.

Example 1: Compute $7526_8 - 3142_8$

(1) Compute the seven's complement of **3142_8** by subtracting each digit from **7**:

7	7	7	7
- 3	- 1	- 4	- 2
4	6	3	5

Octal Subtraction

(2) Add 1 to the seven's complement of the subtrahend, giving the eight's complement of the subtrahend:

$$\begin{array}{r} 4\ 6\ 3\ 5 \\ +\ \ \ \ 1 \\ \hline 4\ 6\ 3\ 6 \end{array}$$

(3) Add the eight's complement of the subtrahend to the minuend and drop the high-order 1, giving the difference:

$$\begin{array}{rcccc} & {\scriptstyle 1} & & {\scriptstyle 1} & \\ & 7 & 5 & 2 & 6 \\ + & 4 & 6 & 3 & 6 \\ \hline & 12-8=4 & 11-8=3 & & 12-8=4 \\ \hline \mathbf{1} & \mathbf{4} & \mathbf{3} & \mathbf{6} & \mathbf{4} \end{array}$$

So $\mathbf{7526_8 - 3142_8 = 4364_8}$.

The answer can be checked by making sure that $\mathbf{3142_8 + 4364_8 = 7526_8}$.

Thanks for attention